Architectures Orientées Services

Les Web Services

SOAP
WSDL
UDDI

Small Object Application Protocol
Web Service Description Language
Universal Description Discovery & Integration…

De quoi s'agit il ?

- Quoi :
  - *Les Web Services*
- Pour quoi faire ?
  - Mutualiser des composants applicatifs
- Quel Intérêt ?
  - Le marché des applications en ligne
- Comment ça marche ?
  - WSDL, SOAP, UDDI
- Exemples

Autrefois, comme encore bien des fois…

```
PROGRAM PIVOT_DE_GAUSS
C Declaration des variables C
      REAL A(10, 10), B(10, 10), X(10, 10)
      INTEGER m, n, o, p
      INTEGER C LOGICAL err
      REAL det
C Declarations des fonctions C
      LOGICAL MULTIP
      LOGICAL GAUSS
      LOGICAL INV
      REAL DETERM
C Corps du programme principal C
      PRINT*, 'Programme de multiplication matricielle'
10    PRINT*, ' ' PRINT*, 'Voulez-vous :'
      PRINT*, '(1)- Entrer A'
      PRINT*, '(8)- Calculer inv(A)'
      PRINT*, '(9)- Calculer det(A)'
      PRINT*, '(0)- Sortir' READ*, C
      IF (C .EQ. 1) THEN
         PRINT*, 'Dimension de A' READ*, m
         PRINT*, ' -> A (carree) '
         CALL LIRE(A, m, m)
      ELSEIF (C .EQ. 2) THEN
         PRINT*, 'Dimension de B' READ*, n, o
         PRINT*, ' -> B '
         CALL LIRE(B, n, o)
      ELSEIF (C .EQ. 3) THEN
         PRINT*, ' A ='
         CALL AFFICH(A, m, m)
      ELSEIF (C .EQ. 4) THEN
         PRINT*, ' B =' CALL AFFICH(B, n, o)
      ELSEIF (C .EQ. 5) THEN
         PRINT*, ' X ='
         CALL AFFICH(X, m, p)
      ELSEIF (C .EQ. 6)
         THEN PRINT*, ' A*B -> X'
          err = MULTIP(A, m, m, B, n, o, X)
(…)
```

```
SUBROUTINE AFFICH (A, m, n)
REAL A(10,10)
INTEGER m, n, i, j
DO 100 i=1, m
     PRINT*, (A(i, j), j=1, n)
100 CONTINUE
RETURN
END
```

…Un programme exécutait des sous-programmes auxquels il avait été dûment lié,

après recopie

dans un exécutable monolithique…

Web Services ?

- **Wikipedia** :The W3C defines a **Web service** as *a software system designed to support interoperable machine-to-machine interaction over a network*.

- Les Web services assurent des dialogues entre applications sur Internet, avec une
  - Indépendance des plates-formes d'exécution
  - Mutualisation de ressources applicatives distantes
  - Coopération entre applications indépendante des langages
- Une façon de réaliser une architecture distribuée sans contrainte « objet »
  - Différente de CORBA (common object broker architecture)
  - Plus simple, plus ouverte,
  - Beaucoup Moins performante
- Trois + Deux protocoles d'organisation pour
  - définir un service : WSDL , Web Service Description Language
  - exposer/découvrir, décrire un service : UDDI
  - appeler un service : SOAP, Small Object Aplication Protocol

  - sécuriser un service : WS-Security :
    - modalités d'usage de XML-Encription et XML-Signature dans SOAP.
  - Fiabiliser WS-ReliableExchange:
    - un protocole de fiabilisation d'échange entre Web services.
- D'autres protocoles pour composer des services : BPML, WSFL, BPEL4WS...

Web Services : Intérêt économique

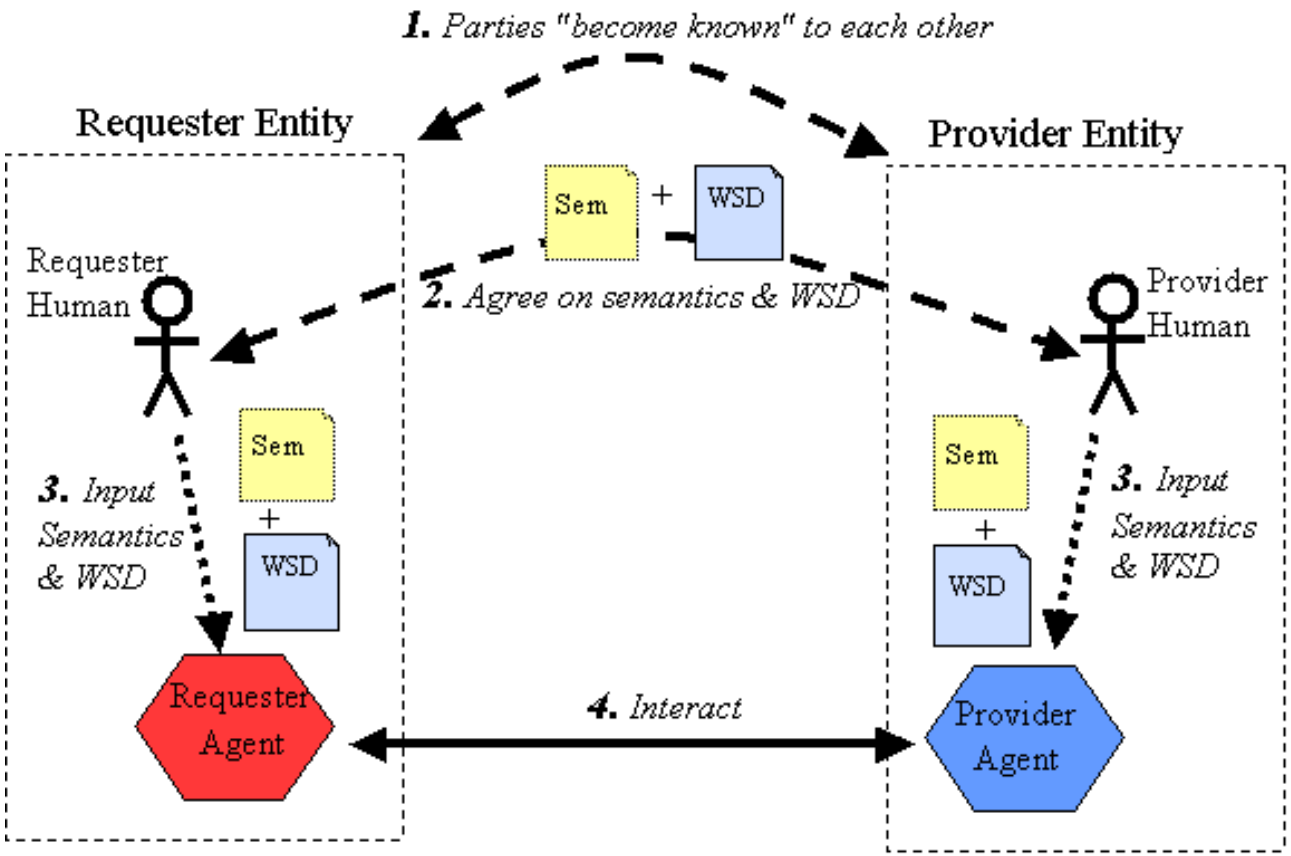
Achat de licences ?

ou

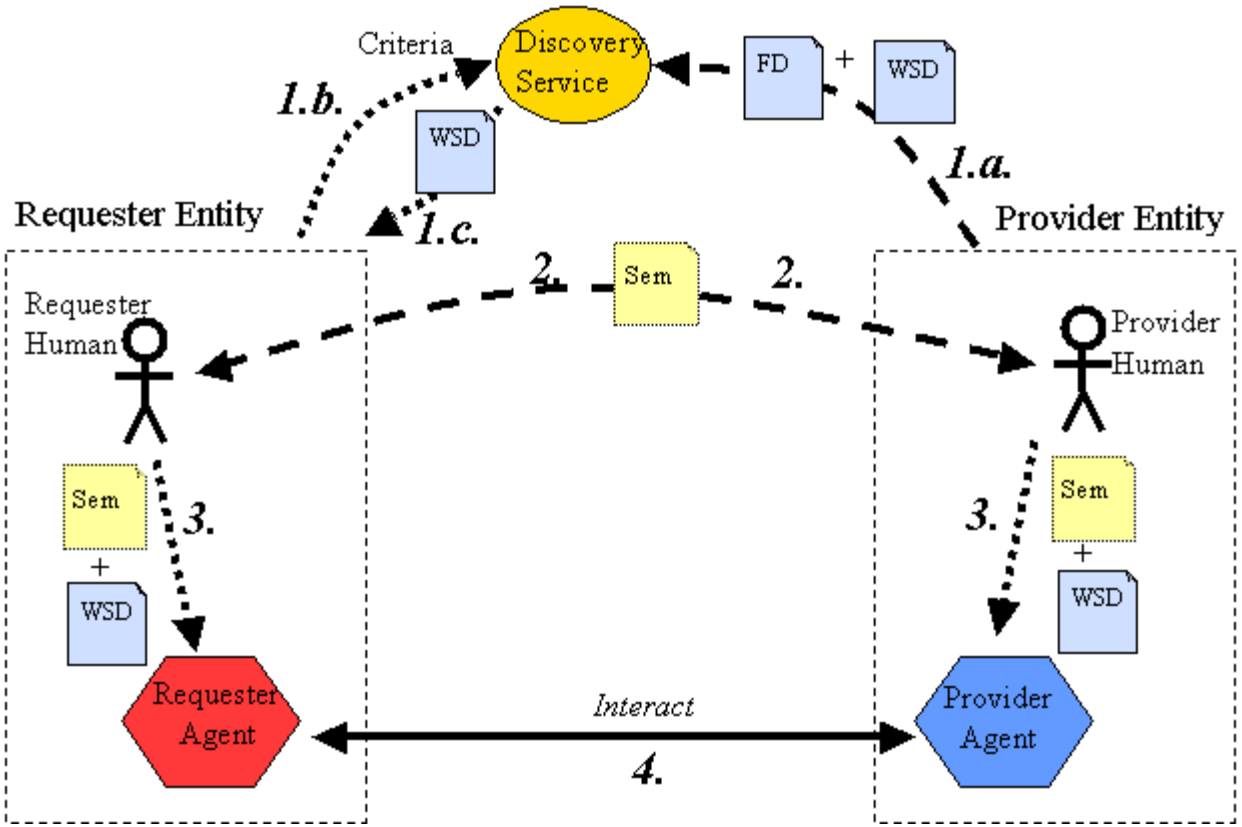
Facturation au service rendu ?

