

A Model-driven Approach to Service Policies

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

Abstract

A Service represents an underlying capability offered by a service provider. A service description describes two facets of a service – the service functionality (capability on-offer) and the terms at which the service is offered (terms of offer). The capability on-offer satisfies the goal of a service consumer under the constraints of the terms of offer. Service Policies are used to define the terms of offer of a service offering. Policies could potentially apply to service-level, domain-level or technical (infrastructural) aspects. In this paper, we present a systematic model-driven development approach to deal with service policies from the perspective of a service provider. Our approach addresses the entire development spectrum of service policies. It addresses definition of service policies using visual models and attaching these policy models to service capability description models. It also addresses transforming these policy models to executable specifications and finally enforcing these policies during service invocation.

1 INTRODUCTION

Service-oriented computing paradigm deals with organizing and utilizing distributed capabilities under the control of different ownership domains[1]. A service represents an underlying capability offered by a service provider that meets the goals of one or more service consumers. Every service description, has a functional part representing the underlying capability on-offer and a non-functional part representing the terms in which the capability is offered (a.k.a. terms of offer). The functionality satisfies a goal of the consumer under the constraints of the terms of offer. The terms of offer of a service which applies constraints on the service capability is defined using service policies. In general, a service policy defines constraints or conditions of use of a service[1]. Consider the example of a *ShippingService*, the capability on-offer is to ship packages from one place to another, the terms of offer could be the time-to-delivery and rates of shipping. In this paper, we address four broad issues related to service policy development.

Firstly, service policies are currently focussed towards technical or infrastructural aspects such as security, trust and reliable messaging. We take a broad-based view of service policy development. In our view, service policies would address three-levels of

aspects – *service-level* (availability, pricing, promotions and quality of service), business or *domain-level* (compliance, industry regulations) and *technical-level* (security, trust). While technical policies are defined by IT experts, the service-level and domain policies would be defined by domain experts.

Secondly, we address independent development of service policies. Traditionally, service descriptions have had a bias towards describing service functionality as opposed to non-functional terms of offer (e.g. WSDL[2] for web service description). Lately, there have been efforts to address description of non-functional terms of offer in service descriptions (Features & Properties in WSDL 2.0 and the WS-Policy framework[3]). However, service development approaches still consider service policies in the confined context of the underlying service functionality which they constraint. Instead, service policies could be developed independently by domain experts and could later be applied on a chosen set of services in the services portfolio through well-defined quantification and fine-tuning. For example, the security expert could define encryption and authentication policies independently and later apply it to selected services in the portfolio.

Thirdly, our proposed development approach is based on model-driven development[4] of service policies. We choose a model-driven approach for the following reasons:

- Current specifications to defining web service policies are at varying levels of expressivity, complexity and more importantly maturity. A model-driven approach would help raise the level of abstraction and tackle the *Evoloving Standards Problem*[5].
- Policies on service-level and domain-level aspects are always independently defined by domain experts rather than the IT experts. Domain experts would prefer visual models rather than having to deal with XML specifications to define service policies.

Lastly, we address unintrusive change of service policies. As service policies represent the terms of offer of a service, they are frequently altered as opposed to the capability on offer. Reason, the same capability on offer is offered under different terms of offer for different customers and customer segments. When service policies change, the change should be unintrusive – without requiring major changes to underlying service realization mechanisms.

In essence, our development approach to service policy is holistic. It addresses entire spectrum of service policy development, is broad-based with involvement of domain experts and is model-driven. In this paper, our contributions are the following:

- We provide mechanism to concretely define service-level, domain-level and technical aspects and their vocabulary on which policy constraints are applied.
- We define a MOF2-based[6] service policy metamodel to visually define service policies
- We discuss transformation of the policy models defined to concrete executable specifications



- We also discuss about policy enforcement during service invocation.

Example

Throughout this paper, we use the example of a fictitious *ShippingService* provided by FedEx® (all the scenarios and the services presented in this paper are fictitious). The service represents an underlying capability of shipping an item from one place to another. The service capability view[5](fig. 1) provides a functional view of the *ShippingService*.

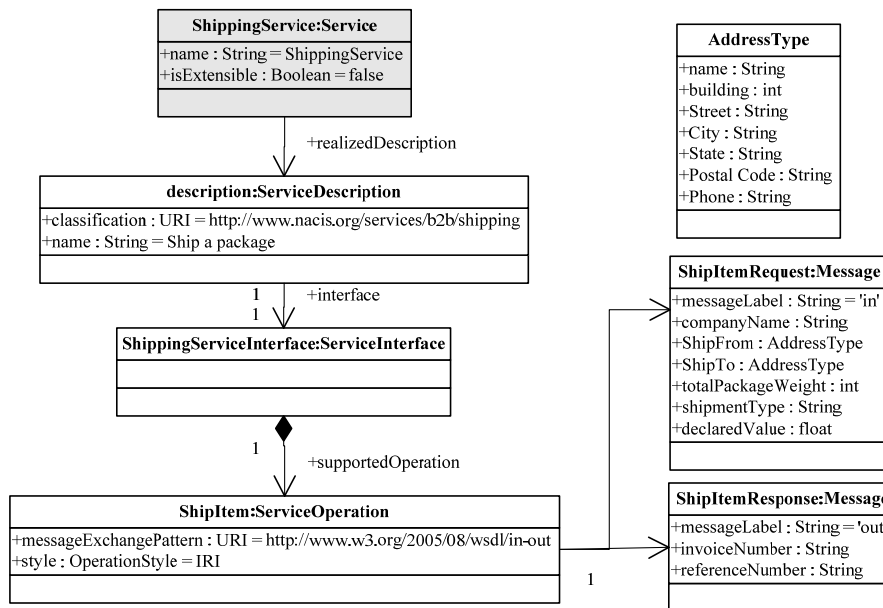


Fig. 1. Shipping Service – Service Capability View

Fig. 3 presents the corresponding WSDL 2.0[2] functional description of the *Shipping Service*. The *ShippingService* defines a *ShipItem* operation which supports shipping a package from one place to another. In addition to the *ShipItem* operation, there could be other operations (fig. 2) such as *Get Rates & Transit Times* – an operation which provides rates and transit times between two locations and *Schedule Pick-up* – an operation which supports pick-up of items from consumer’s location. In addition, FedEx® also offers the *Package Tracking Service* which supports tracking a shipment through its *Track Shipment* operation.

