Chapter 6
Registering and Discovering
Web services

Web Services:
Principles & Technology
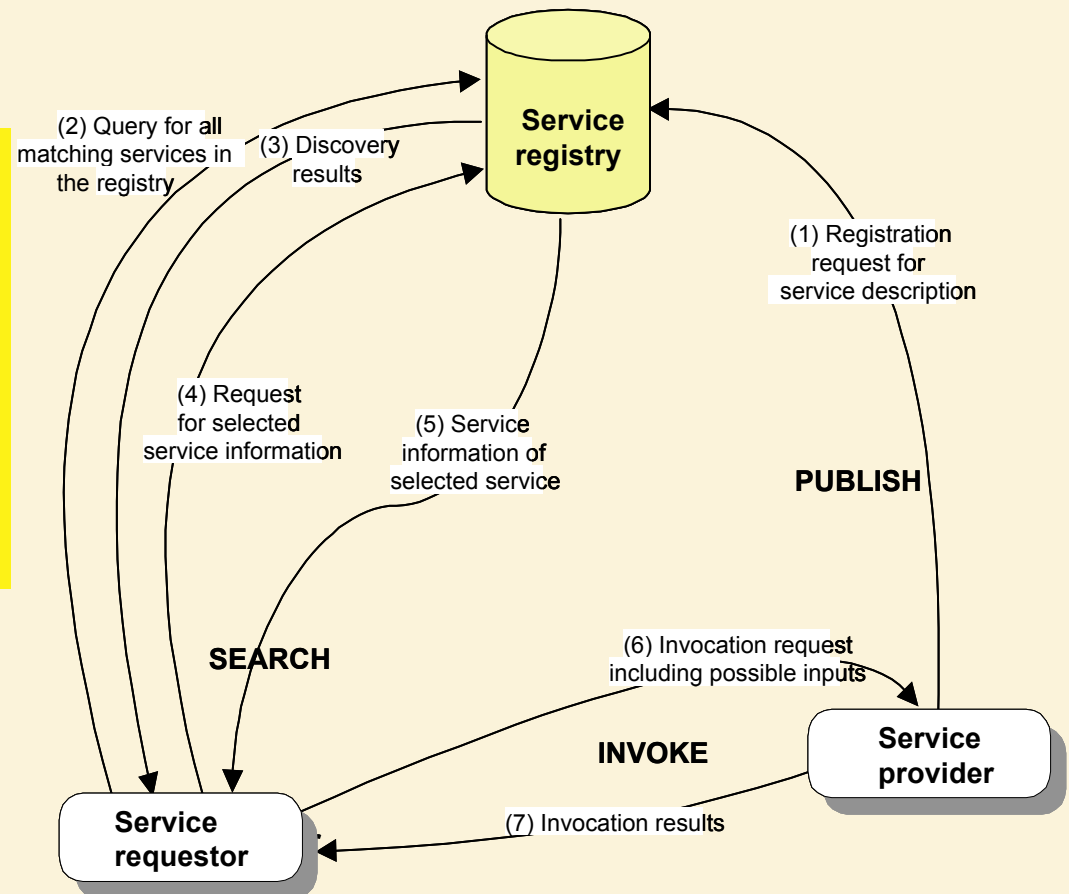
Mike P. Papazoglou
mikep@uvt.nl

Topics

- *Service registries and discovery*
- *Universal Description, Discovery and Integration*

SOA interactions between actors

- ## Problem to solve:
  - How to find the service a client wants among a large collection of services and providers.

  - The client does not necessarily need to know which provider provides the service.

Service registry

(2) Query for all matching services in the registry

(3) Discovery results

(1) Registration request for service description

(4) Request for selected service information

(5) Service information of selected service

PUBLISH

SEARCH

(6) Invocation request including possible inputs

INVOKE

Service provider

(7) Invocation results

Service requestor

Service registries

- To discover Web services, a service registry is needed. This requires describing and registering the Web service

  - Compare/Contrast with URLs

- Publication of a service requires proper description of a Web service in terms of business, service, and technical information.

- Registration deals with persistently storing the Web service descriptions in the Web services registry.

- Two types of registries can be used:

  – The document-based registry: enables its clients to publish information, by storing XML-based service documents such as business profiles or technical specifications (including WSDL descriptions of the service).

  – The metadata-based service registry: captures the essence of the submitted document.

Service discovery

- Service discovery is the process of locating Web service providers, and retrieving Web services descriptions that have been previously published.

- Interrogating services involve querying the service registry for Web services matching the needs of a service requestor.

  – A query consists of search criteria such as:

    - the type of the desired service, preferred price and maximum number of returned results, and is executed against service information published by service provider.

  – Discovering Web services is a process that is also dependent on the architecture of the service registry.

- After the discovery process is complete, the service developer or client application should know the exact location of a Web service (URI), its capabilities, and how to interface with it.

Types of service discovery

Static

- The service implementation details are bound at design time and a service retrieval is performed on a service registry.

- The results of the retrieval operation are examined usually by a human designer and the service description returned by the retrieval operation is incorporated into the application logic.

Dynamic

- The service implementation details are left unbound at design time so that they can be determined at run-time.

- The Web service requestor has to specify preferences to enable the application to **infer/reason** which Web service(s) the requester is most likely to want to invoke.

