Description et Annuaire pour les Web Services WSDL & UDDI

Didier DONSEZ

Université Joseph Fourier (Grenoble 1)
IMAG

Didier.Donsez@imag.fr

WSDL & UDDI

- WSDL *Web Services Description Language*
  - Description de Services Web
    - http://www.w3c.org
- UDDI *Universal Description, Discovery and Integration*
  - Registre/Annuaire global de Services Web
    - http://www.uddi.org

WSDL
Web Services Description Language

WSDL

- Spécification (09/2000)
  - Ariba, IBM, Microsoft
  - TR W3C v1.1 (25/03/2001)
- Objectifs
  - Décrire les services
    comme un ensemble d'opérations et de messages abstraits
    relié (bind) à des protocoles et des serveurs réseaux
- Grammaire XML (schema XML)
  - Modulaire (import d'autres documents WSDL et XSD)

Éléments d'une définition WSDL

- <types>
  - Contient les définition de types utilisant un système de typage (comme XSD).
- <message>
  - Décrit les noms et types d'un ensemble de champs à transmettre
    - Paramêtres d'une invocation, valeur du retour, …
- <porttype>
  - Décrit un ensemble d'opérations. Chaque opération a zero ou un message en entrée, zero ou plusieurs message de sortie ou de fautes
- <binding>
  - Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP1.1, MIME, …). Un porttype peut avoir plusieurs liaisons !
- <port>
  - Spécifie un point d'entrée (endpoint) comme la combinaison d'un <binding> et d'une adresse réseau.
- <service>
  - Une collection de points d'entrée (endpoint) relatifs.

Élément <types>

- Contient les définition de types utilisant un système de typage (comme XSD).

- Exemple

```
<!-- type defs -->
 <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>
```

Élément <message>

- Décrit les noms et types d'un ensemble de champs à transmettre
  - Paramêtres d'une invocation, valeur du retour, …

Exemple

```
<!-- 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>
```

Element <porttype>

- Décrit un ensemble d'opérations.

- Plusieurs types d'opérations
  - **One-way**
    - Le point d'entrée reçoit un message (<input>).
  - **Request-response**
    - Le point d'entrée reçoit un message (<input>) et retourne un message corrélé (<output>) ou un ou plusieurs messages de faute (<fault>).
  - **Solicit-response**
    - Le point d'entrée envoie un message (<ouput>) et recoit un message corrélé (<input>) ou un ou plusieurs messages de faute (<fault>).

      Binding HTTP : 2 requêtes HTTP par exemple
  - **Notification**
    - Le point d'entrée envoie un message de notification (<ouput>)
- Paramêtres
  - Les champs des messages constituent les paramêtres (in,out, inout) des opérations

Element <porttype>

- Exemple

```
<!-- 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>
```

Élément <binding>

- Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP GET/POST, MIME, …).
  - Un porttype peut avoir plusieurs liaisons !

- Exemple de binding sur SOAP et HTTP

```
<!-- 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>
```

Élément <binding>

- Exemple de binding avec SOAP et SMTP

```
<definitions ...>
  <types>
    <schema targetNamespace="http://stockquote.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="SubscribeToQuotes">
        <complexType><all><element name="tickerSymbol" type="string"/></all></complexType>
      </element>
      <element name="SubscriptionHeader" type="uriReference"/>
    </schema>
  </types>

  <message name="SubscribeToQuotes">
    <part name="body" element="xsd1:SubscribeToQuotes"/>
    <part name="subscribeheader" element="xsd1:SubscriptionHeader"/>
  </message>

  <portType name="StockQuotePortType">
    <operation name="SubscribeToQuotes">
      <input message="tns:SubscribeToQuotes"/>
    </operation>
  </portType>
```

Élément <binding>

- ...

```
<binding name="StockQuoteSoap" type="tns:StockQuotePortType">
  <soap:binding style="document" transport="http://stockquote.com/smtp"/>
  <operation name="SubscribeToQuotes">
    <input message="tns:SubscribeToQuotes">
      <soap:body parts="body" use="literal"/>
      <soap:header message="tns:SubscribeToQuotes" part="subscribeheader" use="literal"/>
    </input>
  </operation>
</binding>
<service name="StockQuoteService">
  <port name="StockQuotePort" binding="tns:StockQuoteSoap">
    <soap:address location="mailto:subscribe@stockquote.com"/>
  </port>
</service>
</definitions>
```