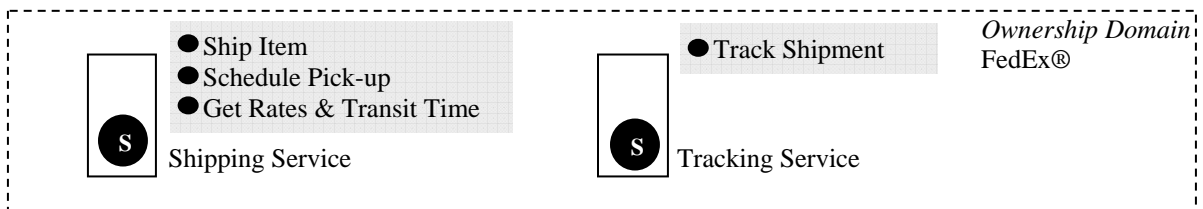


Fig. 2. FedEx® Ownership Domain and the Services

```

- <description xmlns="http://www.w3.org/ns/wsdl" xmlns:tns="http://service.fictitious.com/shipping/fedex"
  targetNamespace="http://service.fictitious.com/shipping/fedex">
  <documentation>Fictitious FedEx Shipping Service</documentation>
- <types>
- <xsd:schema xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  elementFormDefault="qualified" targetNamespace="http://service.fictitious.com/shipping/fedex">
- <xsd:element name="AddressType">
+ <xsd:complexType>
</xsd:element>
- <xsd:element name="ShipItemRequest">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element minOccurs="1" maxOccurs="1" name="CompanyName" type="xsd:string" />
  <xsd:element minOccurs="1" maxOccurs="1" name="ShipFrom" type="tns:AddressType" />
  <xsd:element minOccurs="1" maxOccurs="1" name="ShipTo" type="tns:AddressType" />
  <xsd:element minOccurs="1" maxOccurs="1" name="TotalPackageWeight" type="xsd:integer" />
  <xsd:element minOccurs="1" maxOccurs="1" name="ShipmentType" type="xsd:string" />
  <xsd:element minOccurs="1" maxOccurs="1" name="DeclaredValue" type="xsd:float" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
- <xsd:element name="ShipItemResponse">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element minOccurs="0" maxOccurs="1" name="InvoiceNumber" type="xsd:string" />
  <xsd:element minOccurs="0" maxOccurs="1" name="Reference Number" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</types>
- <interface name="ShippingServiceInterface">
- <operation name="ShipItem" pattern="http://www.w3.org/ns/wsdl/in-out">
  <documentation>Ship Item Operation</documentation>
  <input element="tns:ShipItemRequest" />
  <output element="tns:ShipItemResponse" />
</operation>
</interface>
+ <binding xmlns:wsoap="http://www.w3.org/ns/wsdl/soap" name="ShippingServiceEndpoint"
  interface="tns:ShippingServiceInterface" type="http://www.w3.org/ns/wsdl/soap" wsoap:version="1.2"
  wsoap:protocol="http://www.w3.org/2006/01/soap11/bindings/HTTP"/>
+ <service name="ShippingService" interface="tns:ShippingServiceInterface">
</description>

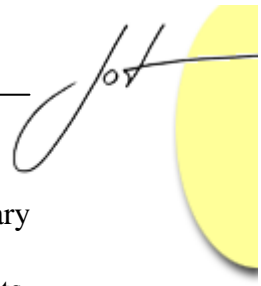
```

Fig. 3. WSDL 2.0 Snippet for the Shipping Service

2 GENERIC POLICY FRAMEWORK

The most important aspect of our model-driven development approach of service policy is a MOF2-compliant service policy metamodel. The metamodel with appropriate tooling, supports the domain experts as well as IT experts to define policies using visual models. In order to arrive at a policy metamodel, it is important to understand the generic policy model – an abstract model for service policies. The generic policy model consists of four functional layers to describe service policies[7] (fig. 4):

- *Vocabulary Specification Layer*: Deals with specification of the vocabulary associated with various policy domains representing independent aspects. These aspects could be technical, service-level or domain-level aspects. Vocabulary consists of vocabulary items, and their applicable values which would then be used in service policies. It also involves specifying the semantics and syntax associated



with the vocabulary items. Constraints are always specified on these vocabulary items in the constraint specification layer.

- *Constraint Specification Layer*: Deals with specification of policy constraints, which would ideally be constraints on the agreeable values of vocabulary items. Constrained vocabulary items are assertions which are the building blocks of a policy.
- *Policy Specification Layer*: Deals with specification of acceptable combinations of the constrained vocabulary items. Each combination of constrained vocabulary items represents an alternative.
- *Bindings Specification Layer*: Deals with specification of application of the service policies on various policy subjects. Policy subjects could be services, ownership domains as well as individual operations. Binding layer supports the quantification and fine-tuning of policies for different policy subjects.

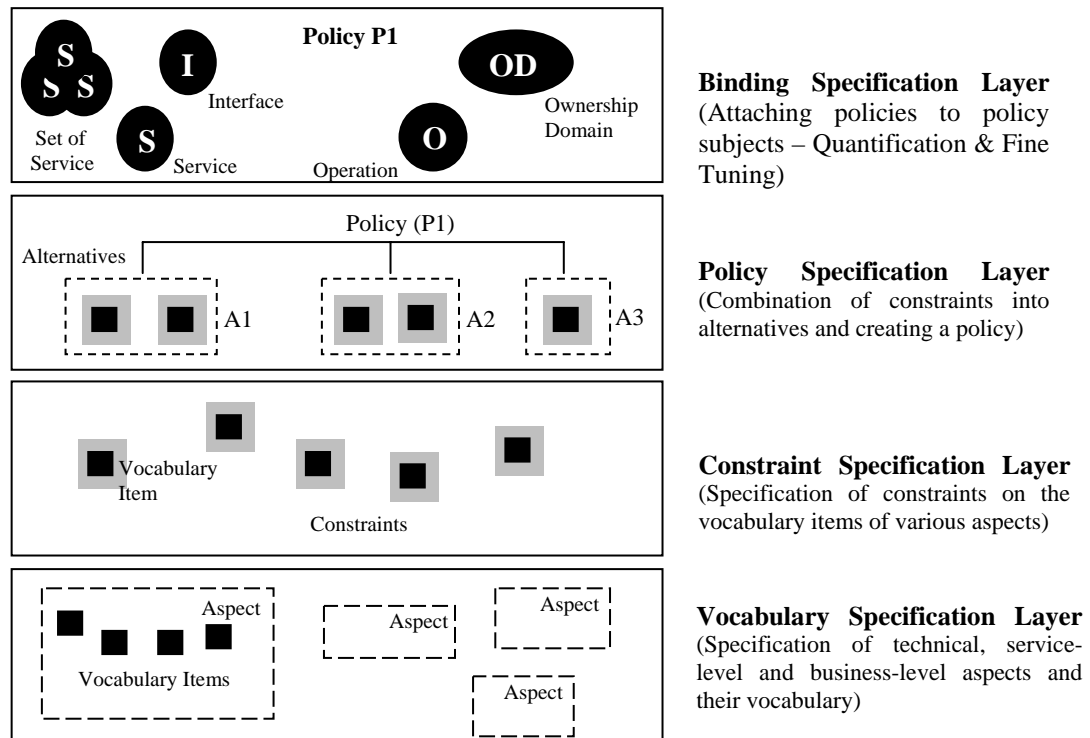
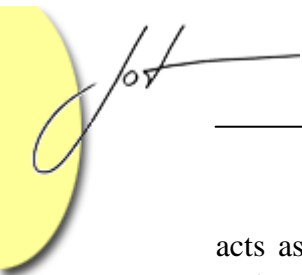


Fig. 4. Generic Policy Model

This generic policy model is largely representative of several policy proposals which are based on propositional logic with the assertions representing an indivisible unit and their combinations through conjunction or disjunction representing a policy.

Service Policy Metamodel

The MOF2-compliant service policy metamodel supports model-driven development of service policies based on the generic policy model. Our service policy metamodel (fig.5)



acts as a foundation for modeling service policies by domain experts. The goal of the metamodel is two-folds a) have minimum number of elements with maximum expressivity b) reduce the representational gap for domain experts to use it. Our next step would be to create an UML-profile[8] for the services policy metamodel to leverage existing UML tooling. We explain the key classes, associations and constraints if any, in the metamodel below. The 'Core' in the fig. 5 represents the UML2:: Infrastructure[9] package.

Service Policy: A service policy defines a set of enforceable constraints which would be applied on a policy subject. It presents these enforceable constraints as a set of alternatives. A service policy reflects the point of view of a service participant who is the owner of the policy. Since our focus is service-provider centric, the service participant in our case is the service provider. Service Policy extends the *Core: NamedElement*.