- Based on application logic quality of service considerations such as best price, performance, security certificates, and so on, the application chooses the most appropriate service, binds to it, and invokes it.
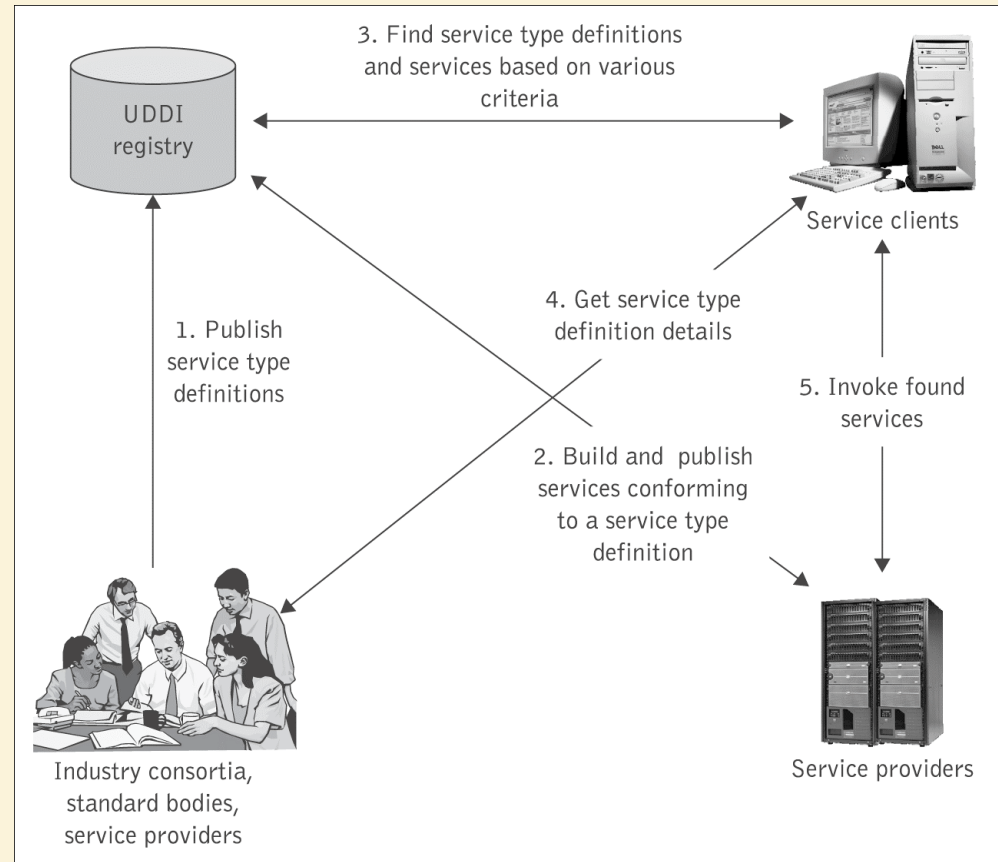
Topics

- *Service registries and discovery*
- *Universal Description, Discovery and Integration*

What is UDDI?

- The universal description, discovery, and integration is a registry standard for Web service description and discovery together with a registry facility that supports the WS publishing and discovery processes.

- UDDI enables a business to:
  - **describe** its business and its services;
  - **discover** other businesses that offer desired services;
  - **integrate (interoperate)** with these other businesses.

- Conceptually, a UDDI business registration consists of three inter-related components:
  - "white pages" (address, contact, and other key points of contact);
  - "yellow pages" classification info. based on standard industry taxonomies; and
  - "green pages", the technical capabilities and information about services.

The UDDI usage model

- An enterprise may set up multiple *private* UDDI registries in-house to support intranet and e-Business operations.

- Public UDDI registries can be set up by customers and business partners of an enterprise.
  - Services must be published in a public UDDI registry so that potential clients and service developers can discover them.