Élément <service>

- Une collection de points d'entrée (endpoint) relatifs

- Exemple

```
<?xml version="1.0" ?>
<definitions name="urn:AddressFetcher"
        targetNamespace="urn:AddressFetcher2"
        xmlns:typens="urn:xml-soap-address-demo"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
...

 <!-- service decln -->
 <service name="AddressBookService">
  <port name="AddressBook" binding="AddressBookSOAPBinding">
   <soap:address location="http://www.mycomp.com/soap/servlet/rpcrouter"/>
  </port>
 </service>

</definitions>
```

Outils

- Générateur WSDL à partir de déploiement SOAP ou EJB, …
- Générateur de proxy SOAP à partir de WSDL, …

Toolkit

- IBM Web Services Toolkit
  - Outils + demo-tutorial *Gourmet2GO*

UDDI

Universal Description, Discovery and Integration

UDDI

- Spécification (09/2000)
  - Ariba, IBM, Microsoft +260 autres sociétés
- Objectifs
  - annuaire mondial d'entreprises pour permettre d'automatiser les communications entre prestataires, clients, etc.
  - plusieurs entrées indéxées : nom, carte d'identité des sociétés, description des produits, services, services applicatifs invocables à distance (références des connexions)
    - Indéxation des catalogues propriétaires

      (ebXML, RosettaNet, Ariba, Commerce One, etc.)
- Grammaire XML (schema XML)
  - Soumission/intérogation basé sur SOAP et WSDL

What is UDDI?

- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a business registry on the web
- Partnership among industry and business leaders
- *Universal Description*, *Discovery*, and *Integration*

What Problems Do We Solve?

Broader B2B

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Describe Services

Smarter Search

A flower shop in Australia wants to be "plugged in" to every marketplace in the world, but doesn't know how

Discover Services

Easier Aggregation

A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

Integrate Them Together

UDDI v1 Implementation

Manufacturers

Flower Shops

Marketplaces

UDDI Business Registry

- Programmatic descriptions of web services
- Programmatic descriptions of businesses and the services they support
- Programming model, schema, and platform agnostic
- Uses XML, HTTP, and SOAP
- Free on the Internet

How UDDI v1 Works

1. 

SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

4. 

Marketplaces, search engines, and business apps query the registry to discover services at other companies

2. 

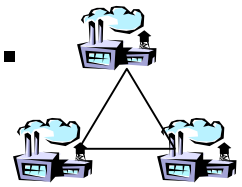
Businesses populate the registry with descriptions of the services they support

UDDI Business Registry

Business Registrations

Segrvice Type Reistrations

3. UBR assigns a programmatically unique identifier to each service and business registration

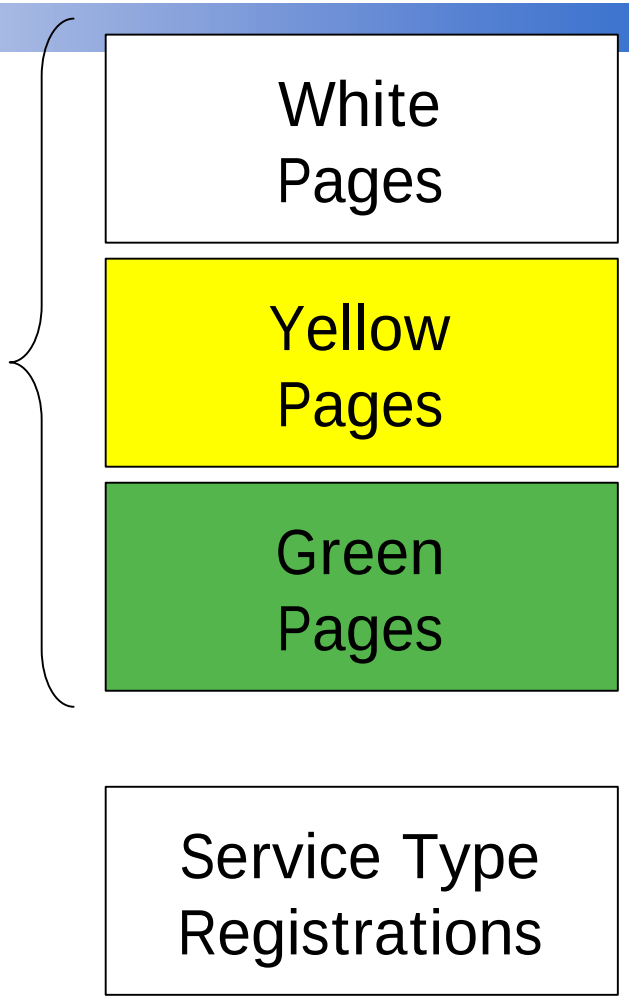
5. 