Policy Subject: A policy subject represents an entity on which a policy is applied. A policy subject extends the *Core: Element*. The policy subjects could be Ownership Domain (supports physical or administrative partition of services); Service, Service Interface, Service Operation, Message and Interaction Point (end point). 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 related policy subjects together in order to apply the same policy on them. More than one policy could also be applied on the policy scope. The policy scope supports quantification of service policy by domain experts.

Service Participant: A service participant could be a service provider or a consumer. A service provider policy is communicated to the consumer along with the service description.

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'. Every policy alternative would have more than one policy assertion.

Policy Assertion: Every policy alternative would have one or more policy assertions. A policy assertion is a constraint applied on a vocabulary item (constrained element) of a particular domain. The policy assertion specifies the allowable range, range of values, or set of values for a vocabulary item. It has an operator associated with it – the operator is a predicate operator used to describe constraints. The policy assertion could be optional in nature and could represent a 'preference' of the service participant. It extends the *Core: Constraint*.

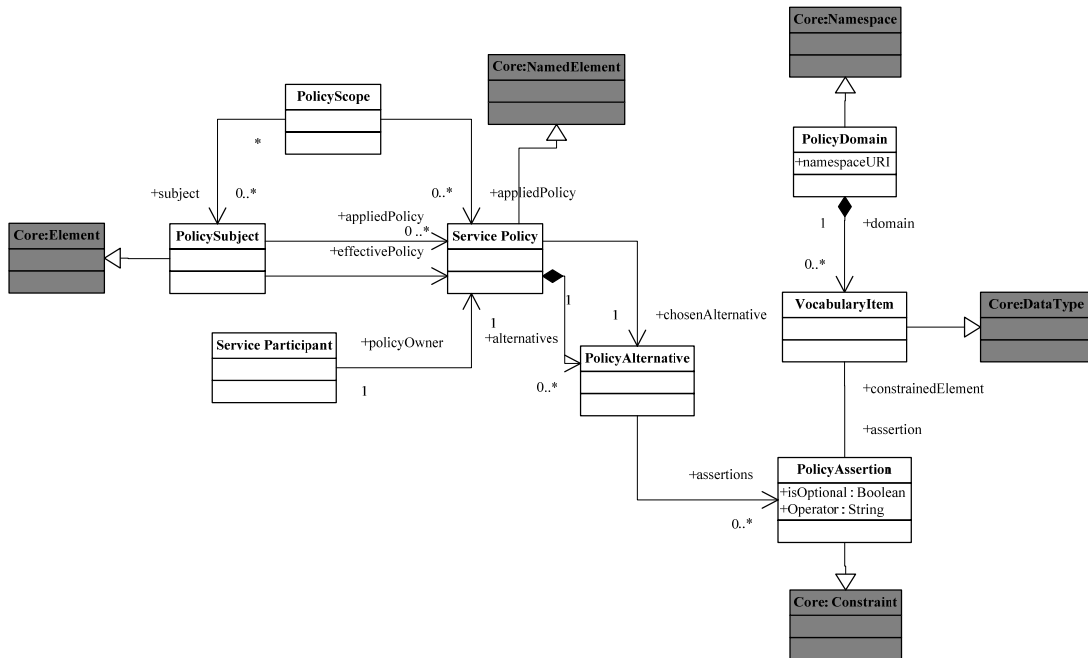
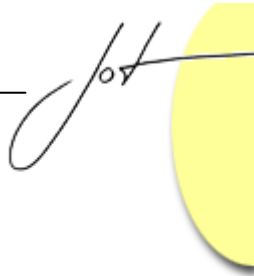


Fig. 5. Service Policy Metamodel

Policy Domain: A policy domain represents a grouping of assertions belonging to a particular aspect such as pricing, availability, security & trust etc. A policy domain is identified by a name and a namespace URI and it extends the *Core: Namespace*.

Vocabulary Item: A vocabulary item represents semantics associated with a particular aspect and belongs to a policy domain. Every vocabulary item has a set of applicable values. The vocabulary items for a particular domain (aspect) are defined by the domain expert. Vocabulary Item extends the *Core: DataType*.

Policy Assertion

A policy assestion is an atomic unit of a service policy. It represents a constraint on a vocabulary item representing different technical, service-level or domain-level (business) aspects. A policy assertion could be represented as:

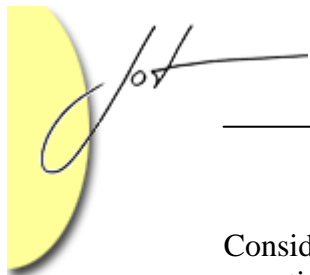
$$\text{Policy Assertion} = \{ \text{VI}, \text{PO}, \text{AV}, \text{C} \}$$

Where, VI - vocabulary item representing a particular aspect

PO - predicate operator

AV - accepted value / values or range of values

C – category of the assertion (Mandatory / preference)



Consider examples of security assertion (use of kerberos[10] security token) and pricing assertion (cost of service-access) below:

Security Token Assertion = { 'Security Token', 'Equals', 'Kerberos', 'Preference' }

Pricing Assertion = { 'Cost of Access', 'Equals', '1 EUR', 'Mandatory' }

3 VOCABULARY SPECIFICATION – DEFINING POLICY DOMAINS AND THEIR VOCABULARY

Vocabulary specification involves identifying policy domains and describing their vocabulary. The policy domain vocabulary involves defining *Vocabulary Items* to describe the policy domain. The vocabulary items would have a type and a range of acceptable values. The policy assertions apply constraints on the vocabulary items by specifying agreeable values for the vocabulary items. Policy domains address aspects that represent independent concerns such as security, pricing etc. These concerns are crosscutting in nature as they apply to a set of services in the services portfolio and not just a single service. From aspect-oriented software development[11] literature, we refer to these crosscutting concerns represented by the policy domains as aspects. We group these aspects as *technical*, *service-level* or *domain-level* (business) aspects. It is important to identify these aspects early in the life-cycle of service development and define their vocabulary in order to use them in service policies.

Policy Domain Aspect Catalog

Since the technical, service-level and domain-level aspects are reusable assets in services development, it is important to document and catalog these aspects. Notably, this catalog of aspects is extensible and could be extended to create additional aspects either by extending existing aspects or by adding new aspects. We have defined a standard schema (table 1) to document aspects, this would facilitate better communication among stakeholders during early-stage design and development activities. A formal definition of this schema is done using XML-Schema[12] (*aspect.xsd*). A pictorial XML-Schema is presented in fig. 6. In the remainder of this section, we present the top-level technical, service-level and domain-level aspects we have identified. An important point to note is that the vocabulary for these aspects would evolve and standardize over a period of time. Existing ontologies[13] could also be used to standardize the vocabulary.

Technical Aspects

Technical aspects addresses infrastructural and messaging concerns such as security, trust and transactions. These aspects must be conveyed through service policies to enable secure, trusted and reliable conversation between the service provider and the consumer. Fig. 5 presents the top-level technical aspects we have identified.