UDDI registry

3. Find service type definitions and services based on various criteria

Service clients

1. Publish service type definitions

4. Get service type definition details

5. Invoke found services

2. Build and publish services conforming to a service type definition

Industry consortia, standard bodies, service providers
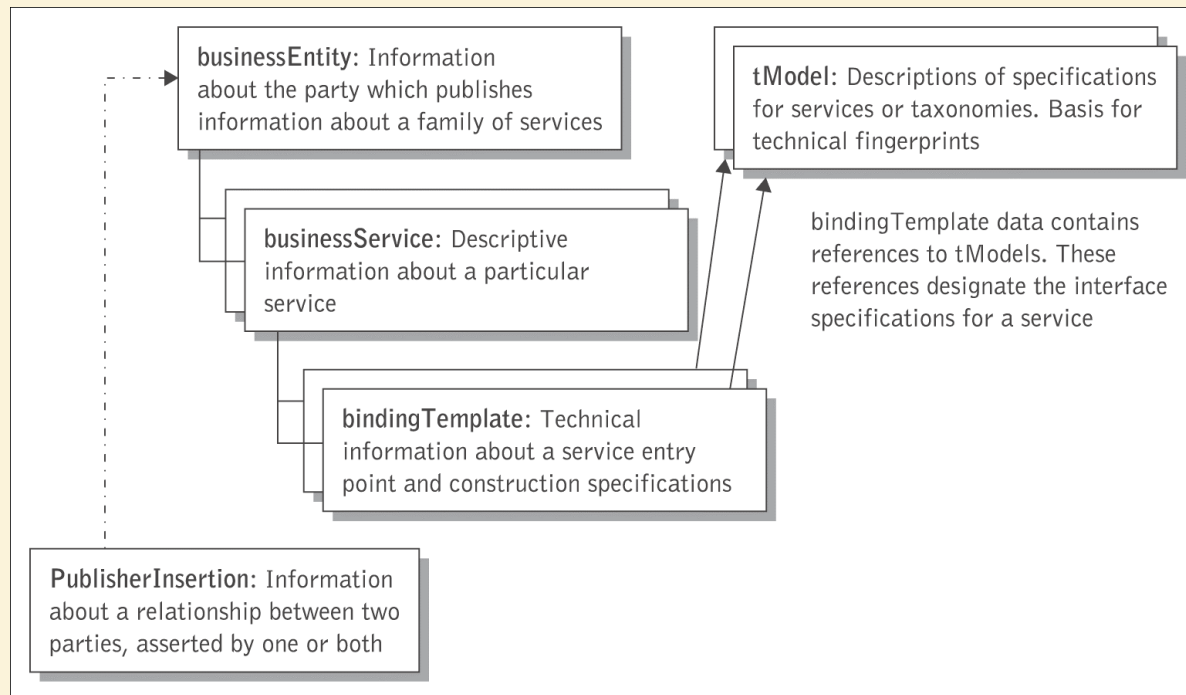
Service providers
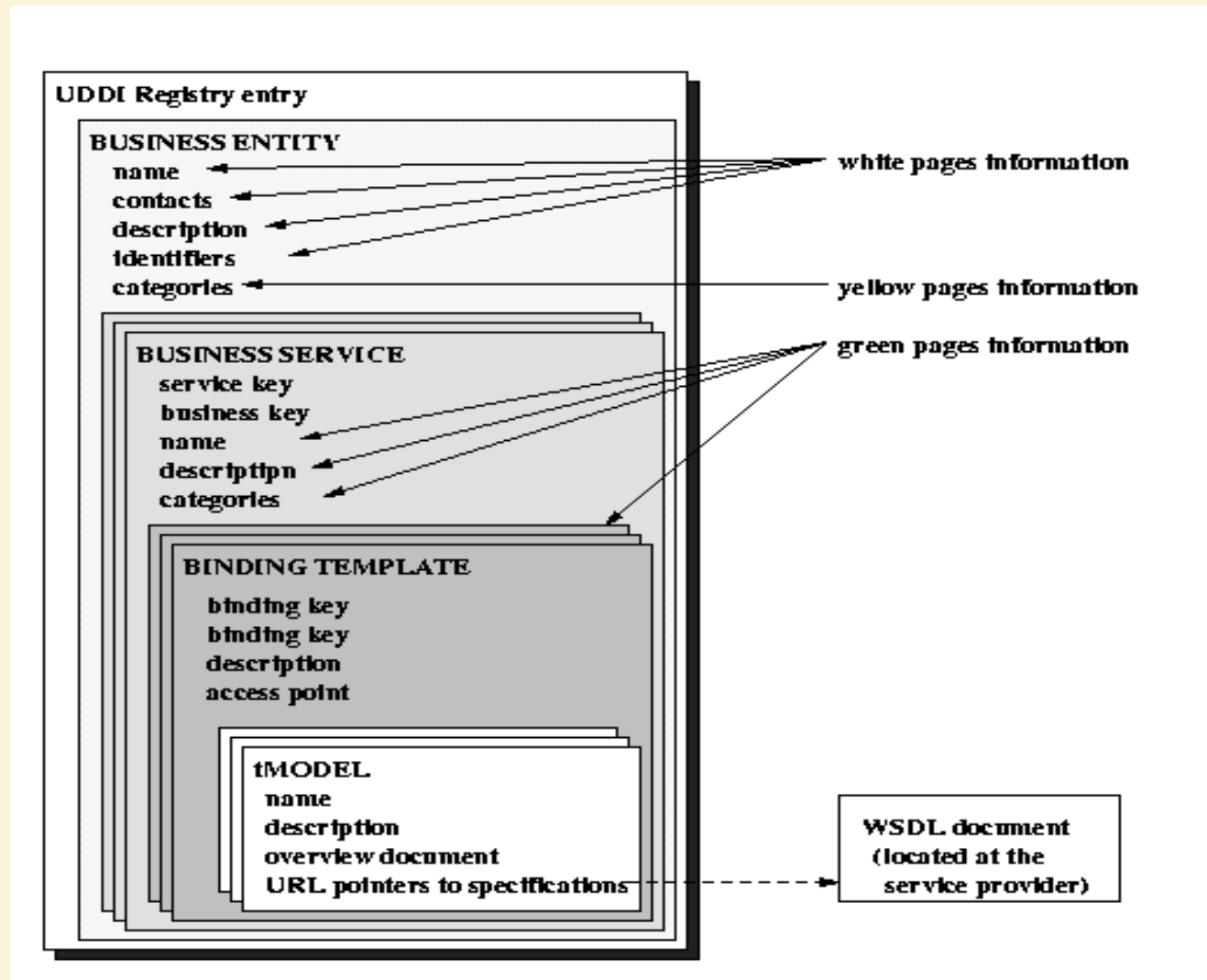
UDDI – main characteristics