Business uses this data to facilitate easier integration with each other over the Web

Registry Data

- Businesses register public information about themselves

- Standards bodies, Programmers, Businesses register information about their Service Types



White Pages

- Business Name
- Text Description
  - list of multi-language text strings
- Contact info
  - names, phone numbers, fax numbers, web sites…
- Known Identifiers
  - list of identifiers that a business may be known by - DUNS, Thomas, other

Yellow Pages

- Business categories
  - 3 standard taxonomies in V1
    - Industry: NAICS (Industry codes - US Govt.)
    - Product/Services: UN/SPSC (ECMA)
    - Location: Geographical taxonomy
  - Implemented as name-value pairs to allow any valid taxonomy identifier to be attached to the business white page

Green Pages

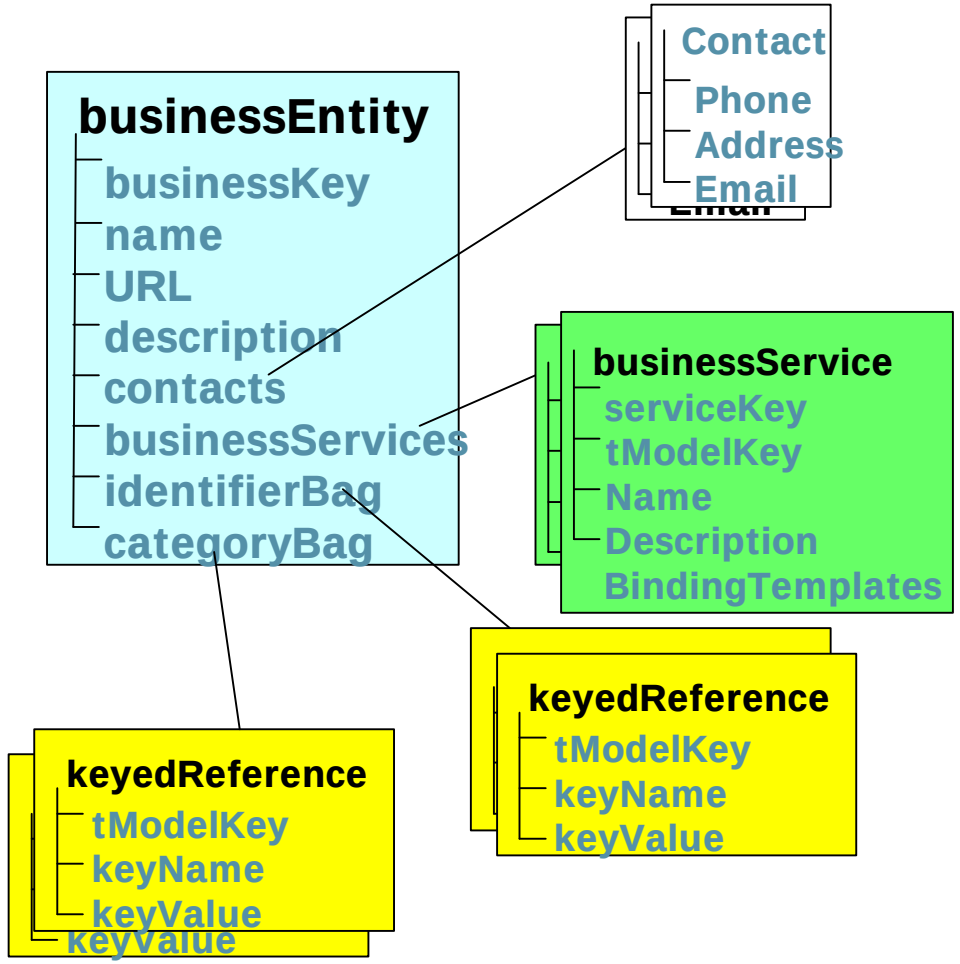
- New set of information businesses use to describe how to “do e-commerce” with them
  - Nested model
    - Business processes
    - Service descriptions
    - Binding information
  - Programming/platform/implementation agnostic
  - Services can also be categorized