- *Nostalgie des mainframes, et des terminaux*
  - *Exploitation centralisée,*
  - *TSO, MVS/XA, systèmes transactionnels…*
- Retour vers la centralisation des applications au sein des entreprises
  - Simplification des déploiements d'applications,
  - Réduction des coûts de licence,
  - Maîtrise centralisée des activités et des données ?
  - Rente par péage à l'usage lors de chaque appel.

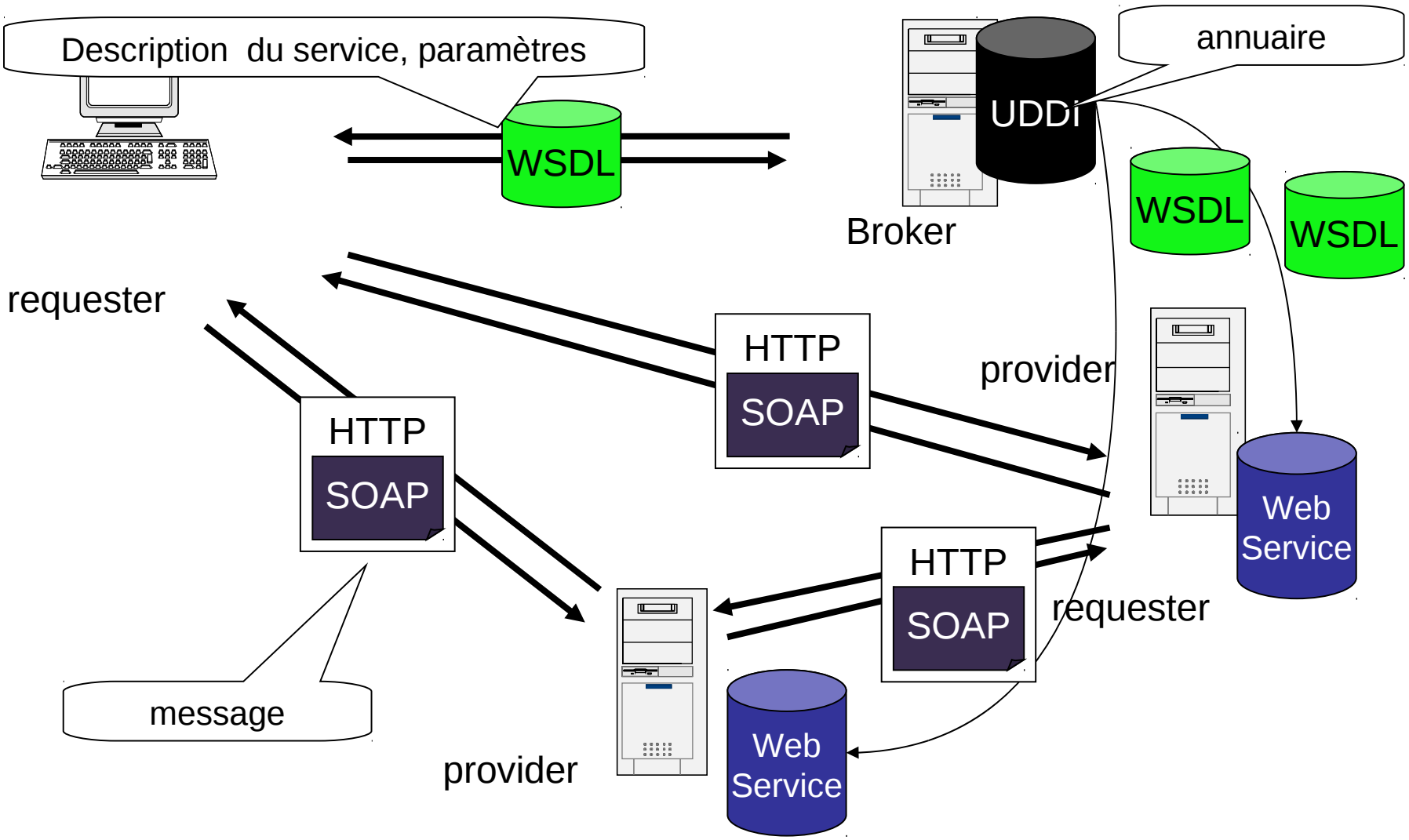
Cas d'usage : conventions et échanges



Cas d'usage : découverte



Architecture de Web Services

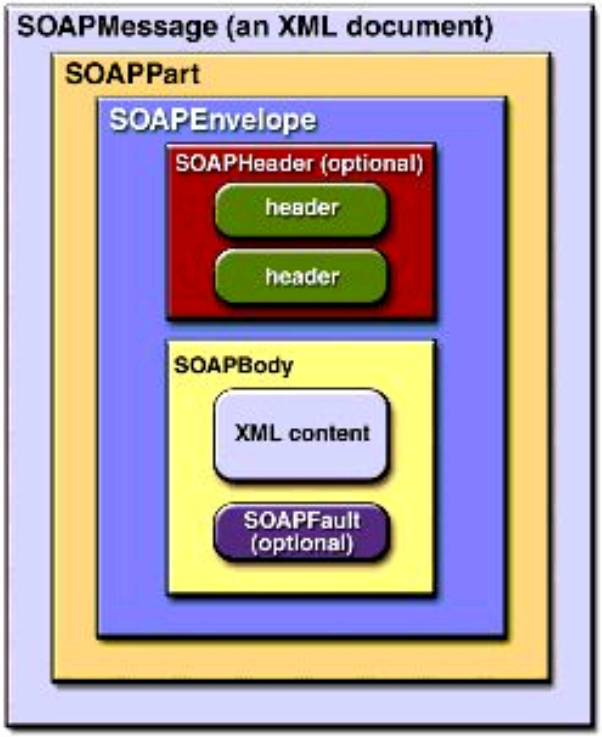


Principes de réalisation

- Un mécanisme de transport de demande de service entre client et serveur, connectés au réseau
  - **trois couches horizontales:**
    - Couche de transport : structure des messages entre applications pour la découverte et les échanges.
    - Couche de sémantique : données échangées selon des modèles métiers
    - Couche de gestion des processus : gestion des processus métiers répartis sur différentes applications distantes sur le réseau
  - **trois fonctions verticales**
    - Service d'Annuaire (découverte)
    - Service de sécurité (authentification)
    - Service de transaction (orchestration)

SOAP ? *Simple Object Access Protocol*

- Des messages que s'échangent de nouvelles sortes d'applications informatiques
  - SOAP est la partie de la couche communication des Web Services
  - SOAP peut être utilisé :
    - pour l'appel de méthodes SOAP RPC (Remote Process Call)
    - pour l'échange de messages (SOAP Messaging)



SOAP…

- Comment ça marche ? :
  - Les requêtes et réponses transitent via le protocole HTTP.
- Requête:
  - GET – retourne la ressource identifiée par la requête.
  - HEAD – retourne l'entête identifiée par la requête.
  - POST – envoie des valeurs sans limite de taille vers le serveur.
  - PUT – stocke une ressource pour la requête.
  - DELETE – supprime la ressource identifiée par la requête.
  - OPTIONS – retourne les méthodes HTTP supportées par le serveur.
  - TRACE – retourne le contenu des entêtes utilisées avec l'option TRACE.

- HTTP 1.0 inclut seulement les méthodes GET, HEAD et POST. Bien que les serveurs J2EE requièrent seulement la version HTTP 1.0, beaucoup de serveurs supportent la version HTTP 1.1.

Réponses

- Une réponse HTTP contient :
  - le code du résultat
  - l'en-tête
  - le corps
- Voici quelques erreurs identifiées par un numéro précis sur le protocole HTTP :

  404 – indique que la ressource demandée est indisponible.
  401 – indique que la requête requière une authentification HTTP.
  500 – indique une erreur interne au serveur HTTP.
  503 – indique que le serveur est surchargé.

- Exemple :

```
HTTP/1.0 200 OK
  Content-Type: text/xml; charset="utf-8"
  Content-Length: 359
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetCurrentTemperatureResponse xmlns:m="WeatherStation">
            <m:temperature>26.6</m:temperature>
        </m:GetCurrentTemperatureResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Pour la fiabilité des données transmises, IBM a mis en place un standard : HTTPR (*HTTP Reliable, HTTP Fiable*). Ce standard doit permettre d'assurer l'envoi ou la réception fiable de messages SOAP.

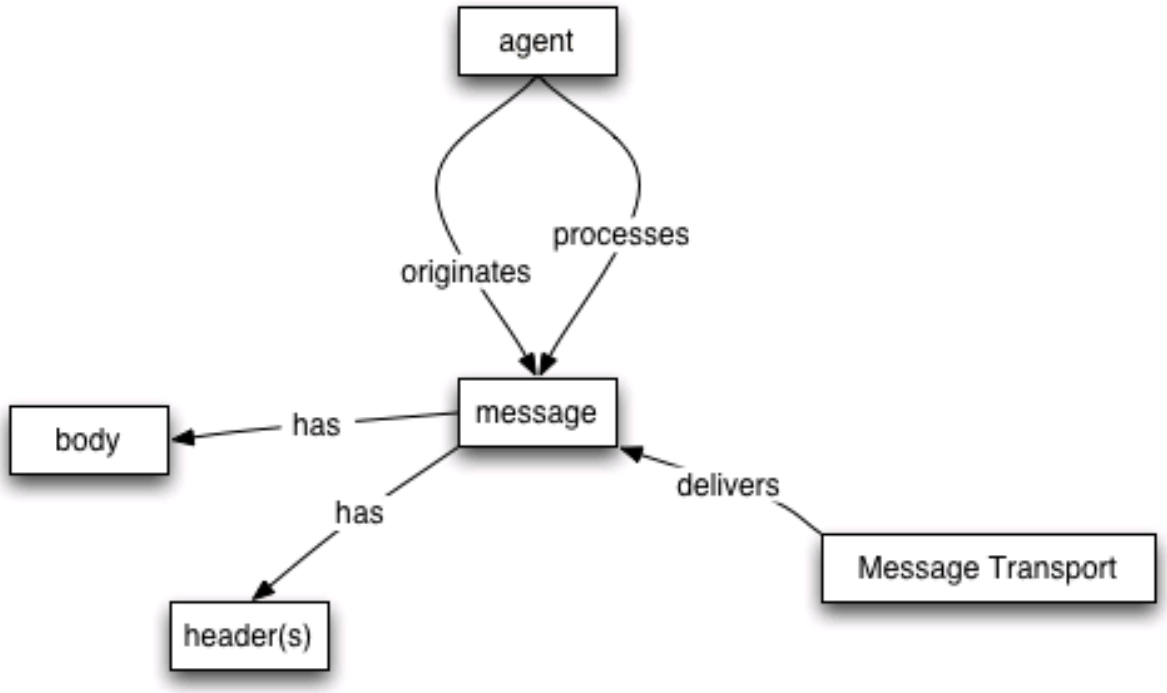
Exemple SOAP