- UDDI provides a mechanism to categorize businesses and services using **taxonomies**.
  - Service providers can use a taxonomy to indicate that a service implements a specific domain standard, or that it provides services to a specific geographic area
  - UDDI uses standard taxonomies so that information can be discovered on the basis of categorization.
- **UDDI business registration: an XML document** used to describe a business entity and its Web services.
- UDDI is not bound to any technology. In other words,
  - An entry in the UDDI registry can contain any type of resource, independently of whether the resource is XML based or not, e.g., the UDDI registry could contain information about an enterprise's electronic document interchange (**EDI**) system or even a service that, uses the **fax machine** as its primary communication channel.
  - While UDDI itself uses XML to represent the data it stores, it allows for other kinds of technology to be registered.

UDDI – Data Structures

- UDDI also defines a data structure standard for representing company and service description information. The UDDI XML schema defines four core types of information.These are
  - *businessEntity*: a description of the organization that provides the service.
  - *businessService*: a list of all the Web services offered by the business entity.
  - *bindingTemplate*: describes the technical aspects of the service being offered.
  - *tModel*: ("technical model")is a generic element that can be used to store technical information on how to use the service, conditions for use, guarantees, etc.

businessEntity: Information about the party which publishes information about a family of services

businessService: Descriptive information about a particular service

bindingTemplate: Technical information about a service entry point and construction specifications

PublisherInsertion: Information about a relationship between two parties, asserted by one or both

tModel: Descriptions of specifications for services or taxonomies. Basis for technical fingerprints

bindingTemplate data contains references to tModels. These references designate the interface specifications for a service
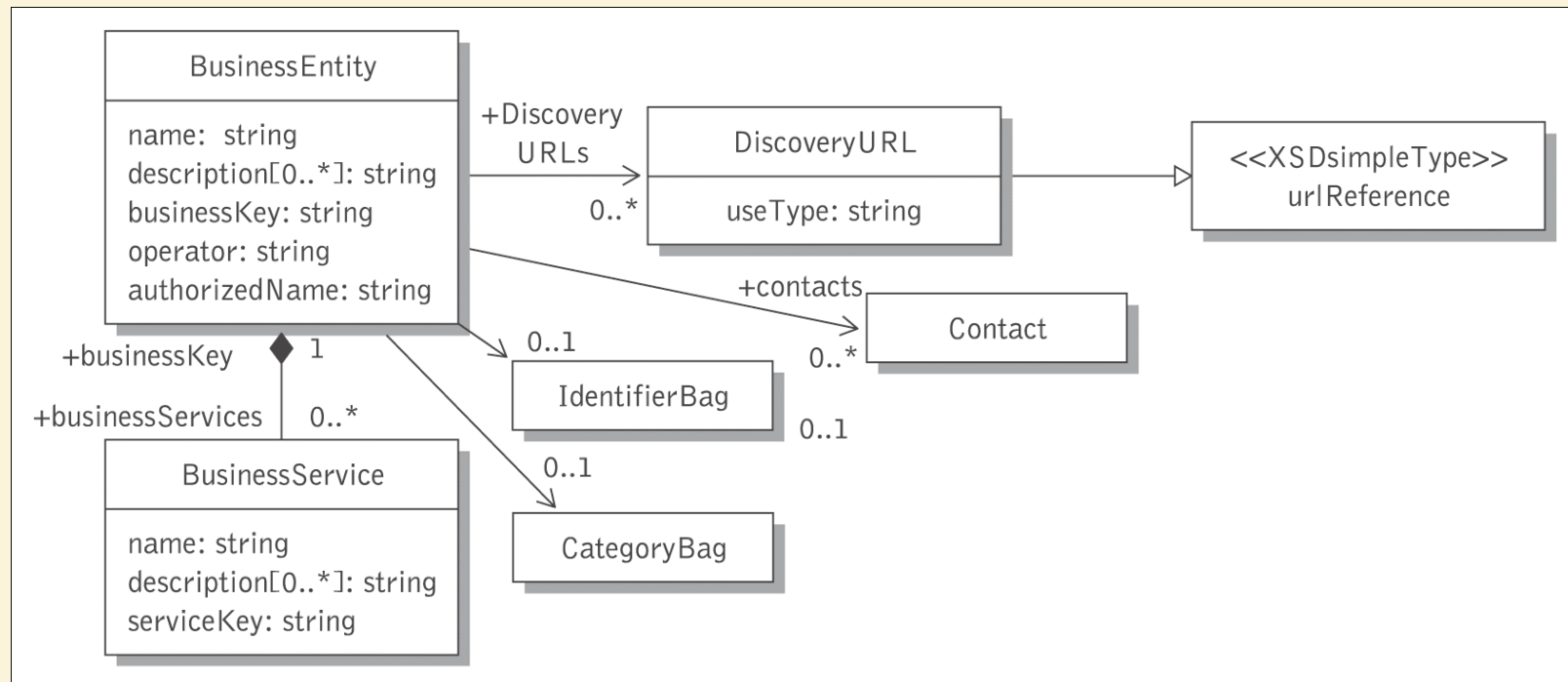
UDDI – Data Structures

Business entity

- The generic white and yellow pages information about a service provider is stored in the businessEntity data structure.