Name of Concern	The name of the concern addressed by the aspect	
Type of Aspect	Denotes the aspect type	
Related Aspects	Denotes related aspects for this aspect	
Context	Denotes the context for this aspect	
Rationale & Discussion	Provides a brief description of the aspect and its application	
Quantification	Denotes applicability of the aspect. It could be: - List of services in the services portfolio - Select services, interfaces, operations or interaction points (end-points) - Ownership Domains	
Vocabulary	Vocabulary defines a set of vocabulary items and their applicable values	
Vocabulary Items	Type	Applicable Values
Domain terms to describe the aspect	Type of vocabulary item	Acceptable values for the vocabulary item

Table. 1. Standard Scheme for Documenting and Cataloging Aspects

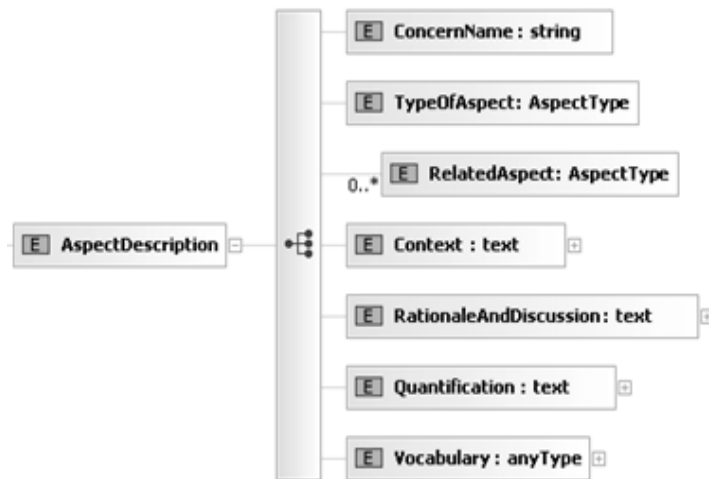


Fig. 6. A XML-Schema (pictorial representation) for Documenting Aspects

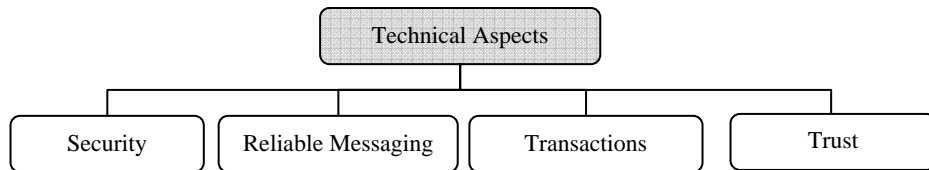


Fig. 5. Technical Aspects

- *Security*: Security deals with message level security between the service provider and consumer thereby guaranteeing a secure conversation. Security mainly

involves end-to-end message integrity, message confidentiality and authentication. As an example, we use the catalog schema to document the security aspect (Table 2).

- *Trust*: Trust is closely related to security. In the context of a secure conversation, trust determines the reliability and integrity of the service consumer from the perspective of the provider or vice-versa. In order to prove integrity, the consumer requests a token from a trusted third-party (e.g. Kerberos token from a Kerberos Token Distribution Center) and sends this to the provider to establish its identify.
- *Reliable Messaging*: Reliable messaging deals with end-to-end reliable and guaranteed delivery of messages between a service provider and a consumer.
- *Transactions*: Transactions addresses standard transaction mechanisms for short-duration ACID transactions as well as long-running business transactions.

Service-Level Aspects

Service-level aspects addresses service concerns such as quality of service, privacy of service consumers, pricing and availability. It also addresses how to promote the use of services in the services marketplace. Fig. 6 describes the top-level service-level aspects we have identified.

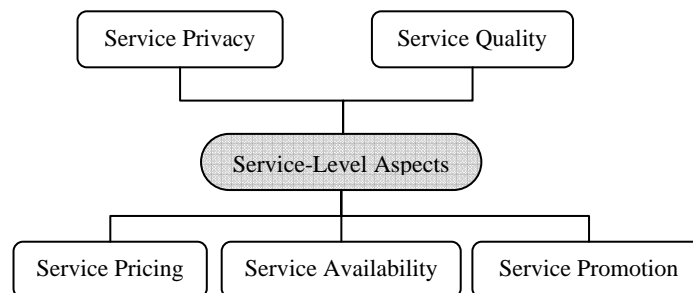
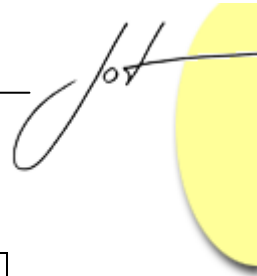


Fig. 6. Service-Level Aspects



Name of Concern	Secure Conversation	
Type of Aspect	Security	
Related Aspects	Trust	
Context	Security addresses secure conversation between the service provider and the consumer.	
Rationale & Discussion	Security addresses issues such as authentication, encryption and integrity of messages between the provider and the consumer.	
Quantification	Externally exposed services needing secure access	
Vocabulary		
Vocabulary Terms	Type	Applicable Values
Username	String	
PasswordType	String	Clear Text, Digest
PasswordValue	String	
IsBinarySecurityTokenRequired	Boolean	
BinaryEncodingType	String	Base64, Hex, UU
BinaryEncodingTokenType	String	Kerberos, X.509 (variants)
BinaryEncodingTokenValue	anyType	
isDigitalSignatureRequired	Boolean	
SignatureMethod	String	
HashMethod	String	SHA1, MD5
DigestValue	anyType	
IsEncryptionRequired	Boolean	
EncryptionMethod	String	DES, TripleDES, PGP

Table. 2. Technical Aspect - Security

- *Service Availability:* Service availability deals with spatial (location) and temporal availability concerns of a service. It determines the time of the day and the duration for which the service is available. It also determines the geographical reach (countries, regions, cities and states) of the service.
- *Service Pricing:* Service pricing deals with the price at which a service is offered. It also deals with price types, payment modes and the charging styles for the use of a service. As an example, we use the catalog schema to document the pricing aspect (Table 3).
- *Service Promotion:* Service promotion deals with promoting service consumption by customers and market segments by providing them with discounts and rewards.

- *Service Privacy*: Deals with protecting consumer information and ensuring confidentiality of the data exchanged between the service consumer and the provider. It also determines whether the consumer information would be shared with business partners in case of composite service offerings.
- *Service Quality*: Deals with guaranteeing consumers acceptable and agreed upon quality of service such as service availability, response time, performance and reliability.

Domain-Level Aspects

Domain-level aspects address business-level concerns such as compliance to legislative as well as industry regulations, adherence to business rules and following industry conventions (fig. 7).

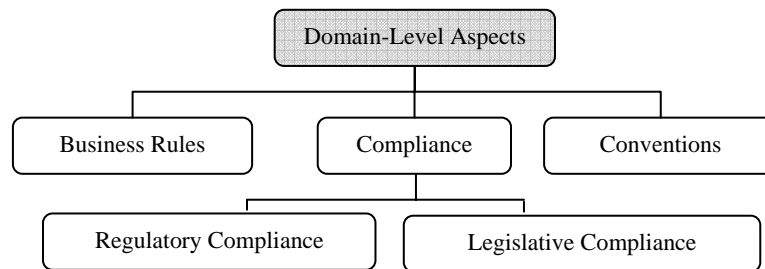
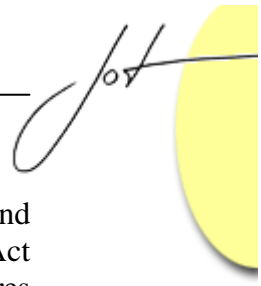


Fig. 7. Service-Level Aspects

- *Business Rules*: Business rules define constraints on the operations, or operational procedure of a business which influences the behavior of the business. Business rules could pertain to business calculations, business policies or restrictions.
- *Compliance*: Compliance addresses issues related to adhering to legislation (rule of the land) or with regulations set by industry regulatory authorities.
- *Conventions*: Conventions deal with generally accepted practices which have been followed in a particular business or industry over a period of time.