```
POST /LocalWeather HTTP/1.0
Host : WWW.minstrm.com
Content-Type: text/xml; charset="utf-8" Content-Length: 328
SOAPAction: "WeatherStation"

<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

   <SOAP-ENV:Body>
    <m:GetCurrentTemperature xmlns:m="WeatherStation">
        <m:scale>Celsius</m:scale>
    </m:GetCurrentTemperature>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

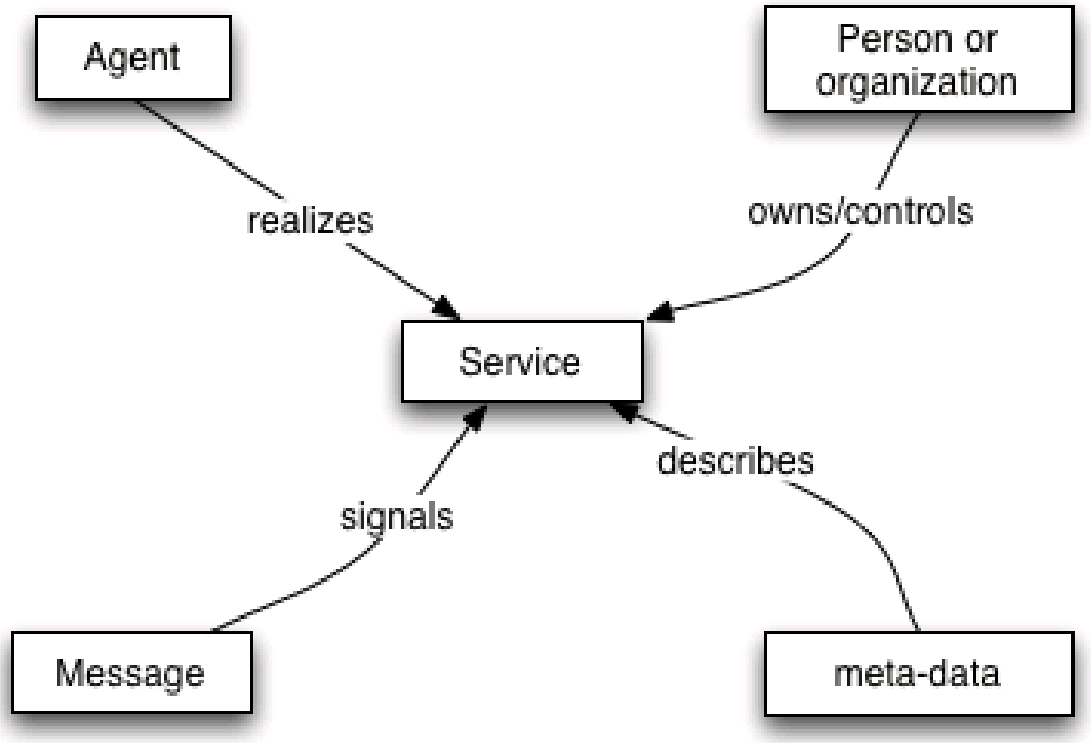
Architectures Orientées Services

Méta-modèles

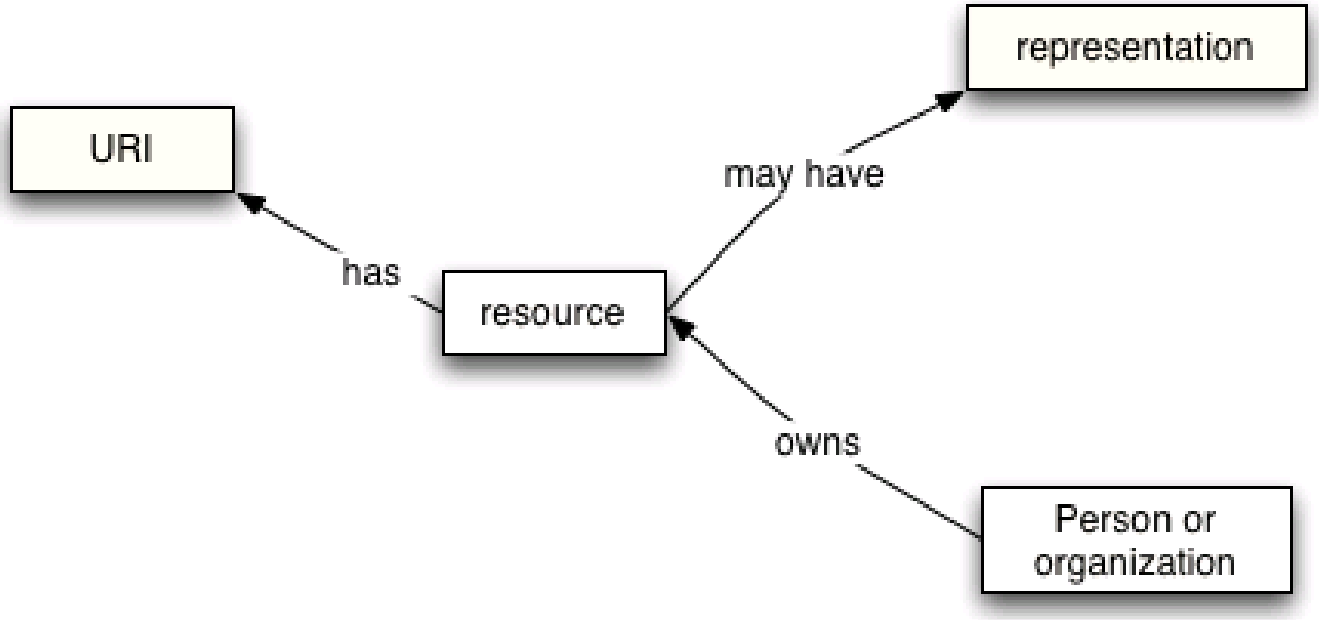
Message Oriented Model



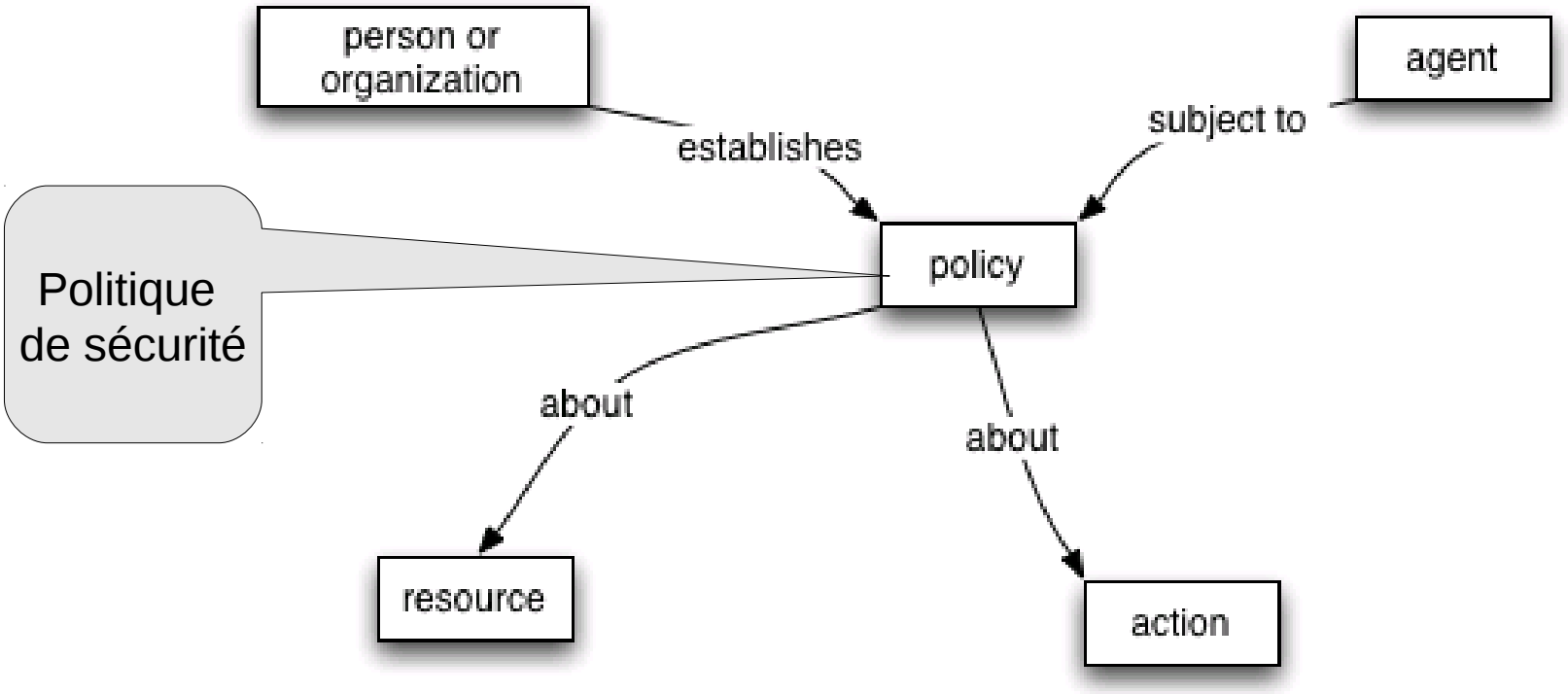
Service Oriented Model