Service Type Registration

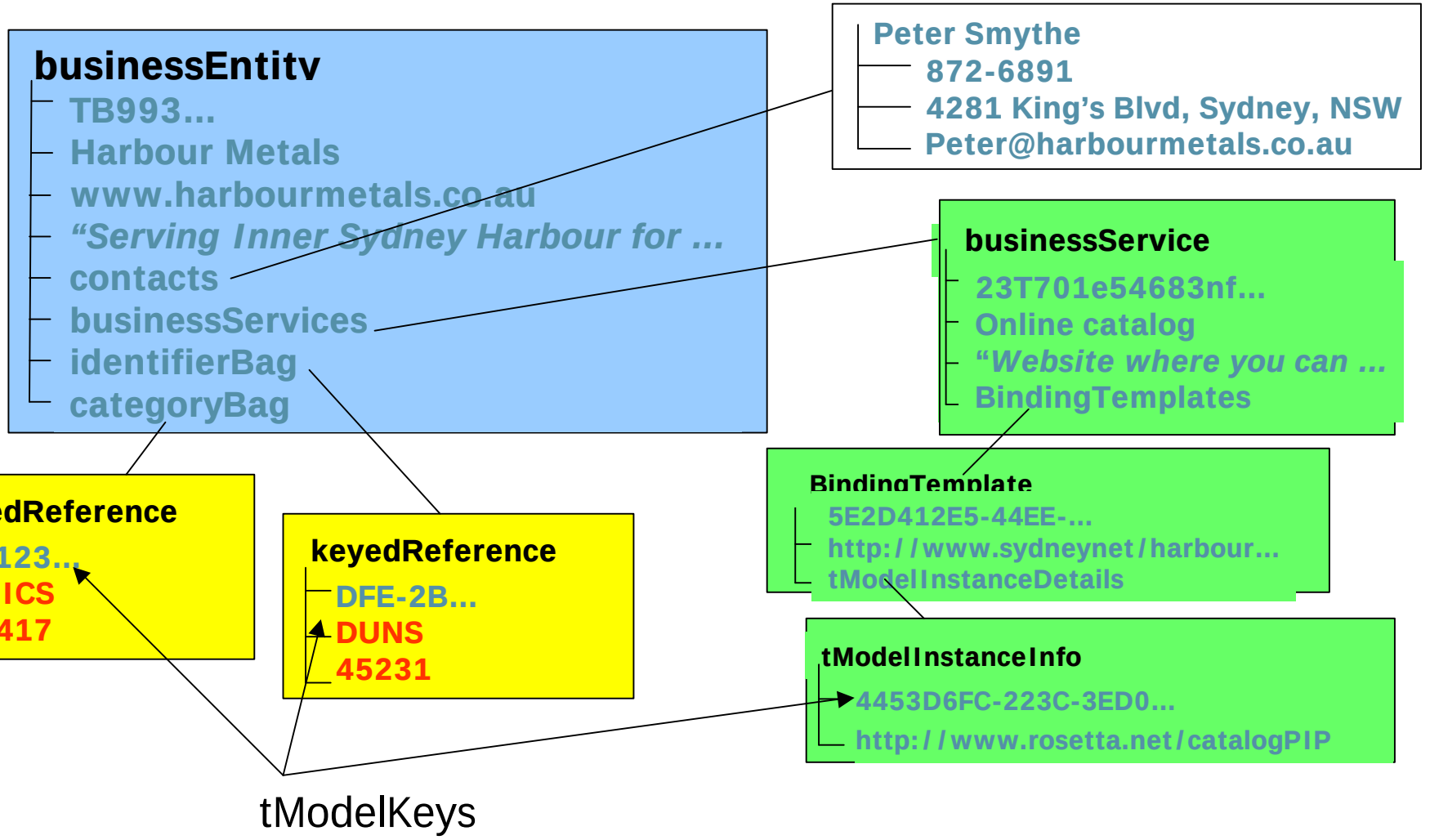
- Pointer to the namespace where service type is described
  - What programmers read to understand how to use the service
- Identifier for who published the service
- Identifier for the service type registration
  - called a tModelKey
  - Used as a signature by web sites that implement those services