Domain-Level Aspects Vs Technical & Service-Level Aspects

Flexible Vocabulary: Technical aspects like security and service-level aspects such as pricing can have a generic vocabulary which could standardize over a period of time or through standard ontologies. The vocabulary of technical or service-level aspects remain similar across businesses or industries. However, in the case of domain-level aspects, though business rules and compliance are broad crosscutting concerns, their specific vocabulary vary. Consider the example of the Shipping industry – a business like FedEx® in the shipping business, has to comply with the ‘Bioterrorism Act 2002 - Prior Notice’ for food shipments. Meanwhile, the aviation industry would have to comply with the ‘Federal Aviation Act’. We note that though compliance remains an aspect across industries, the vocabulary for compliance is flexible. Due to its flexible vocabulary,



domain-level aspects have to be specifically defined for each business by regulatory and governance (domain) experts. We define vocabulary for ‘Compliance to Bioterrorism Act 2002 - Prior Notice’ (table 4) regulation in the Shipping industry. The regulation requires the consumer of the *Shipping Service :: Ship Item* Operation to intimate the FDA with a prior notice for food shipments and use the prior notice confirmation while using *ShipItem*.

Name of Concern	Compliance to Bioterrorism Act 2002 – Prior Notice Regulation	
Type of Aspect	Bioterrorism Act 2002 - Prior Notice Compliance	
Related Aspects	Regulatory Compliance	
Context	Compliance notice to the service consumer while shipping food exports.	
Rationale & Discussion	The aspect deals with compliance to the Bioterrorism Act 2002 – Prior Notice which requires the consumer to use a prior notice confirmation number to ship food exports.	
Quantification	ShippingService::ShipItem Operation	
Vocabulary		
Vocabulary Terms	Type	Applicable Values
IsPriorNoticeRequired	Boolean	
PriorNoticeConfirmationNumber	AlphaNumeric	

Table. 4. Domain-Level Aspect – Compliance to Bioterrorism Act 2002 (Prior Notice)

Restricted Quantification: Quantification deals with the ‘selection’ of services and other policy subjects in the services portfolio for applying a Policy i.e. it determines the policy scope. Unlike, technical and service-level aspects, the domain-level aspects have a limited quantification i.e. they do not have a broad impact on services in the services portfolio. Due to the nature of domain-level aspects they apply to specific services e.g. ‘Compliance to Bioterrorism Act 2002 - Prior Notice’ applied to *ShippingService::ShipItem*.

4 TRANSFORMING POLICY MODELS TO EXECUTABLE SPECIFICATIONS

Once the domain experts model the policies using our service policy metamodel, these policy models have to be converted to appropriate interoperable standards. The policies should also be incorporated into service descriptions. In the *ShippingService* example, the service capability model (in fig. 1) captured the underlying capability on offer. It was then converted to a standard WSDL 2.0 service description (in fig. 2). In the same manner, the policies described using our service policy models have to be transformed to appropriate industry accepted interoperable standards. Since there are multiple – and

sometimes – competing standards, we look at different standards available in each of layers of the generic policy model (table 5).

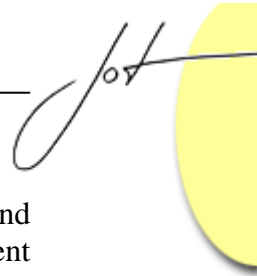
Technically, the policy models created using our service policy metamodel could be transformed to any of these specifications using MOF2 Model to Text Transformation Language (MTL)[14] standard mappings. But we have made certain choices about the standards we would use for transformations. These choices are based on two considerations - industry adoption and support for generic processing of policies.

Generic Policy Model Layer	Standards
Vocabulary Specification	XML Schema Web Ontology Language (OWL)[15] to support specification of domain ontologies
Constraint Specification	<i>Domain Dependent Specification:</i> Domain specific assertions using WS-SecurityPolicy, WS-Trust, WS-ReliableMessagingPolicy[16]. <i>Domain Independent Specification:</i> Domain independent assertions using WS-PolicyConstraints, XACML[17].
Policy Specification	WS-Policy[3] Web Service Policy Language (WSPL)[18]
Binding Specification	WS-PolicyAttachment[19]

Table. 5. Standards relevant to Generic Policy Model layers

Based on industry adoption we choose WS-Policy specification to specify policies. WS-Policy specification has a solid industry backing and is a mature W3C recommendation now. SOA vendors also support policies defined using WS-Policy in their middleware software. Having chosen WS-Policy, choosing WS-Policy Attachment was an obvious option for binding specification. For vocabulary specification and constraint specification: Domain-dependent constraint specification languages like WS-Security policy (security domain) and WS-ReliableMessagingPolicy (reliable messaging domain) have matured and evolved with WS-Policy. They provide standard semantics and constraints to specify security and reliable messaging capability. However, we choose a domain-independent constraint specification language – WS-PolicyConstraints. WS-PolicyConstraints help to specify domain-independent generic constraints using XACML-based functions. We choose the nascent WS-PolicyConstraints over the much adopted domain-dependent constraint languages for the following reason:

- Absence of existing assertion languages to specify domain-specific assertions for service-level aspects such as availability, pricing, promotions as well as domain-specific aspects.



-
- To provide flexibility in rich vocabulary specification for service-level and domain-level aspects across industries and businesses. Domain-dependent assertion languages have currently restricted vocabulary to improve interoperability.
 - Advantage of using a common generic policy handling logic for parsing policies in the SOA middleware instead of having multiple policy handlers.

We have developed a MTL transformation to transform the model developed using the service policy metamodel to preferred specifications (XML Schema, WS-PolicyConstraints, WS-Policy and WS-PolicyAttachment) (fig. 8). The Normal Form of WS-Policy is chosen for the transformation.

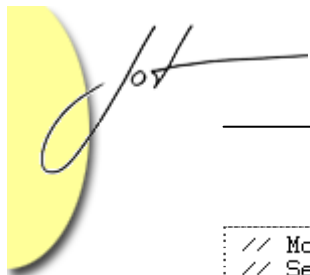
5 WORKED OUT EXAMPLE

In this section, we define a pricing policy which is later applied to the `ShippingService::ShipItem` service operation. The domain experts use the service policy metamodel to define the pricing policies and associate it with the `ShipItem` policy subject.

Pricing Policy: Every customer could be a subscription customer or a regular customer. Subscription customers pay a propotional price based on their use and have a credit period of one month. Regular customers pay an absolute price per service access.

Defining the Pricing Domain Vocabulary

The first step for the pricing expert (domain expert) is to define the domain vocabulary for the service pricing domain (fig. 8). The domain vocabulary is defined using visual models (defined in the service policy metamodel) and they are transformed to simple XML Schema using standard transformations (fig. 9).



```

// Model to Text Language Transformation
// Service Policy Metamodel to WS-Policy Normal Form
@text-explicit
[template public ModelToWSPolicySpec (policy : ServicePolicy) ]

<wsp:Policy
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:wspc="http://research.sun.com/ns/ws-policyconstraints"
  xmlns:wsu="http://docs.oasis-open.org/wssecurity-utility-1.0.xsd"
  wsu:Id = '[policy.name] '>

  <wsp:ExactlyOne>

    [For (alternative:PolicyAlternative | policy.alternatives) ]
    <wsp:All>
      [For (assertion:PolicyAssertion | alternative.assertions) ]
      [AssertionToWSPolicyConstraint(assertion) /]
    </wsp:All>
  </For>
</wsp:ExactlyOne>
</wsp:Policy>
[/template]

//Transform Assertion to WS-PolicyConstraint
@text-explicit
[template public AssertionToWSPolicyConstraint(assertion: PolicyAssertion)]
  <wsp:Apply FunctionId = "&function; [assertion.operator]" >
    <wspc:ResourceAttributeDesignator
      AttributeId = "[assertion.constrainedElement.name]"
      DataType = "[assertion.constrainedElement.type]" />
    <wspc:AttributeValue DataType = &type; [assertion.specification.type]>
      [assertion.specification.stringValue()]
    </wspc:AttributeValue>
  </wsp:Apply>
[/template]