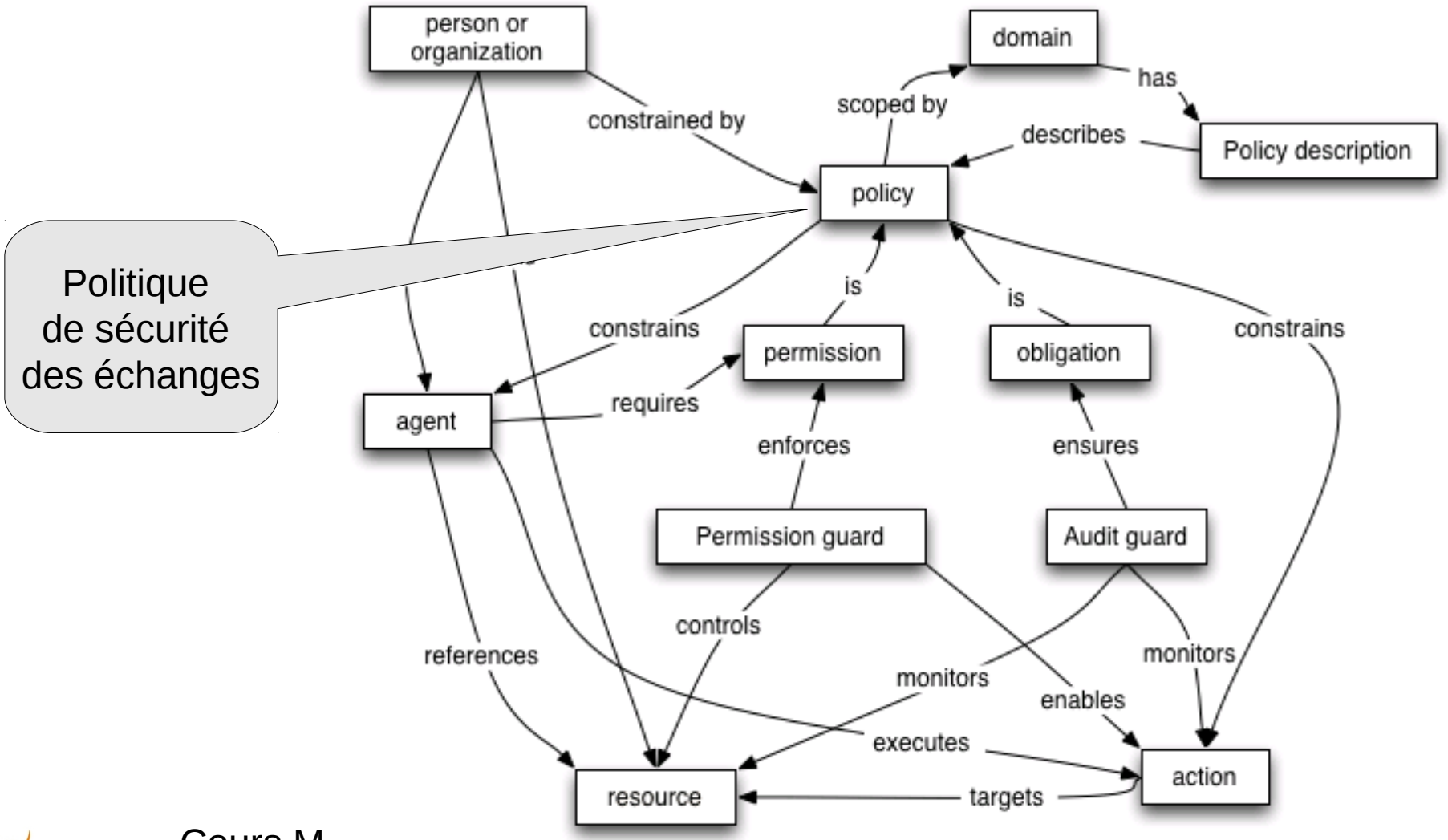
Resource Oriented Model



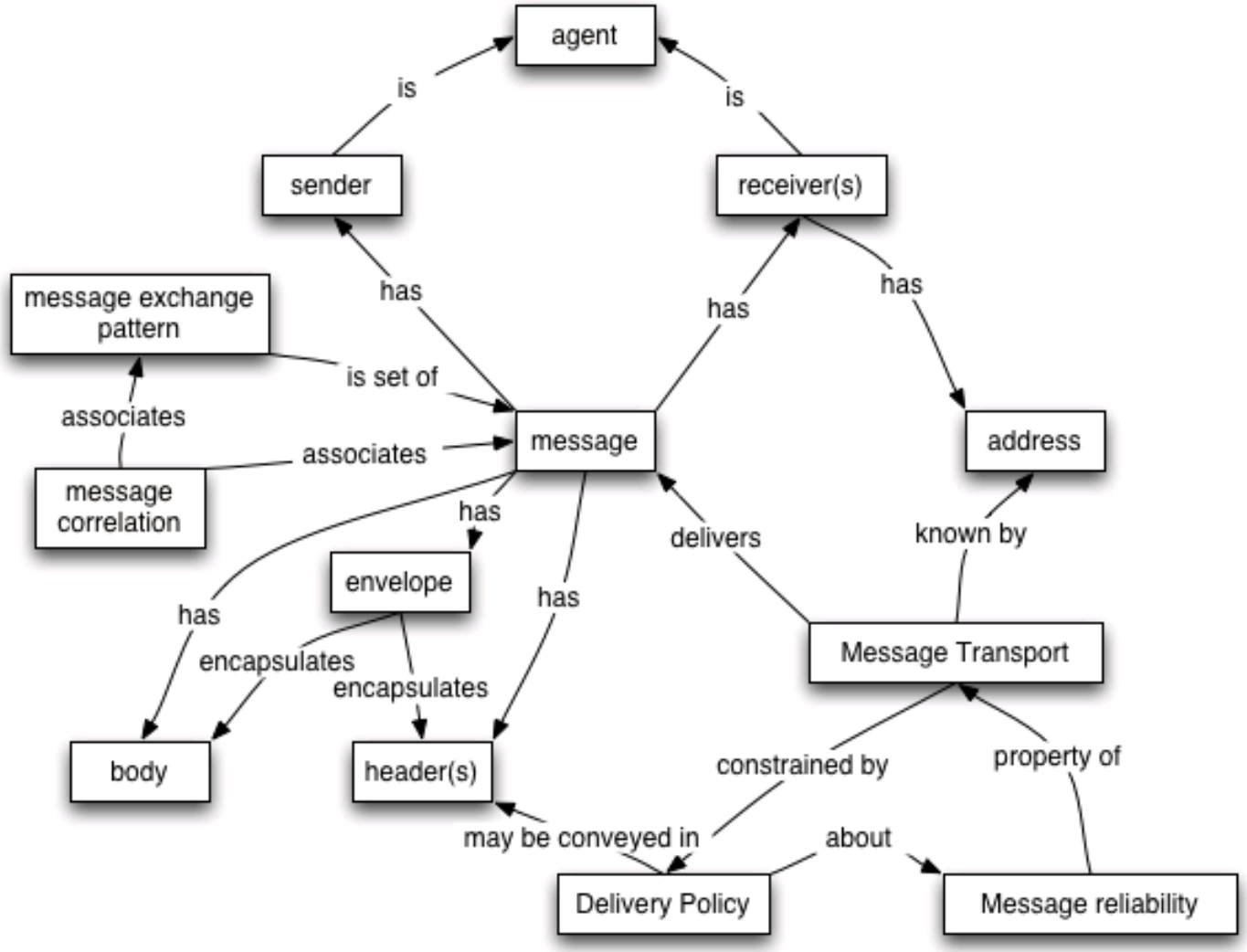
Policy Model



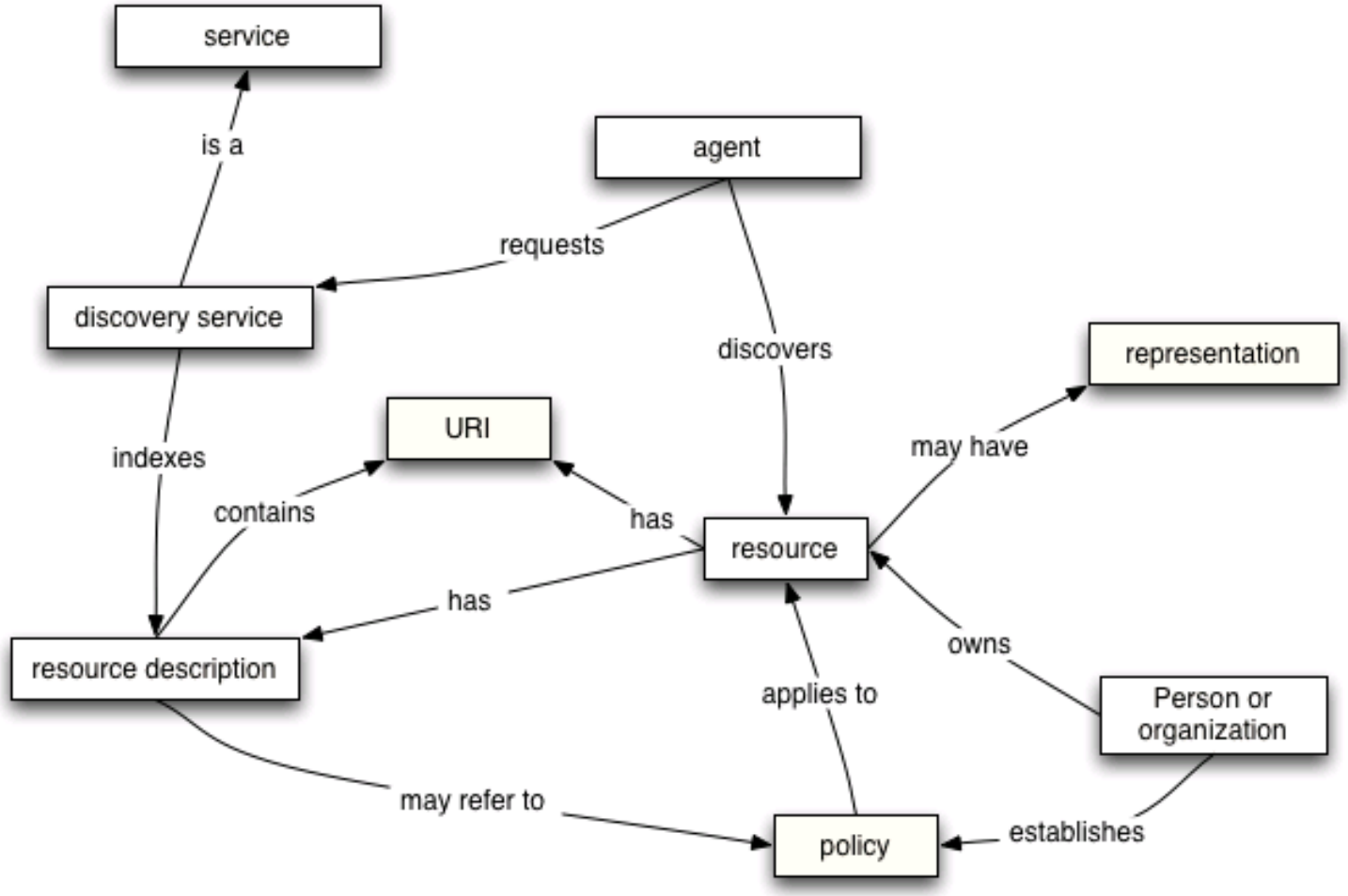
Policy Model (suite)



Message Oriented Model



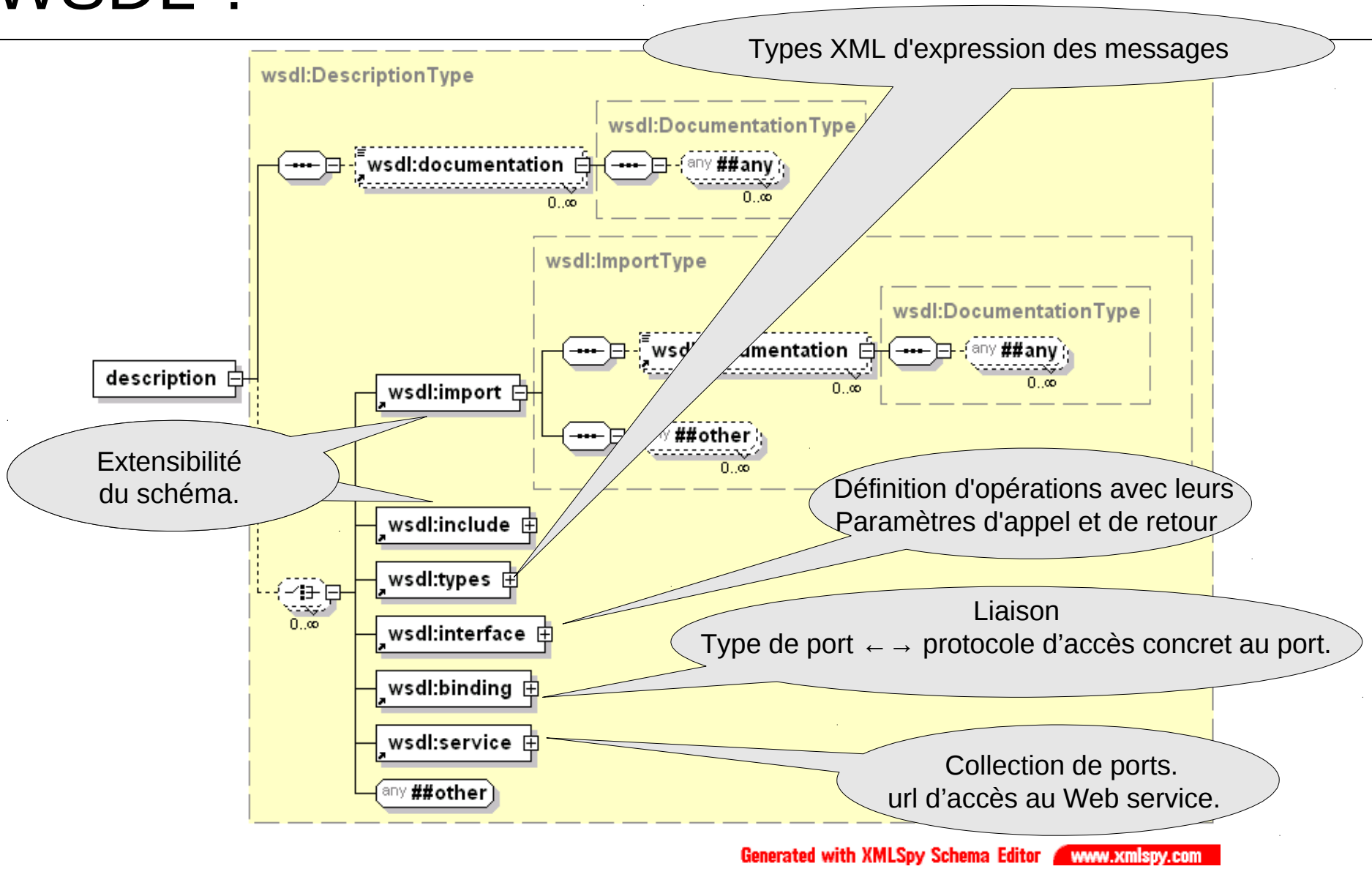
Resource Oriented Model



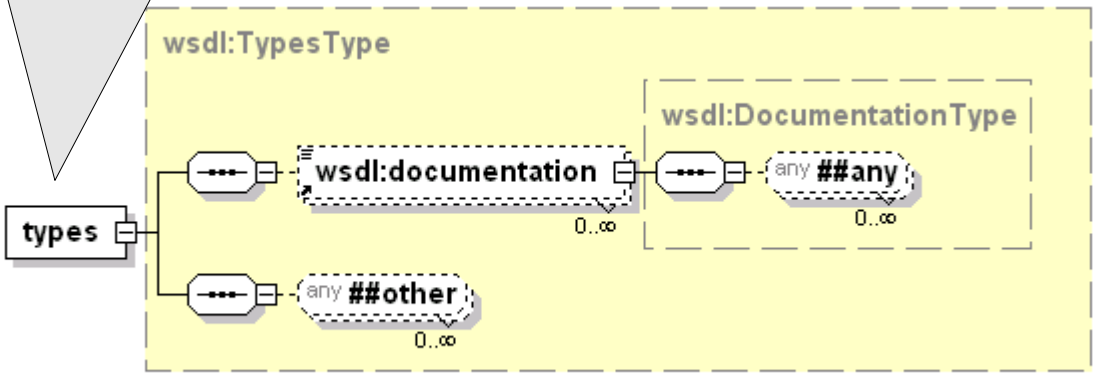
WSDL

- Spécifier ce que l'on attend d'un service, et à partir de quoi il doit l'élaborer.
  - Description de l'interface du service :
    - **Types** : (XML schema) des paramètres cités dans les messages
    - **Messages** : description structurée des données échangées
    - **Opérations** : description abstraite des actions d'un service.