Business Registration

- XML document
- Created by end-user company (or on their behalf)
- Can have multiple service listings
- Can have multiple taxonomy listings



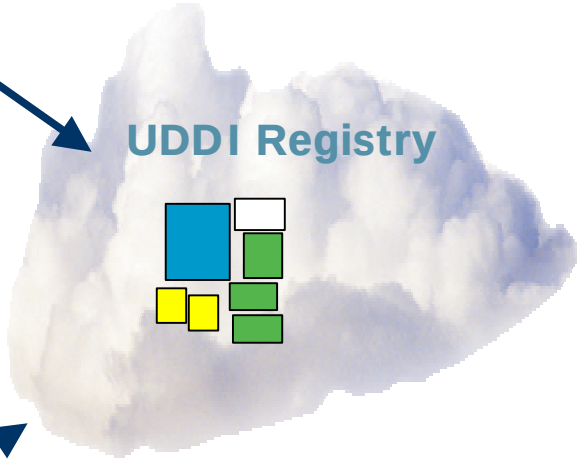
Example of a Registration



UDDI at Work

1. **Harbour Metals creates online website with local ASP**

SydneyNet.com



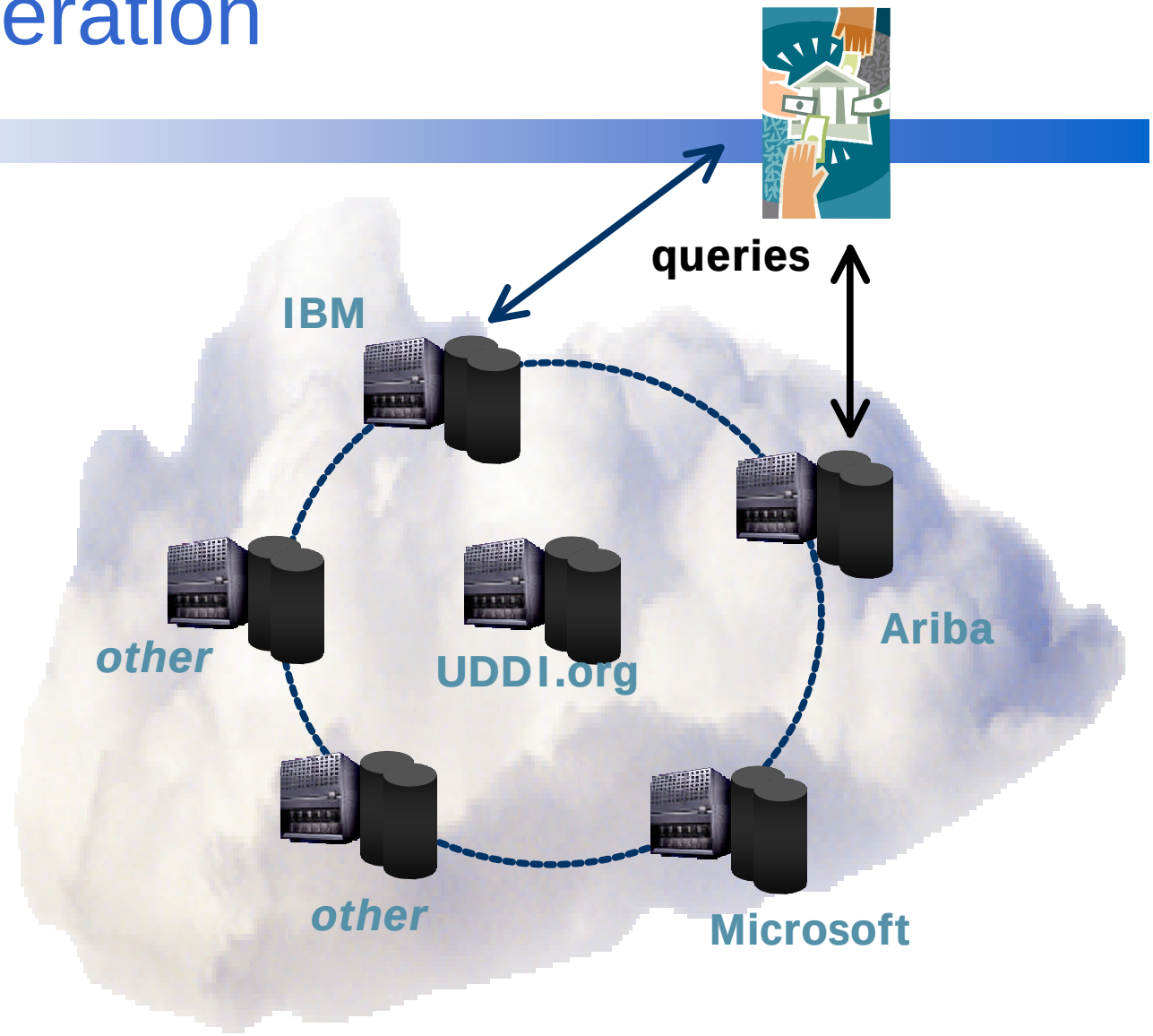
2. **ASP registers Harbour Metals with UBR**