Example – Business entity

```
<businessEntity businessKey="d2300-3aff-.."
                xmlns = "urn:uddi-org:api_v2">
<name xml: lang="en"> Automotive Equipment Manufacturing Inc. </name>
<description xml: lang="en">
   Automotive Equipment, Accessories and Supplies for European firms
</description>
<contacts>
   <contact useType="Sales Contact">
      <description xml: lang="en"> Sales Representative </ description>
      <personName> Reginald Murphy </personName>
      <email useType="primary"> joe.murphy@automeq.com </email>
      <address useType="http">
         <addressLine> http://www.medeq.com/sales/ </addressLine>
      </address>
   </contact>
</contacts>
<businessServices>
   <!-- Business service information goes here -->
</businessServices>
<identifierBag>
   <!-- DUNS Number identifier System -->
   <keyedReference keyName="DUNS Number" keyValue="…" tModelKey="…"/>
</identifierBag>
<categoryBag>
   <!—North American Industry Classification System (NAICS) -->
   <keyedReference
           keyName="Automotive parts distribution" keyValue="…" tModelKey="…"/>
    ……
</categoryBag>
</businessEntity>
```
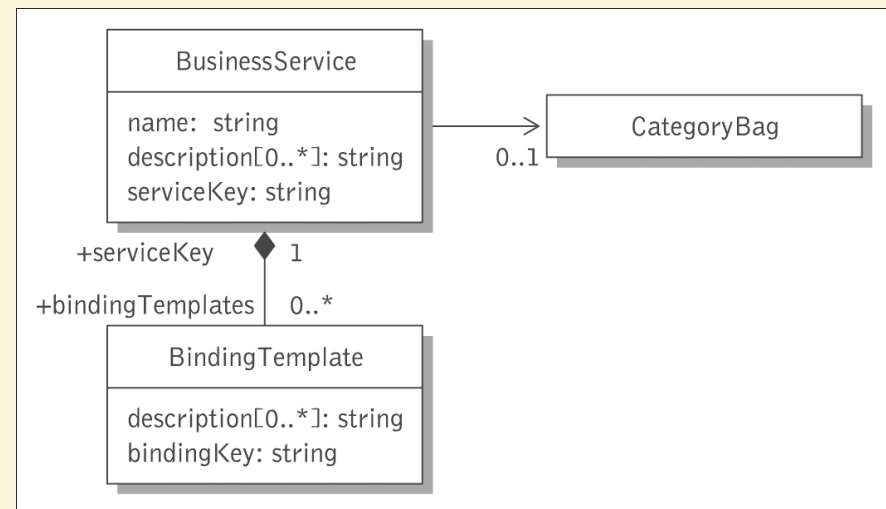
Business service

- The services provided by a business entity are described in business terms using businessService elements.

- A businessEntity can have several businessServices but a businessService belongs to one businessEntity.

- A businessService contains:
  - a **serviceKey** that uniquely identifies the service and the businessEntity (not necessarily the same as where the businessService is found),
  - **name**: as before,
  - **description**: as before,
  - **categoryBag**: as before,
  - **bindingTemplates**: a list to all the bindingTemplates for the service with the technical information on how to access and use the service.

Example – Business service

```
<businessServices>
    <businessService serviceKey=" ">
        <name> Search the Automotive Equipment Manufacturing parts Registry </name>
            <description lang="en">
                Get to the Automotive Equipment Manufacturing parts Registry
            </description>
            <bindingTemplates>
                <bindingTemplate bindingKey="..">
                    <description lang="en">
                        Use your Web Browser to search the parts registry
                    </description>
                    <accessPoint URLType="http">
                        http://www.automeq.com/b2b/actions/search.jsp
                    </accessPoint>
                    <tModelInstanceDetails>
                        <tModelInstanceInfo
                            tModelKey="uddi:.."/>
                    <tModelInstanceDetails>
                </bindingTemplate>
            </bindingTemplates>
    </businessService>
</businessServices>
```

Binding template: "green pages"

- A binding template contains the technical information associated to a particular service:
  - **bindingKey**
  - **serviceKey**
  - **description**
  - **accessPoint**: the network address of the service being provided
  - **tModels**: a list of entries corresponding to tModels associated with this particular binding
  - **categoryBag**: additional information about the service and its binding (e.g., whether it is a test binding, it is on production, etc).
- A businessService can have several bindingTemplates but a binding Template has only one businessService.
- The binding template is seen as a folder, where all the technical information of a service is put together.

```
<bindingTemplate bindingKey="..">
    <description lang="en">
        Use your Web Browser to search the parts registry
    </description>
    <accessPoint URLType="http">
        http://www.automeq.com/b2b/actions/search.jsp
    </accessPoint>
    <tModelInstanceDetails>
        <tModelInstanceInfo
            tModelKey="uddi:.."/>
    <tModelInstanceDetails>
</bindingTemplate>
```

tModel

- A **tModel** is a generic container of information which contains technical information associated with the use of a Web service:
  - the actual interface and protocol used, including a pointer to the WSDL description;
  - description of the business protocol and conversations supported by the service.