    - **Type de port** : ensemble d'opérations abstraites supportées par un ou plusieurs services Web
      - Chaque opération peut avoir un message en entrée, et plusieurs message de sortie et d'erreur
    - **Binding** : un protocole de liaison concret et une spécification de format de données pour un type de port particulier.
  - Description de l'implantation du service :
    - **Port** : un point d'accès unique défini sous forme
      - d'un lien vers le service (binding) et
      - d'une adresse réseau
    - **Service Web** : un ensemble de ports

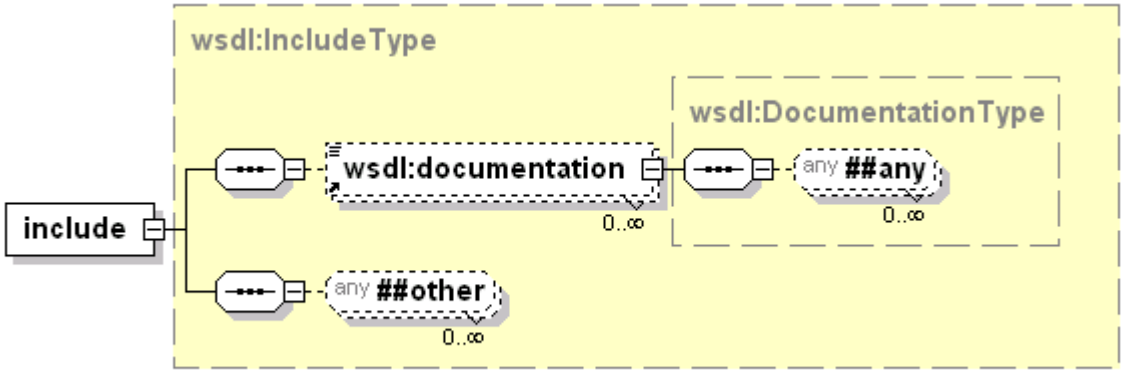
WSDL ?