3. **Marketplaces and search engines query UBR, cache Harbour Metals data, and bind to its services**

4. **Consumers and businesses discover Harbour Metals and do business with it**

Registry Operation

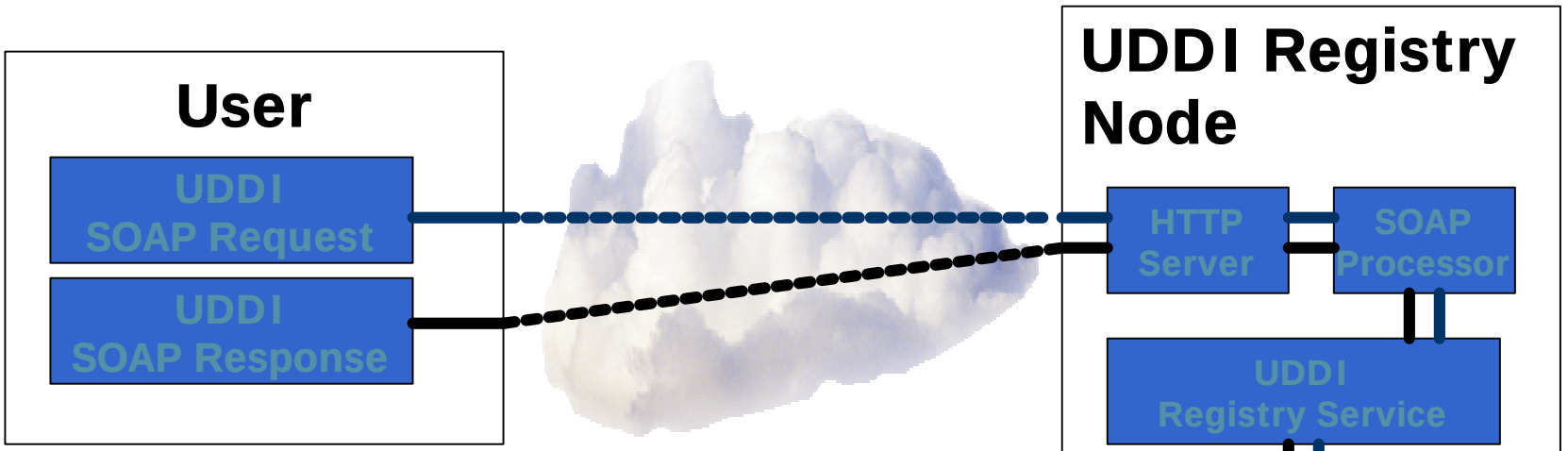
- Peer nodes (websites)
- Companies register with any node
- Registrations replicated on a daily basis
- Complete set of “registered” records available at all nodes
- Common set of SOAP APIs supported by all nodes
- Compliance enforced by business contract



Why a DNS-like Model?

- Enforces cross-platform compatibility across competitor platforms
- Demonstration of trust and openness
- Avoids tacit endorsement of any one vendor's platform
- *May migrate to a third party*