```

Fig. 7. MTL Transformation (Service Policy Metamodel to Specifications)

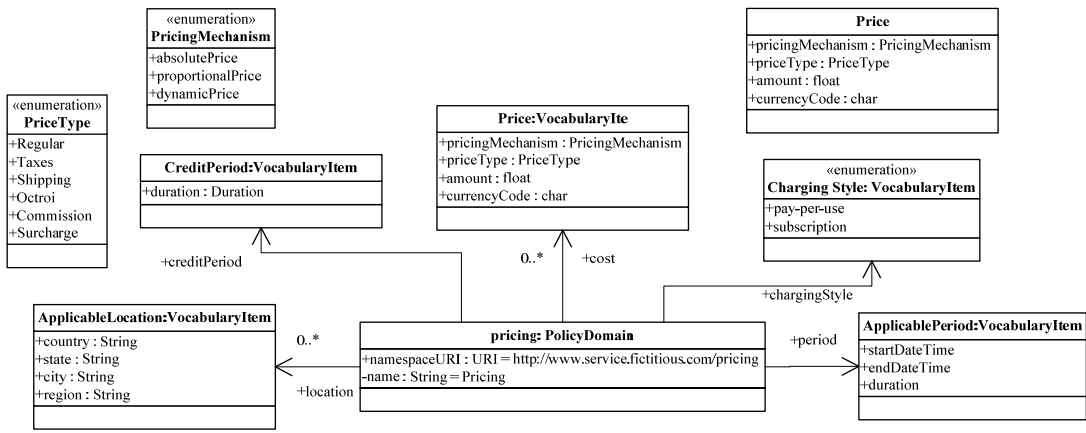
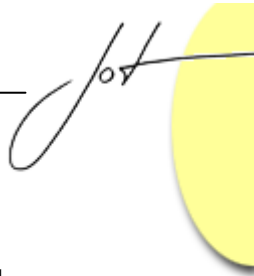


Fig. 8. Service Pricing Vocabulary Definition



Modeling the Pricing Policy and Attaching to Policy Subject

After modeling the pricing vocabulary, the pricing policy has to be modeled by the domain experts. Fig. 10, shows the pricing policy modeled using our services policy metamodel. By applying the MTL transformation, we get a corresponding WS-Policy representing the pricing policy, the policy is attached to the *ShippingService::ShipItem* service operation (policy subject) (fig. 11).

```
<?xml version="1.0" encoding="utf-8" ?>
- <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="ServicePricing">
- <xs:complexType>
- <xs:sequence>
- <xs:element minOccurs="0" maxOccurs="unbounded" name="Price">
- <xs:complexType>
- <xs:sequence>
+ <xs:element name="PricingMechanism">
+ <xs:element name="PriceType">
+ <xs:element name="Amount">
+ <xs:element name="CurrencyCode">
</xs:sequence>
</xs:complexType>
</xs:element>
+ <xs:element minOccurs="0" maxOccurs="unbounded" name="ApplicableLocation">
+ <xs:element name="ApplicablePeriod">
- <xs:element name="ChargingStyle">
- <xs:simpleType>
- <xs:restriction base="xs:string">
<xs:enumeration value="pay-per-use" />
<xs:enumeration value="subscription" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CreditPeriod" type="xs:duration" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Fig. 9. Service Pricing Vocabulary as XML Schema

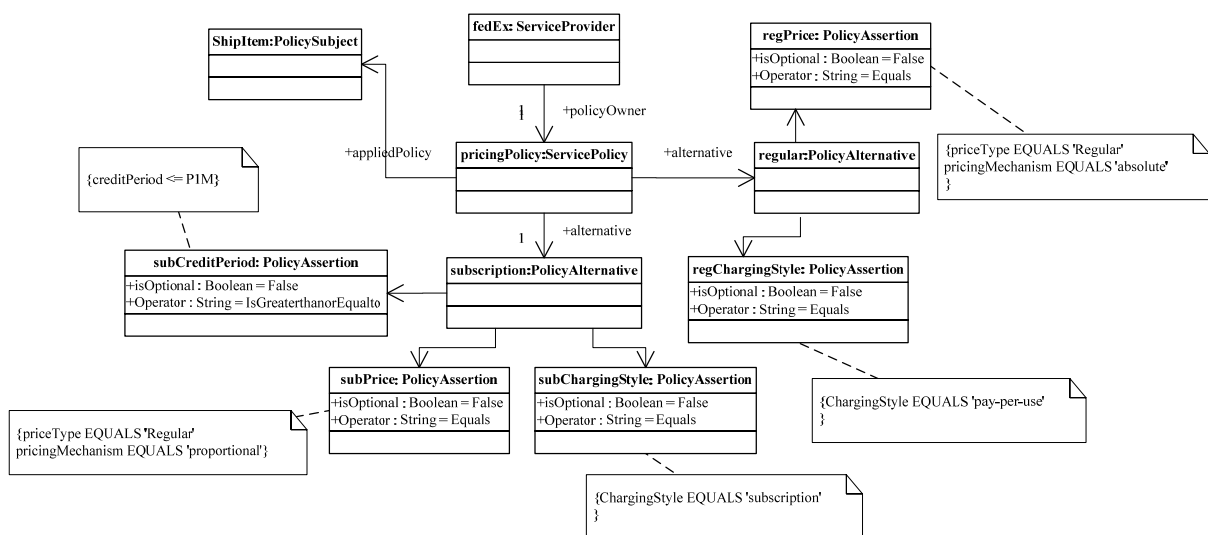


Fig. 10. Service Pricing Policy Model

```

<wsp:Policy wsu:Id = "ServicePricingPolicy">
  <wsp:ExactlyOne>
    <!-- Policy Alternative: Subscription Customers -->
    <wsp:All>
      <Apply FunctionId="&wspc:function:is-less-than-or-equal">
        <AttributeValue DataType="&xsd:duration">P1M</AttributeValue>
        <ResourceAttributeDesignator AttributeId="creditPeriod" DataType="&xsd:duration"/>
      </Apply>
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">Regular</AttributeValue>
        <ResourceAttributeDesignator AttributeId="price/priceType" DataType="&xsd:string"/>
      </Apply>
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">proportional</AttributeValue>
        <ResourceAttributeDesignator AttributeId="price/pricingMechanism" DataType="&xsd:string"/>
      </Apply>
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">subscription</AttributeValue>
        <ResourceAttributeDesignator AttributeId="chargingStyle" DataType="&xsd:string"/>
      </Apply>
    </wsp:All>
    <wsp:All>
      <!-- PolicyAlternative: Regular Customers -->
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">Regular</AttributeValue>
        <ResourceAttributeDesignator AttributeId="price/priceType" DataType="&xsd:string"/>
      </Apply>
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">absolute</AttributeValue>
        <ResourceAttributeDesignator AttributeId="price/pricingMechanism" DataType="&xsd:string"/>
      </Apply>
      <Apply FunctionId="&wspc:function:equals">
        <AttributeValue DataType="&xsd:string">pay-per-use</AttributeValue>
        <ResourceAttributeDesignator AttributeId="chargingStyle" DataType="&xsd:string"/>
      </Apply>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>

```

Policy Attachment