Types XML d'expression des messages
wsdl:DescriptionType
wsdl:DocumentationType
wsdl:documentation
any ##any
0..∞
wsdl:ImportType
wsdl:DocumentationType
any ##any
0..∞
description
wsdl:import
##other
0..∞
Extensibilité
du schéma.
wsdl:include
wsdl:types
Définition d'opérations avec leurs
Paramètres d'appel et de retour
wsdl:interface
Liaison
Type de port ← → protocole d'accès concret au port.
wsdl:binding
wsdl:service
Collection de ports.
url d'accès au Web service.
any ##other
Generated with XMLSpy Schema Editor
www.xmlspy.com

Types XML d'expression des messages



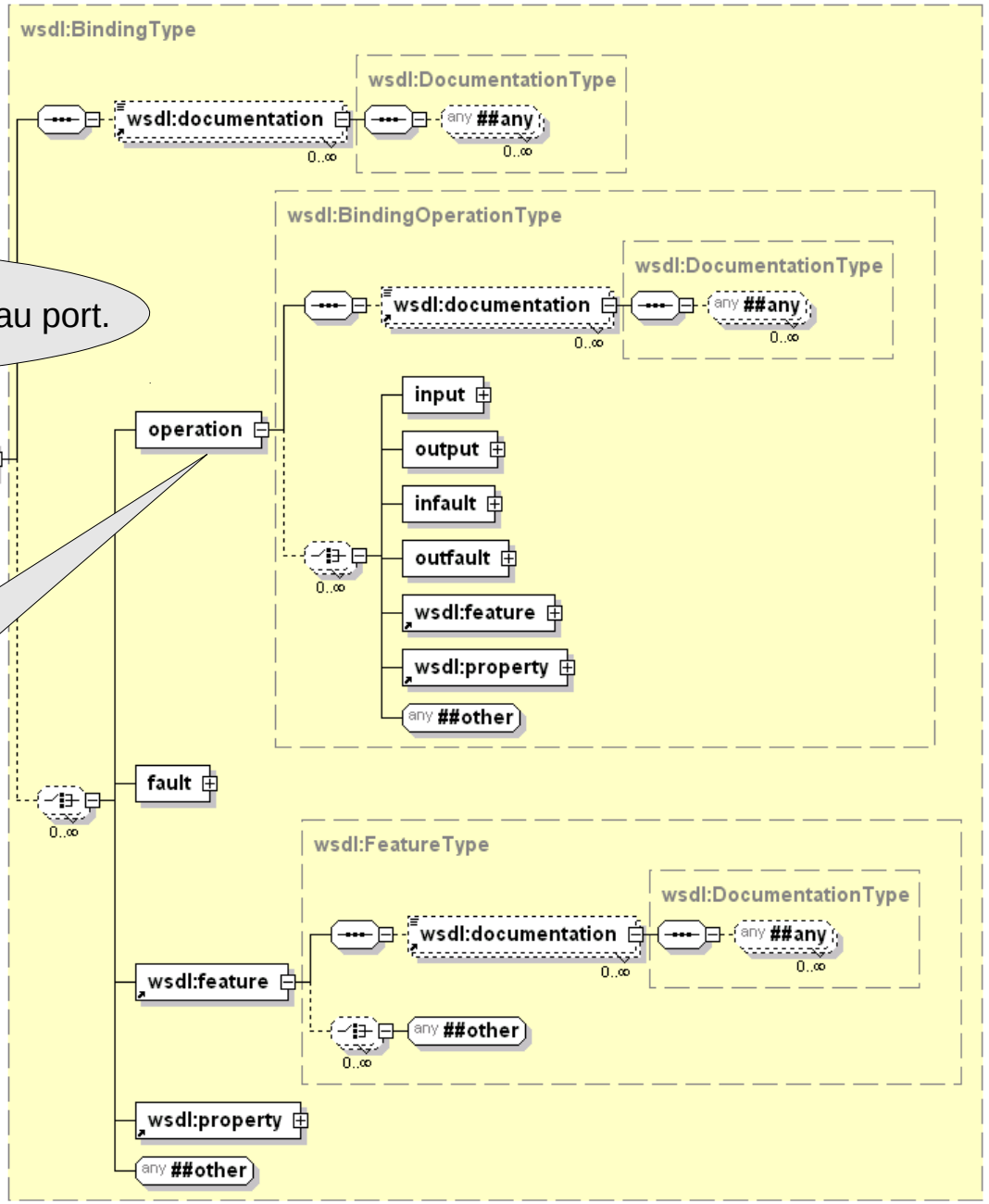
Generated with XMLSpy Schema Editor www.xmlspy.com