- A tModel can point to other tModels and eventually different forms of standardized tModels:
  - tModel for WSDL services, tModels for EDI-based services, RosettaNet Partner Interface Processes (PIPs) …

URL pointing to a zipped file where a description of the PIP 3A1 "Request Quote" can be found

```
<tModel   tModelKey="…" >

  <name> RosettaNet-Org </name>
  <description xml:lang="en">
      Supports a process for trading partners to request and provide quotes
  </description >

  <overviewDoc>
    <description xml:lang="en">
        This compressed file contains the specification in a word
        document, the html guidelines document, and the XML schemas.
    </ description>
    <overviewURL>
          http://www.rosettanet.org/rosettanet/Doc/0/
                        K96RPDQA97A1311M0304UQ4J39/3A1_RequestQuote.zip
    </overviewURL>
  </overviewDoc>

  <categoryBag>
      <keyedReference   keyName=" Trading quote request and provision"
                        keyValue=" 80101704"  tModelKey=" ….."/>
  </categoryBag>
</tModel>
```

List of name–value pairs that are used to record specific taxonomy information, e.g., industry, product or geographic codes, for this <tModel>

<bindingTemplate> and <tModel> in UML
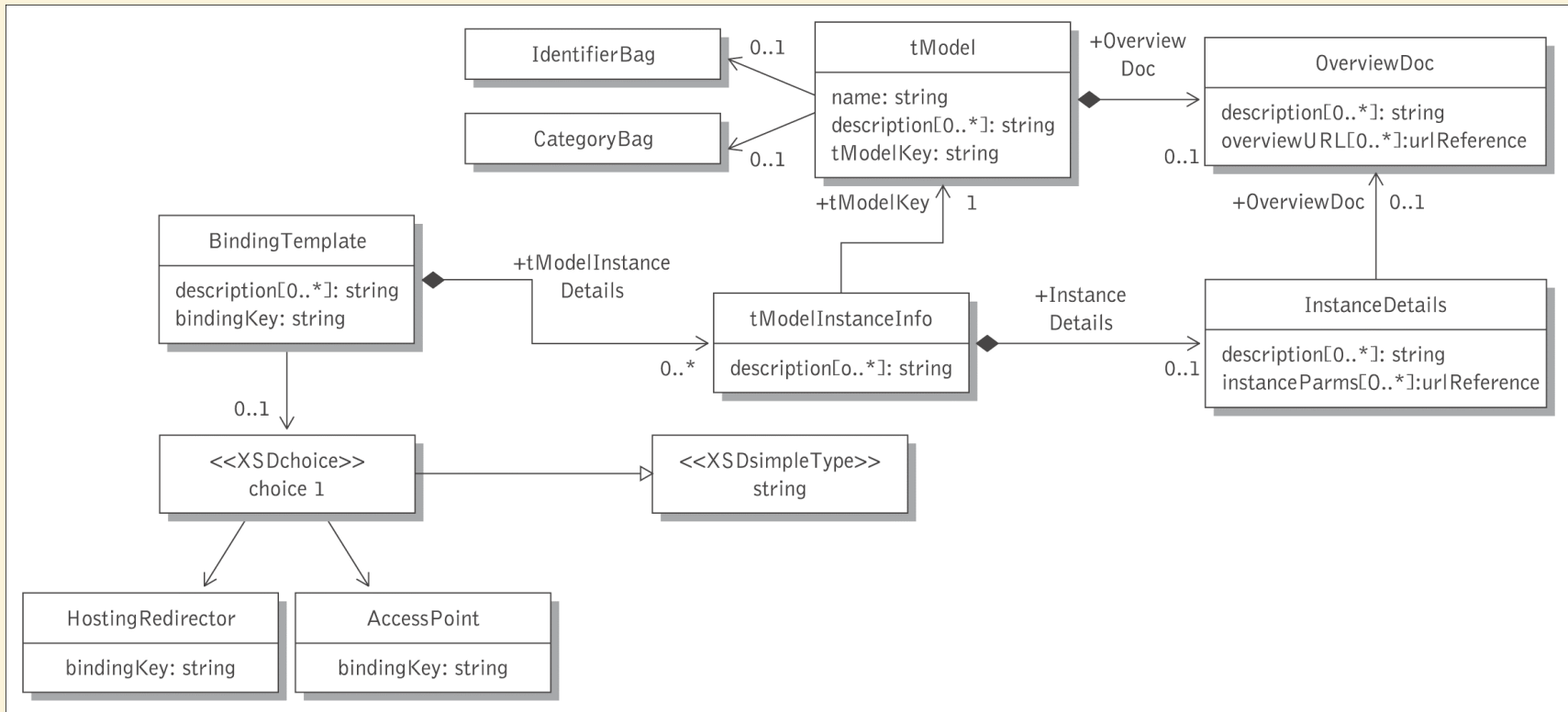
UDDI and WSDL
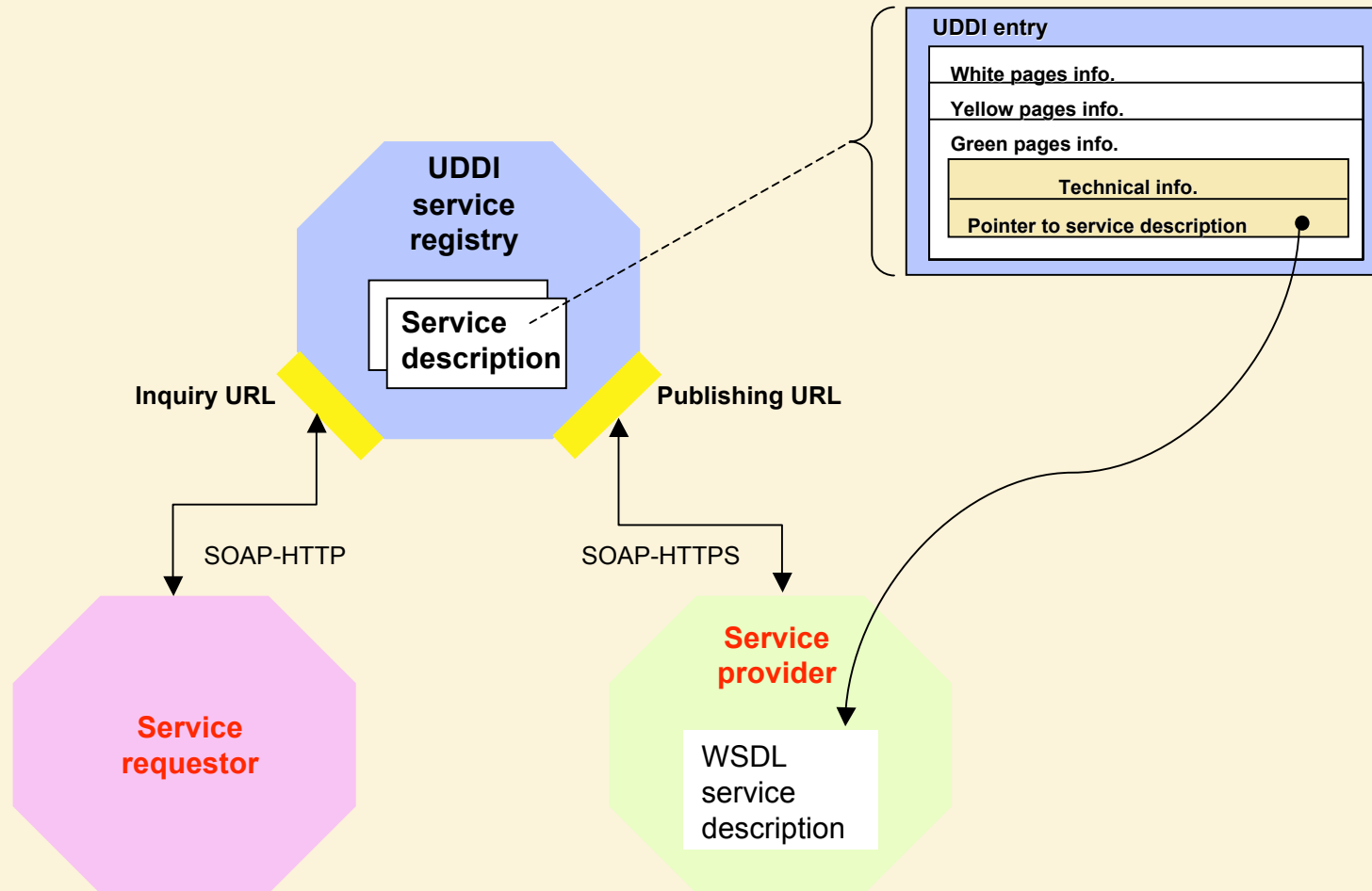
WSDL to UDDI Mapping Model