```

<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:URI>http://...#interface/operation(ShippingServiceInterface/ShipItem)</wsa:Address>
  </wsp:AppliesTo>
  <wsp:PolicyReference URI="http://www.fictitious.com/policies#ServicePricingPolicy" />
</wsp:PolicyAttachment>

```

Fig. 11. Pricing Policy expressed using WS-Policy, WS-PolicyConstraints & WS-PolicyAttachment

6 POLICY ENFORCEMENT AT THE SOA MIDDLEWARE

Once the policies are modeled and associated to the policy subjects, the service descriptions are enhanced with policy information. Now these policies have to be enforced in the SOA middleware (we assume that the services are consumed through SOAP[20]). The most important criterion for policy enforcement is that it has to be *unintrusive*. We use an active SOAP intermediary – the Policy Enforcement Point intermediary *PEP Intermediary* (fig. 12). The PEP intermediary works on the SOAP headers associated with service policies. We use Apache Axis 2.0[21](hereon Axis2) SOAP engine as our PEP SOAP intermediary. We take advantage of the extensible SOAP processing model of Axis2.

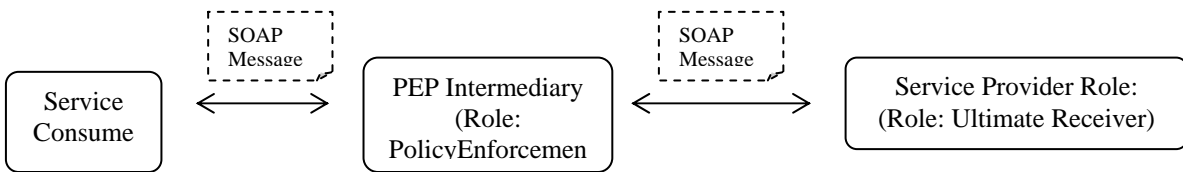
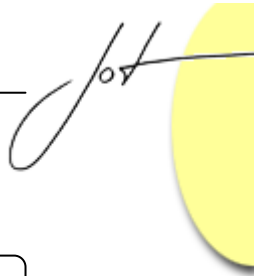


Fig. 12. Unintrusive Policy Enforcement using PEP Intermediary

We have a *Generic Handler* (an XACML policy processor) to handle all service policies – an advantage of using the domain-independent WS-PolicyConstraints. However, if we need application specific programming logic to handle special policy enforcement for certain policy domains, we could optionally choose to have an exclusive *Policy Enforcement Handler (PE Handler)*. The generic handler and the optional PE handlers are part of a user-defined *Policy Enforcement Phase (PE Phase)*. As soon as a new instance of a policy domain is added in the policy model, a corresponding flavoring handler is optionally generated and automatically added to the end of the PE phase. Apart from having the PE handlers and the generic handler, we also have a Consumer Profiling Handler (CPH) which is the first handler that gets invoked in the PE phase. The CPH deals with identifying and profiling the consumer. The consumer profile information is shared with the other handlers using the *MessageContext*. The SOAP headers representing different aspects such as security, pricing etc. have the role <http://fictitious.com/role/policyEnforcement> and the PE intermediary which plays the ‘policy enforcement’ role must understand and process these headers. Once the policy enforcement is done, the SOAP messages are routed to the ultimate receiver (or the service provider).

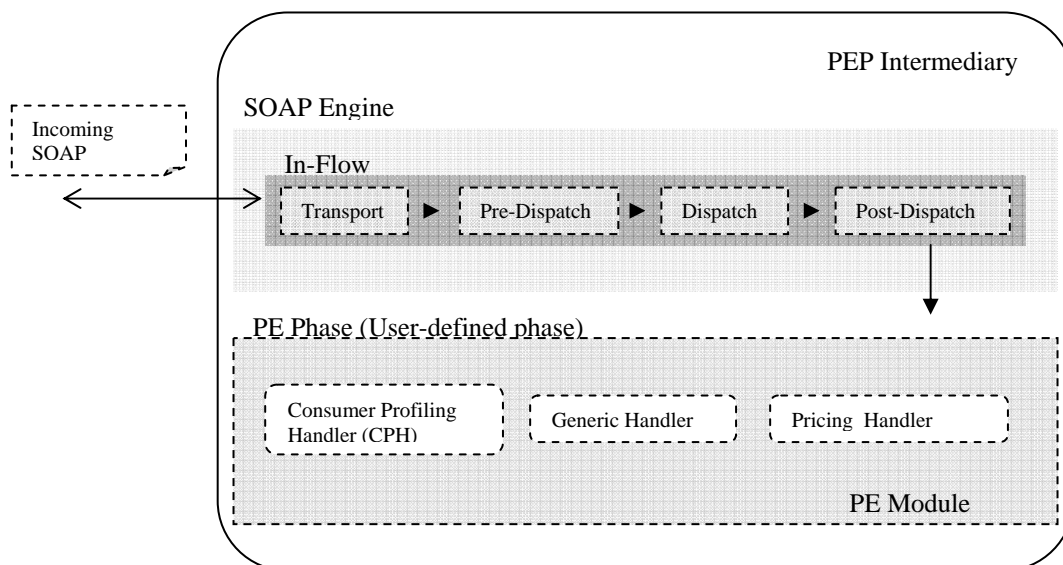


Fig. 13. Inside the PEP Intermediary

Fig. 14 presents a sample SOAP request for the *ShippingService::ShipItem* (the SOAP body is not presented for brevity). The header elements *ConsumerProfile* and *Service Pricing* would be intercepted by the PEP intermediary based on the role. These are then handled by CPH and Pricing Handler respectively.

```

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    <consumer:ConsumerProfile
      xmlns:consumer="http://fictitious.com/consumerprofile"
      xmlns:common="http://fictitious.com/common"
      soap:role="http://fictitious.com/role/policyEnforcement"
      soap:mustUnderstand="true">
      <consumer:reference scheme="UUID">00300571-cecb-1dec-978d-559058888227</consumer:reference>
      <consumer:FormattedName>Take5 Inc</consumer:FormattedName>
      <common:accessTime>2008-03-29T13:20:00.000</common:accessTime>
      <common:Location>
        <common:Country>US</common:Country>
      </common:Location>
    </consumer:ConsumerProfile>

    <pricing:ServicePricing
      xmlns:pricing="http://fictitious.com/servicepricing"
      soap:role="http://fictitious.com/role/policyEnforcement"
      soap:mustUnderstand="true">
      <pricing:chargingStyle>subscription</pricing:chargingStyle>
    </pricing:ServicePricing>
  </soap:Header>
  <soap:Body> .. </soap:Body>
</soap:Envelope>