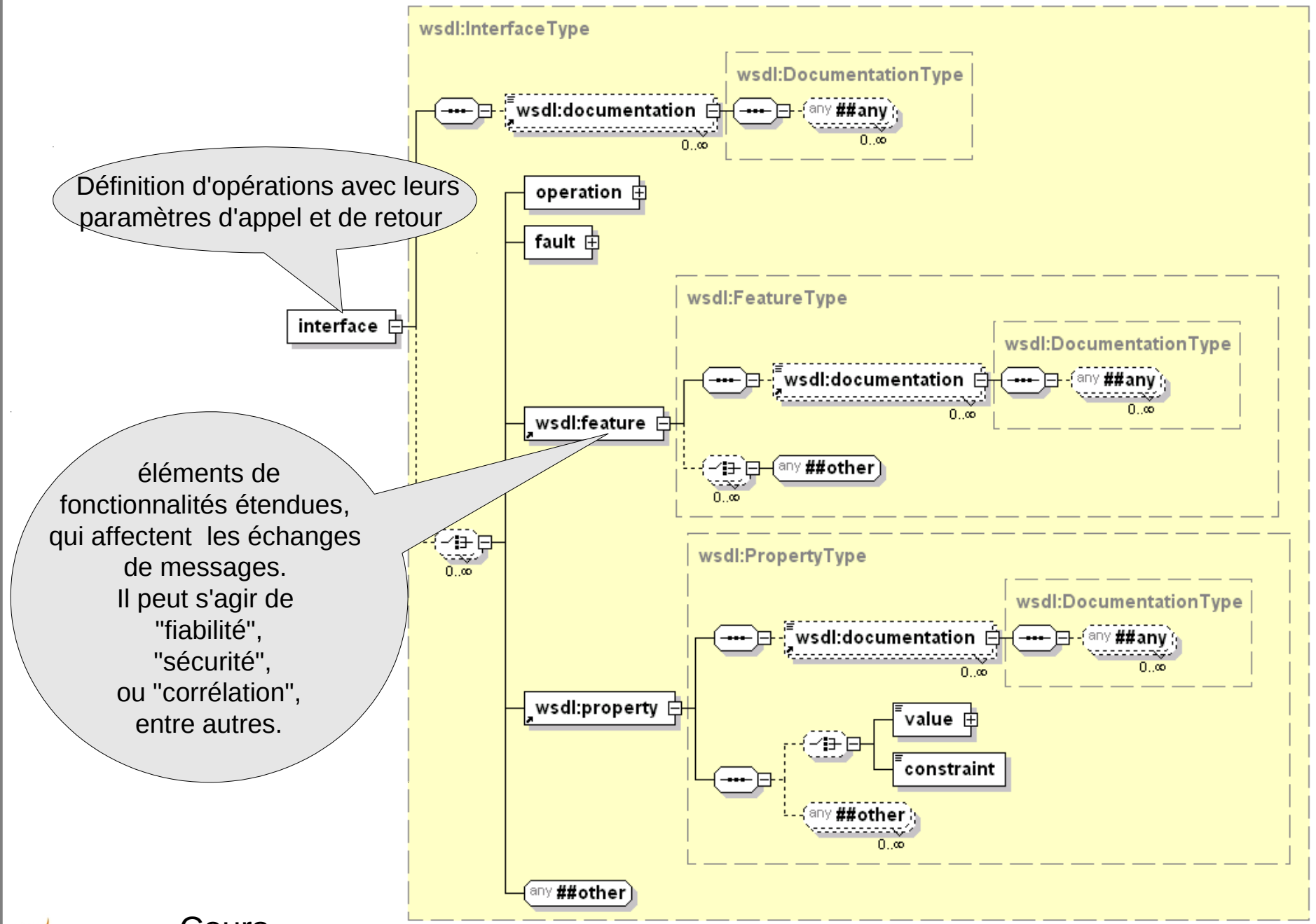


Generated with XMLSpy Schema Editor www.xmlspy.com

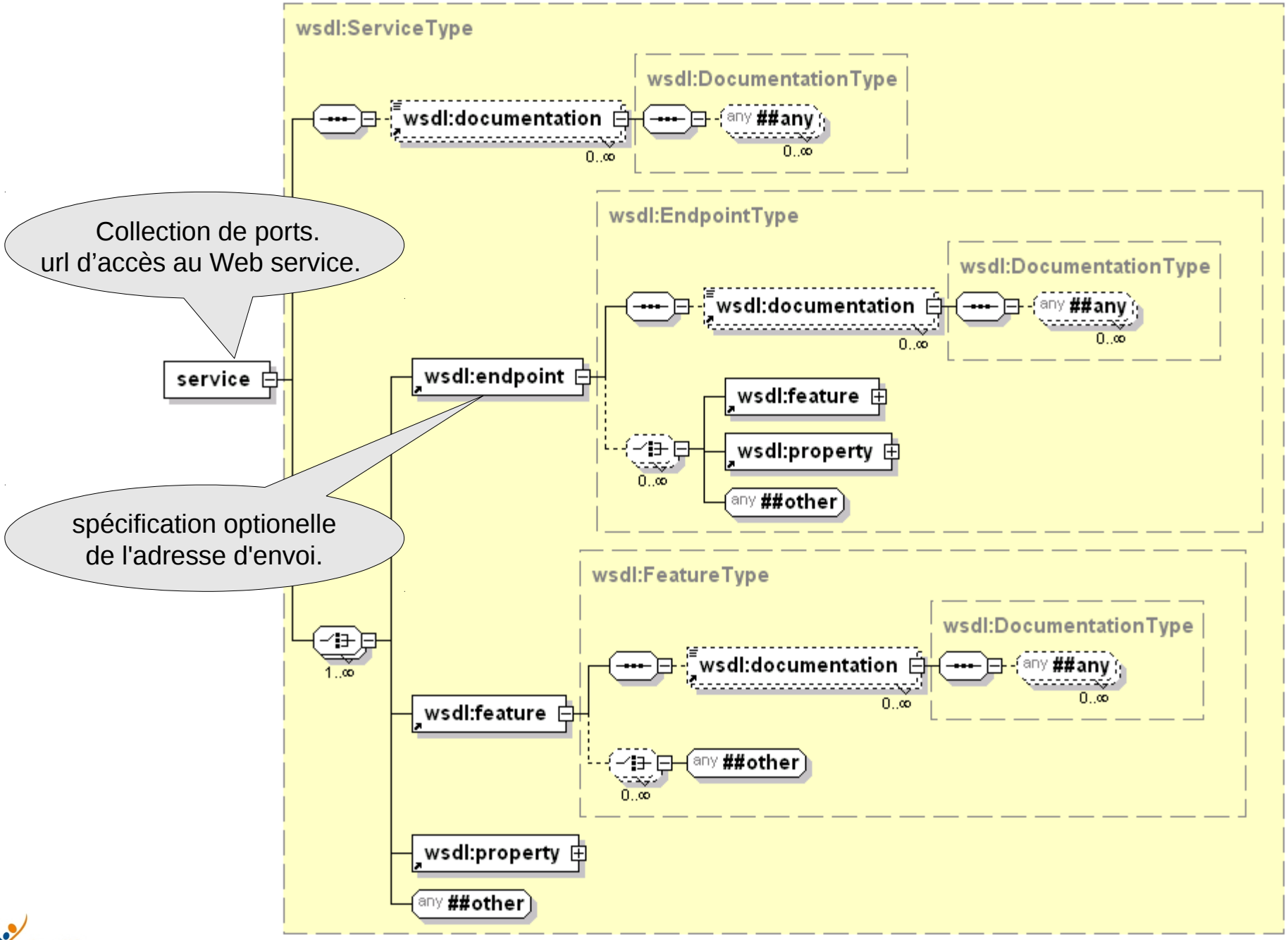
Liaison
Type de port ← → protocole d'accès concret au port.

Opération :
description abstraite
des actions d'un service.





Cours
Henry Boccon-Gibod



« Opérations »

- Une opération peut comporter un message de type input et plusieurs message de type output.
- Les opérations sont typées :
  - One-way : input sans output
  - Request-response : input suivi de output
  - Solicit-response : output suivi de input
  - Notification : output sans input

```
<types>
        <xsd:schema
          targetNamespace="urn:xml-soap-address-demo"
          xmlns:xsd="http://www.w3.org/1999/XMLSchema">
            <xsd:complexType name="phone">
              <xsd:element name="areaCode" type="xsd:int"/>
              <xsd:element name="exchange" type="xsd:string"/>
              <xsd:element name="number" type="xsd:string"/>
            </xsd:complexType>
            <xsd:complexType name="address">
              <xsd:element name="streetNum" type="xsd:int"/>
              <xsd:element name="streetName" type="xsd:string"/>
              <xsd:element name="city" type="xsd:string"/>
              <xsd:element name="state" type="xsd:string"/>
              <xsd:element name="zip" type="xsd:int"/>
              <xsd:element name="phoneNumber" type="typens:phone"/>
            </xsd:complexType>
        </xsd:schema>
</types>
```

```
<!-- message declns -->
<message name="AddEntryRequest">
      <part name="name" type="xsd:string"/>
      <part name="address" type="typens:address"/>
</message>

<message name="GetAddressFromNameRequest">
      <part name="name" type="xsd:string"/>
</message>

<message name="GetAddressFromNameResponse">
      <part name="address" type="typens:address"/>
</message>
```

```
<!-- port type declns -->
<portType name="AddressBook">
      <!– One way operation -->
      <operation name="addEntry">
            <input message="AddEntryRequest"/>
      </operation>
      <!– Request-Response operation -->
      <operation name="getAddressFromName">
            <input message="GetAddressFromNameRequest"/>
            <output message="GetAddressFromNameResponse"/>
      </operation>
</portType>
```

```
<!-- binding declns -->
<binding name="AddressBookSOAPBinding" type="AddressBook">
    <soap:binding style="rpc"  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="addEntry">
        <soap:operation soapAction=""/>
        <input>
            <soap:body use="encoded" namespace="urn:AddressFetcher2"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
            <soap:body use="encoded" namespace="urn:AddressFetcher2"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
    <operation name="getAddressFromName">
        <soap:operation soapAction=""/>
        <input><soap:body use="encoded" namespace="urn:AddressFetcher2"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
            <soap:body use="encoded"      namespace="urn:AddressFetcher2"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
</binding>
```

```
<?xml version="1.0"?>
<definitions name="MonService"
    targetNamespace="http://example.com/MonService.wsdl"
    xmlns:tns="http://example.com/MonService.wsdl"
    xmlns:xsd1="http://example.com/MonService.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

    <message name="Souscription_a_MonService">
      <part name="body"  element="xsd1:Souscription_a_MonService"/>
      <part name="souscriptionheader" element="xsd1:SubscriptionHeader"/>
    </message>

    <portType name="MonServicePortType">
        <operation name="Souscription_a_MonService">
            <input message="tns:Souscription_a_MonService"/>
        </operation>
    </portType>


<!--  suite page suivante  -->
```

```
<binding name="MonServiceSoap"  type="tns:MonServicePortType">
    <soap:binding   style="document"
transport="http://example.com/smtp"/>
    <operation name="Souscription_a_MonService">
        <input message="tns:Souscription_a_MonService">
            <soap:body parts="body" use="literal"/>
            <soap:header
                message="tns:Souscription_a_MonService"
                part="souscriptionheader"
                use="literal"/>
        </input>
    </operation>
</binding>


<service name="MonService_Service">
    <port name="MonServicePort" binding="tns:MonServiceSoap">
        <soap:address location="mailto:souscription@example.com"/>
    </port>
</service>

</definitions>
```

UDDI ?

- Quel service saura me rendre le résultat que j'attend ?
  (...et que je ne sais ou ne veux produire moi-même…)
- UDDI (*Universal Description, Discovery and Intégration*) spécifie la manière de publier et de découvrir les Web Services sur un réseau.
- Pour mettre à disposition un nouveau service, on crée un fichier appelé business Registry qui décrit le service en utilisant un langage dérivé de l'XML selon les spécifications de UDDI.

Principe d'annuaire

- UDDI permet de classer et de rechercher des Web Services.
  - le recours aux Web Services devient inefficace lorsque l'accès à l'information est encombré, le principe d'annuaire universel a été créé pour pallier cette difficulté

- Les annuaires UDDI diffèrent des annuaires LDAP (*Lightweight Directory Access Protocol*)
  - LDAP référence aussi bien des personnes que des ressources matérielles ou logicielles et gére les droits d'accès à ces ressources.

- Les annuaires UDDI sont des annuaires privés sur le Web dont l'usage vise les échanges Business to Business.
  - On y trouve aussi bien des informations techniques (documents WSDL) que des informations à caractère général sur une entreprise (page Web d'accueil ou de produits, par exemple).

- La recherche se fait grâce à un moteur de recherche intégré au site de l'opérateur UDDI choisi.
  Ce moteur de recherche vous permet d'affiner votre recherche selon plusieurs critères :
  - Nom de l'entreprise
  - La localisation de l'entreprise
  - Identifiant de l'entreprise
  - Le nom du Web Service
  - …

UDDI… recherche de Web Services

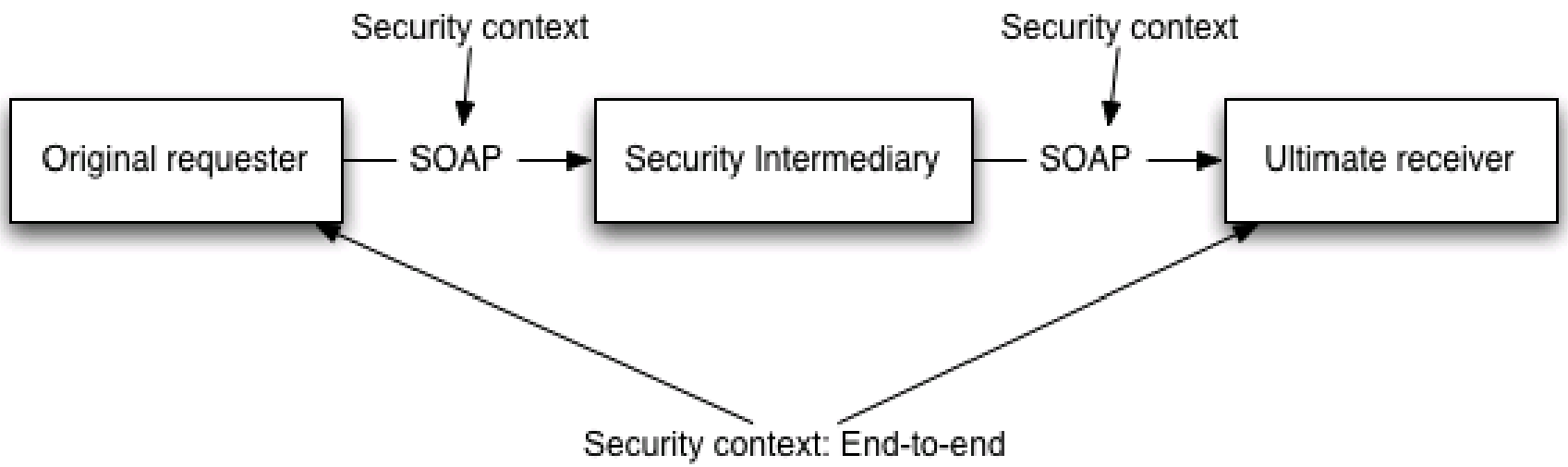
- L'API de requête regroupe les appels aux sites opérateurs qui n'ont pas besoin d'authentification particulière.
- Les annuaires UDDI ont pour but de localiser virtuellement des Web Services hébergés sur les serveurs du monde entier.
- Lors de votre recherche vous pouvez ainsi vous renseigner sur tous les services mis à disposition d'une entreprise, avoir des informations sur l'entreprise, …
- Les opérateurs UDDI vous certifient la sécurité et l'intégrité des Web Services contenus dans leurs annuaires.
- Les appels aux sites des opérateurs donnent accès aux différents éléments de l'enregistrement d'un Web Service dans un annuaire UDDI :
  - ***find_binding*** : récupère la liaison du service considéré.
  - ***find_business*** : récupère l'identité de l'entreprise productrice du Web Service.
  - ***find_relatedbusiness*** : récupère la liste des entreprises étant reliées (filiale, département, partenaire, …) à l'entreprise productrice du Web Service.
  - ***find_service*** : récupère la définition du service.
  - ***find_tmodel*** : récupère le modèle de données associé.
  - ***get_bindingDetail*** *:* récupère, par une liaison précédemment établie par ***find_binding*** les champs individuels.
  - ***get_businessDetail, get_businessDetailExt*** : récupère une entité précédemment établie par ***find_business*** les attributs individuels.
  - ***get_serviceDetail*** : récupère un service précédemment établi par ***find_service*** les attributs individuels du service (prototypes des méthodes).
  - ***get_tmodelDetail*** : récupère un modèle établie par ***find_tmodel*** les champs individuels.