UDDI and SOAP



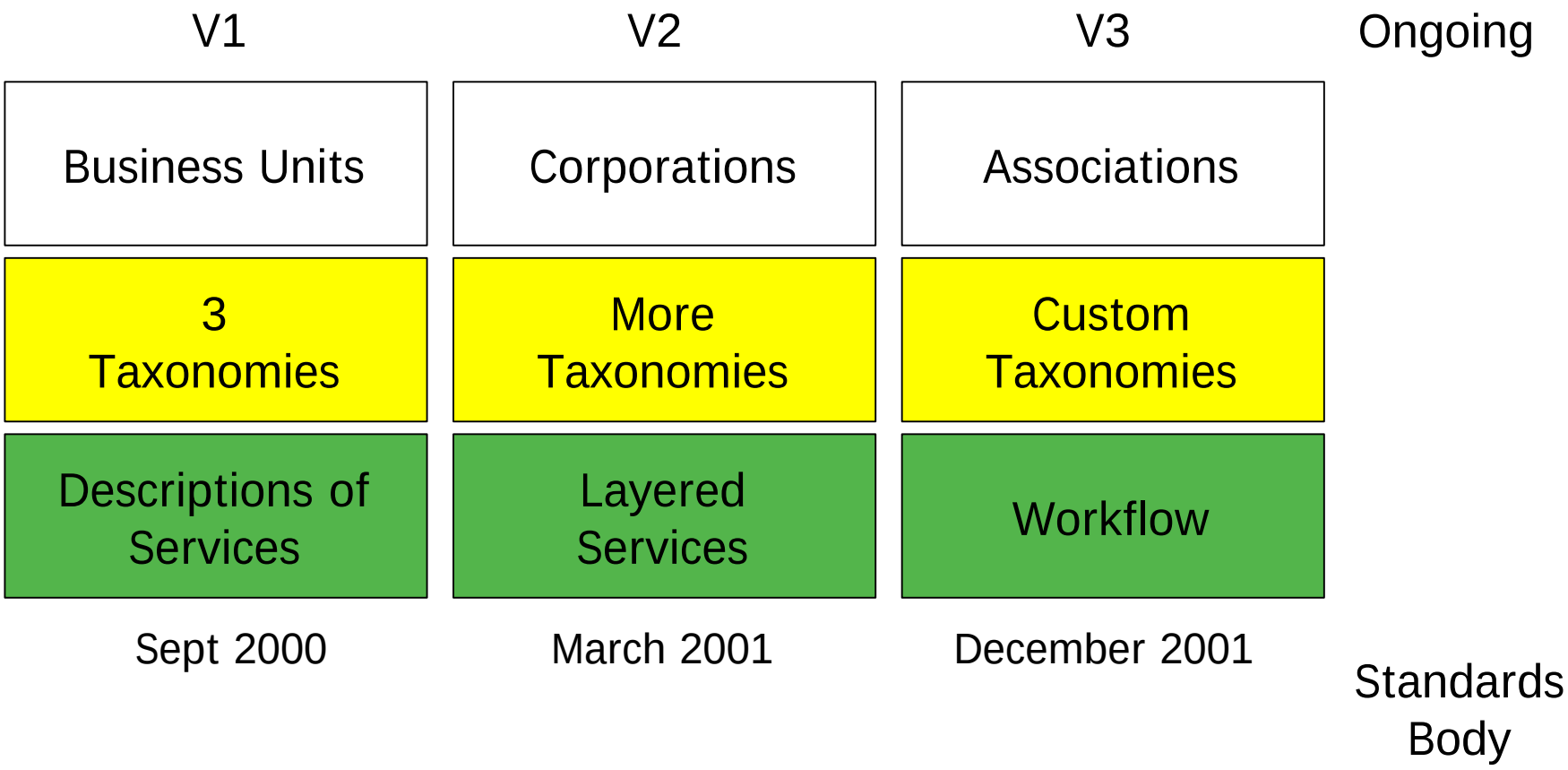
Create, View, Update, and Delete registrations