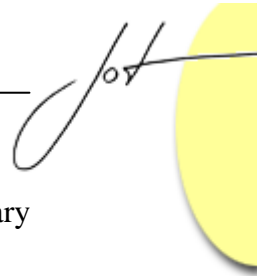
```

Fig. 14. Sample SOAP Request for *ShippingService::ShipItem*

7 RELATED WORK

Model-driven approaches to developing web services[23,24,22] are increasingly getting popular. OMG realized the need to standardize model-driven services development – the result – RFP (request for proposal) UML Profile and Metamodel for Services (UPMS)[25], hereon RFP UPMS. However, the RFP UPMS does not address Service Policies, the focus is on Service Modeling – capability and contract modeling. The OASIS SOA Reference Model (SOA-RM)[1] and the WS-Arch[26] (Web Services Architecture) describe service policies in detail. Our approach complies to the SOA-RM. In our approach, we consider all aspects of service policy modeling by addressing the 4-layers of the generic policy model.

A close related work – Ortiz et al.'s [27] work on modeling extra-functional properties deals with modeling services based on the Service-Component Architecture (SCA) and defining extra-functional properties. They have developed a UML Profile for SCA and to model extra-functional properties[28]. However the focus of their approach is not on independent policy development – by describing alternatives and constraints – instead the focus is on defining extra-functional properties at the modeling level and representing it using WS-Policy. Policy enforcement implementations are based on aspect-oriented techniques[29]. Moreover, the aspects dealt (e.g. logging) are more technical in nature, in comparison our approach addresses technical, service-level and



domain-level aspects. Also Ortiz et al.'s approach does not address vocabulary specification for policy domains and constraint specification.

With respect to vocabulary specification, O'Sullivan has done extensive work on non-functional properties in service descriptions; he has also produced concrete XML syntax of service properties[30], which could be reused as vocabularies. Also ontologies such as QoSOnt[31] (an ontology for QoS) could be reused to describe policy vocabulary.

With respect to policy enforcement implementations, we use a SOAP intermediary to handle policy enforcements. However, there are variety of approaches[33,32] (including Ortiz et al's) using aspect-oriented programming techniques to handle crosscutting aspects like service management and adaptability. Our approach could complement those approaches and provide means to identify aspects and aid in the entire life-cycle of service policy development. Later, we could enhance our approach to support AOP-based quantification.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we addressed broad-based independent service policy development using a model-driven approach. We deal with all stages of service policy development related to early-stage services development based on the generic policy framework. We also addressed different levels of policy aspects – technical, service-level and domain-level aspects. We demonstrated our approach using the *ShippingService* and a Service Pricing Policy example. As part of the future work, we would investigate the following:

Support for Modeling Semantic Policies: The policy specification is more syntactic in nature. In order to support policy matching (through policy intersection) between the consumer and the provider, the role of semantics – the underlying meaning of the vocabulary items, constraints and alternatives are important. There are proposals to add semantics to service policies[34] and express policies using Web Ontology Language (OWL)[35], we would investigate ways to support such semantic service policy descriptions during policy modeling.

Aspect-oriented Policy Quantification & Enforcement: We currently achieve abstraction and modularity in policy enforcement using the PEP intermediary. However we would investigate aspect-oriented techniques to support quantification and enforcement of service policies

Defining Dependencies between Policy domains: We would investigate modeling dependencies between policy domains, e.g. the relation between service pricing and promotions.

REFERENCES

- [1] (OASIS), *Reference Model TC: Oasis reference model for service oriented architectures (working draft 10)*, Technical Report, Organization for the Advancement of Structured Information Standards (OASIS), 2005.
- [2] J.J. Moreau et al., *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, 2006.
- [3] Ü. Yalçınalp et al., *Web Services Policy 1.5 - Framework*, W3C, 2007.
- [4] D.S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley, 2003.
- [5] H. Jegadeesan and S. Balasubramaniam, "A MOF2-based Services Metamodel," *Journal of Object Technology*, Vol 7, No. 8, November-December 2008. http://www.jot.fm/issues/issue_2008_11/article1/
- [6] O.M.G. (OMG), *Meta Object Facility (MOF) Specification 2.0 Core*, 2006.
- [7] A. Anderson, "Web services policies," *IEEE Security & Privacy Magazine*, vol. 4, 2006, pp. 84-87.
- [8] L. Fuentes-Fernández and A. Vallecillo-Moreno, "An Introduction to UML Profiles," *UPGRADE, The European Journal for the Informatics Professional*, vol. 5, 2004, pp. 5-13.
- [9] O.M.G. (OMG), "UML 2.0 Infrastructure Specification," *OMG formal document*, pp. 03-09.
- [10] J. Kohl and C. Neuman, *The Kerberos Network Authentication Service (V5)*, RFC 1510, September 1993, 1993.
- [11] R.E. Filman, *Aspect-oriented Software Development*, Addison-Wesley, 2005.
- [12] D.C. Fallside, "XML Schema Part 0: Primer," *W3C Candidate Recommendation CR-xmlschema-0-20001024*, World Wide Web Consortium (W3C), Oct, 2000.
- [13] N.F. Noy and D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Knowledge Systems Laboratory*, March, 2001.
- [14] O.M.G. (OMG), *MOF Model to Text Transformation Language v1.0*, 2008; <http://www.omg.org/spec/MOFM2T/1.0/PDF>.
- [15] D.L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview, W3C Recommendation," *World Wide Web Consortium*, 2004.
- [16] Ü. Yalçınalp et al., "Web Services Policy 1.5-Guidelines for Policy Assertion Authors," *W3C Working Group Note 12*, Nov. 2007.



-
- [17] A.H. Anderson, "Domain-Independent, Composable Web Services Policy Assertions," *Proc. of the 7th IEEE Int'l Workshop on Policies for Distributed Systems and Networks, IEEE CS*, 2006, pp. 149–152.
- [18] A.H. Anderson, "An introduction to the Web Services Policy Language (WSPL)," *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 2004, pp. 189-192.
- [19] A.S. Vedamuthu et al., "Web Services Policy 1.5-Attachment," *W3C Working Draft 31*, Jul. 2006.
- [20] M. Gudgin et al., "SOAP Version 1.2," *W3C Working Draft*, 2002.
- [21] S. Perera et al., "Axis2, Middleware for Next Generation Web Services," *Proc. ICWS 2006*, 2006, pp. 833-840.
- [22] K. Bai'na et al., "Model-Driven Web Service Development," *Advanced Information Systems Engineering: 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004: Proceedings*, 2004.
- [23] C. Emig et al., "Model-Driven Development of SOA Services," Technical report, Forschungsbericht, 2007.
- [24] M. Brambilla et al., "Model-driven Development of Web Services and Hypertext Applications," *SCI2003, Orlando, Florida, July, 2003*.
- [25] O.M.G. (OMG), *UML Profile and Metamodel for Services (UPMS) RFP*, 2006; <http://www.omg.org/cgi-bin/apps/doc?soa/06-09-09.pdf>.
- [26] W. (W3C), *Web Services Architecture*, 2004; <http://www.w3.org/TR/ws-arch/>.
- [27] G. Ortiz, J. Hernandez, and P.J. Clemente, "How to Deal with Non-functional Properties in Web Service Development," *Web Engineering: 5th International Conference, ICWE 2005, Sydney, Australia, July 27-29, 2005: Proceedings*, 2005.
- [28] G. Ortiz and J. Hernández, "Toward UML Profiles for Web Services and their Extra-Functional Properties," *Proc. Int. Conf. on Web Services, Chicago, EEUU, September, 2006*.
- [29] G. Ortiz et al., "How to Model Aspect-Oriented Web Services," *Workshop on Model-driven Web Engineering*.
- [30] J. O'Sullivan, D. Edmond, and A.H.M. ter Hofstede, *Formal description of non-functional service properties*, Technical report, Queensland University of Technology, Brisbane, 2005. Available from <http://www.service-description.com>, .
- [31] G. Dobson, R. Lock, and I. Sommerville, "QoSOnt: a QoS Ontology for Service-Centric Systems," *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, 2005, pp. 80-87.

- [32] G. Ortiz and F. Leymann, "Combining WS-Policy and Aspect-Oriented Programming," *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, Washington, DC, USA: IEEE Computer Society, 2006, p. 143.
- [33] F. Baligand and V. Monfort, "A concrete solution for web services adaptability using policies and aspects," *Proceedings of the 2nd international conference on Service oriented computing*, 2004, pp. 134-142.
- [34] K. Verma and A. Sheth, "Semantically Annotating a Web Service," *IEEE Internet Computing*, vol. 11, 2007, pp. 83-85.
- [35] B. Parsia, V. Kolovski, and J. Hendler, "Expressing WS Policies in OWL," *Policy Management for the Web Wkshp*, May, 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