UDDI: publication d'un Web Service

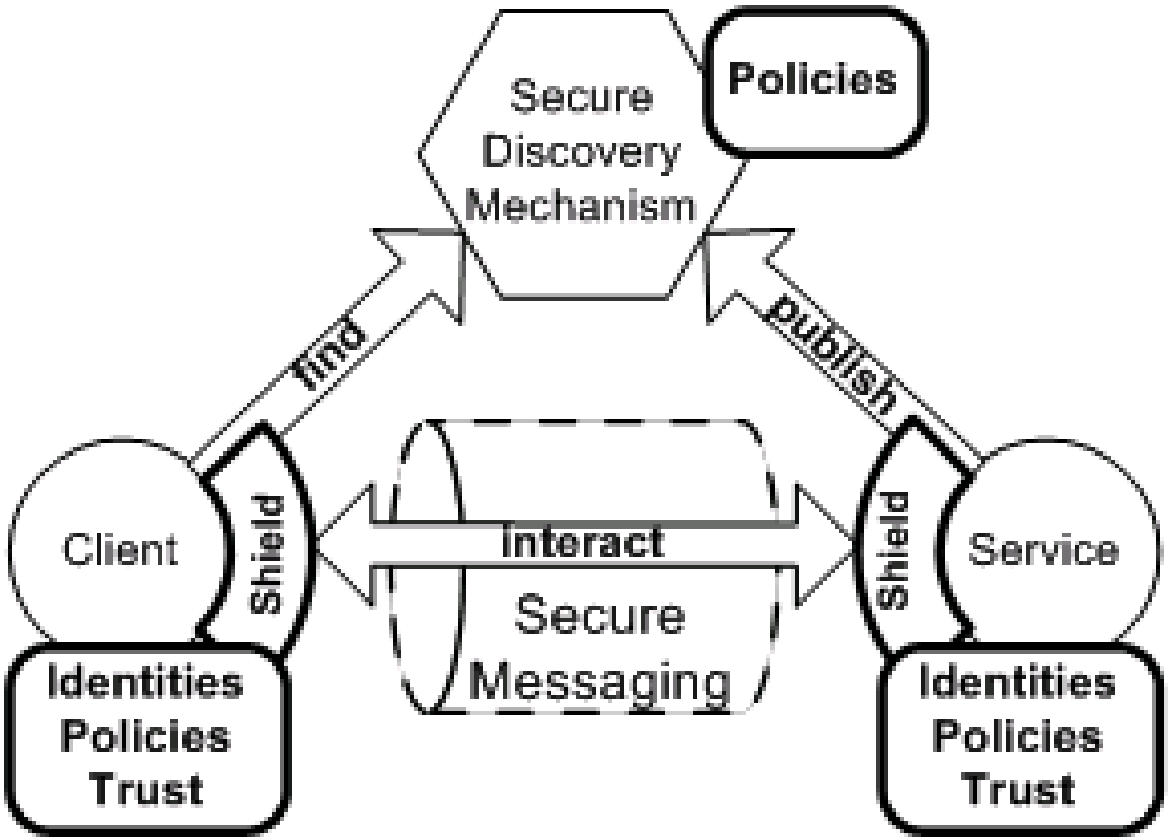
- Comme dans WSDL, la liaison UDDI regroupe, pour un protocole de communication donné, les données techniques nécessaires à l'exploitation du Web Service (adresse IP, noms de domaines, les informations sur les modalités d'usage du service, …)
- La publication par une entreprise d'un Web Service requière que celle-ci s'authentifie auprès du site de l'opérateur UDDI. L'entreprise doit s'enregistrer chez l'opérateur si cela n'est pas déjà le cas.
- Une fois le site de l'opérateur choisi, les modifications ultérieures ou la mise à jour de cet enregistrement doivent être faites auprès du même opérateur.
- Lors de l'enregistrement, vous pouvez enregistrer simultanément un ensemble d'entreprises affiliées, des relations entres entreprises indépendantes décrivant des accords croisés.

- L'API se décompose en 3 groupes :
  - La manipulation (*save* et *delete*)
  - L'authentification des commandes par jeton (*get_authToken* et *discard_authToken*)
  - L'ajout de relations inter entreprises (*joint_ventures*)
- L'avenir de l'UDDI
  - Au niveau des services, on pouvait penser que la proposition d'annuaire UDDI apporterait une solution définitive.
  - On constate qu'il n'en est rien, il ne convient pas à une problématique d'échanges entre entreprises se connaissant.
  - Il se voit maintenant concurrencer par WS-Inspection (proposé par IBM et Microsoft, pourtant à l'origine de UDDI). Moins ambitieux puisque consistant en une simple exposition, par agrégation, des services d'une entreprise, il est toutefois plus adapté à cette seconde problématique.

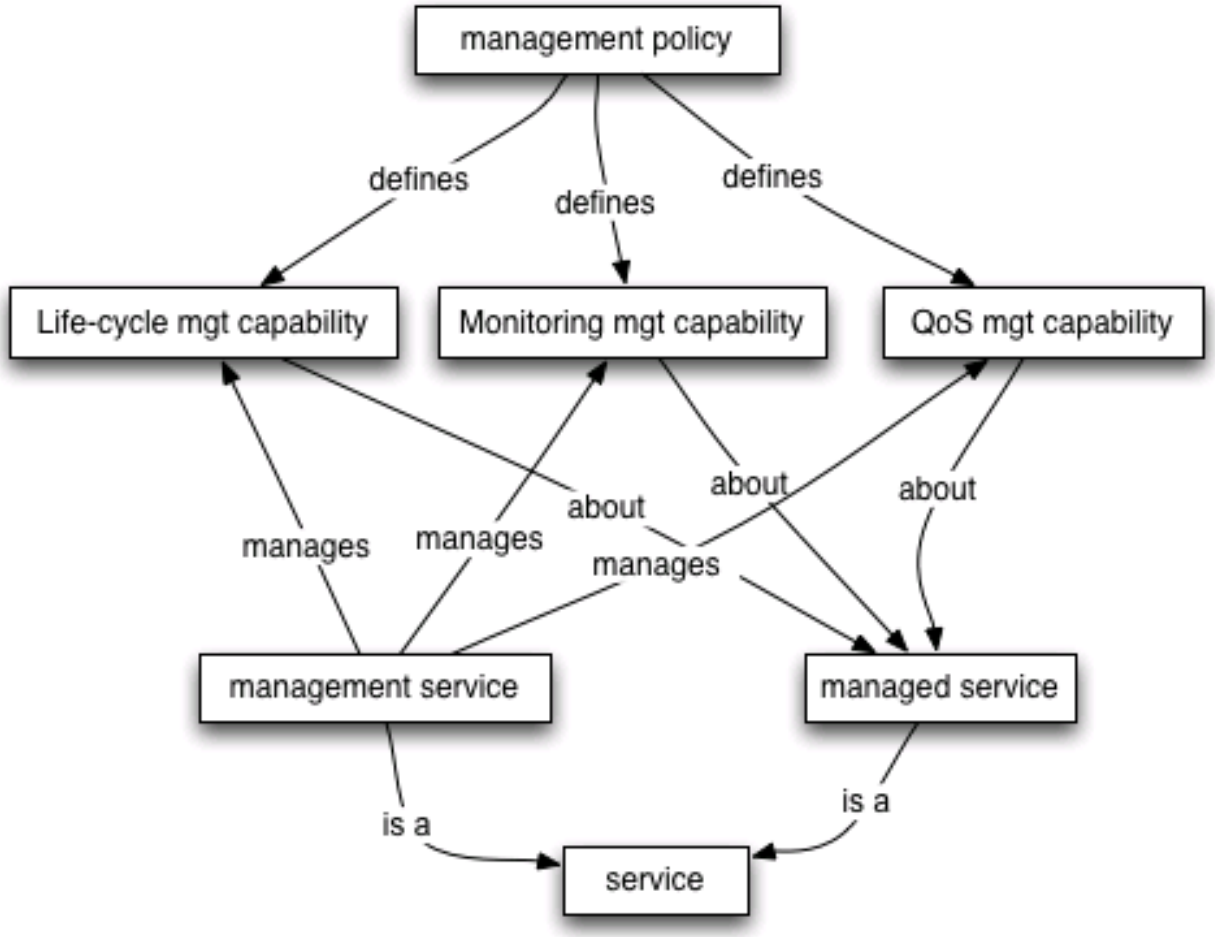
Sécurité


Security context
Security context
Original requester
SOAP
Security Intermediary
SOAP
Ultimate receiver
Security context: End-to-end

Découverte sécurisée



Assurance Qualité des Web Services



Fin du module