Implementation-neutral

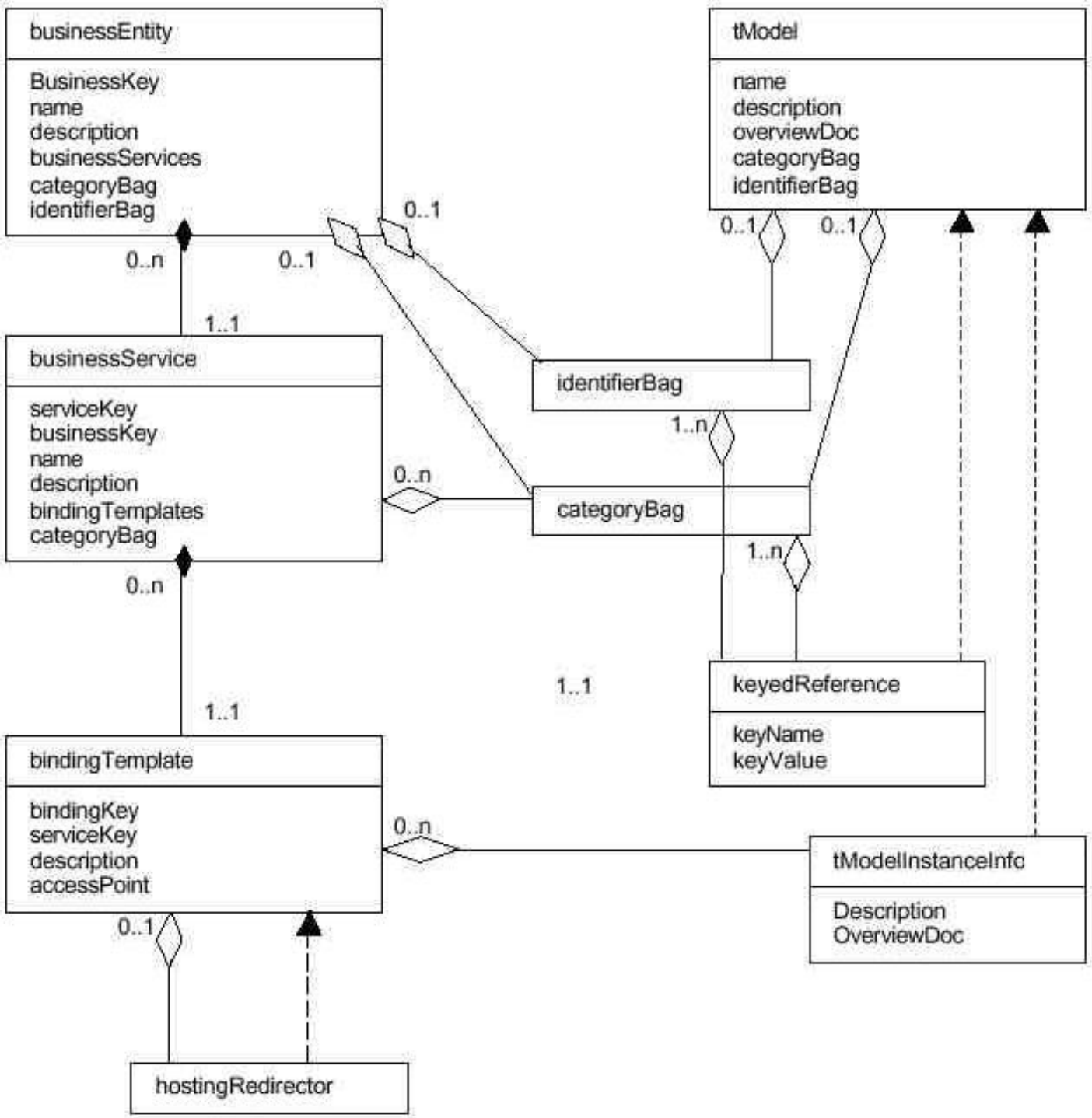
Registry APIs (SOAP Messages)

- Inquiry API
  - Find things
    - find_business
    - find_service
    - find_binding
    - find_tModel
  - Get Details about things
    - get_businessDetail
    - get_serviceDetail
    - get_bindingDetail
    - get_tModelDetail
- Publishers API
  - Save things
    - save_business
    - save_service
    - save_binding
    - save_tModel
  - Delete things
    - delete_business
    - delete_service
    - delete_binding
    - delete_tModel
  - security…
    - get_authToken
    - discard_authToken

UDDI Roadmap



Le modèle de données v1 UML



Summary

- Significant effort that unites existing standards with a shared implementation
- Open process with clear roadmap to a standards body
- Industry momentum

Bibliographie et Webographie

- Web Services
  - Philippe Mougin, Christophe Barriolade, Web Services, Business Objects and Component Models, WhitePaper Orchestral Networks, July 2001, http://www.orchestranetworks.com/us/solutions/0105_whitepaper.cfm
- WSDL
  - La spécification du W3C
    - http://www.w3.org/TR/wsdl
- UDDI
  - http://www.uddi.org
  - http://www.juddi.org
    - Projet UDDI pour Java