- The service desc. in WSDL documents is **complementary** to the information found in UDDI business/service entries. These two constructs work together naturally.

  – UDDI and WSDL distinguish clearly between interface and implementation.

  – By decoupling a WSDL specification and registering it in UDDI, we can populate UDDI with standard interfaces that have multiple implementations, providing a landscape of business applications that share interfaces.

Mapping WSDL to UDDI schemas
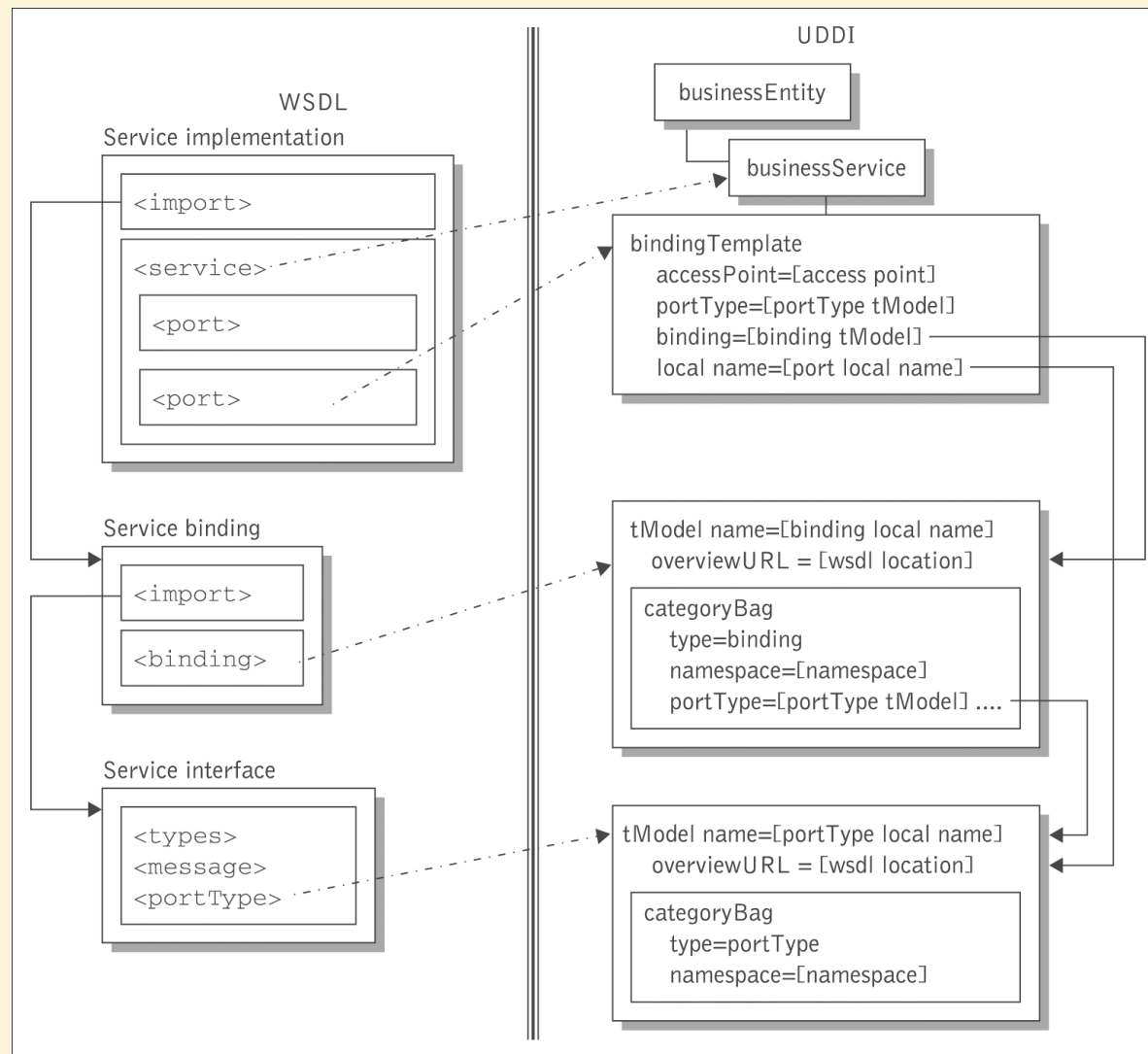
Example of UDDI <tModel> created from WSDL <portType> element

```
<tModel tModelKey="uuid:e8cf1163…" >
  <name>
    PurchaseOrderPortType
  </name>
  <overviewDoc>
    <overviewURL>
      http://supply.com:8080/PurchaseOrderService.wsdl
    <overviewURL>
  <overviewDoc>

  <categoryBag>

    <keyedReference
      tModelKey="uuid:d01987d1…"
      keyName="portType namespace"
      keyValue="http://supply.com/PurchaseService/wsdl" />

    <keyedReference
      tModelKey="uuid:6e090afa…"
      keyName="WSDL type"
      keyValue="portType" />

  </categoryBag>
</tModel>
```

The <tModel> name is the same as the WSDL name of the <portType>.

- Information in the <portType> about messages, operations, etc. is not duplicated in UDDI so the <portType tModel> must refer to the WSDL document in which the <portType> is defined.
- Development tools that generate a programming language interface from the <portType>, or validate requests against the definition of the <portType>, can retrieve the associated WSDL document.

Overview of WSDL to UDDI mapping

```
<import namespace="http//.."
    location="http://..">

<wsdl:service name="PurchaseOrderService">
    <wsdl:port  name="PurchaseOrderPort"
        binding="tns: PurchaseOrderSOAPBinding">
        <soapbind:address
          location="http://supply.com:8080/
          PurchaseOrderService"/>
    </wsdl:port>
</wsdl:service>
```

```
<businessService
 <name> Purchase Order Service </name>
 <bindingTemplate
  <accessPoint> http://supply.com:8080/PurchaseOrderService</accessPoint>
    <tModelInstanceInfo
      tModelKey="uuid:49662926-f4a…">
    </tModelInstanceInfo>
    <tModelInstanceInfo
      tModelKey="uuid:e8cf1163…">
    </tModelInstanceInfo>
 </bindingTemplate>
 <categoryBag>
 <keyedReference  keyName="service namespace"
   keyValue=""http://supply.com/PurchaseService/wsdl"/>
 <keyedReference  keyName="service local name"
 keyValue="PurchaseOrderService"/>
 </categoryBag>
</businessService>
```

```
<import namespace="http//.."
    location="http://..">

<wsdl:binding name="PurchaseOrderSOAPBinding"
  type="tns: PurchaseOrderPortType">
<soapbind:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http/"/>
 <wsdl:operation name="SendPurchase">
     <soapbind:operation  style="rpc"
        soapAction="http://supply.com/ …"/>
 <wsdl:input> <soapbind:body use="literal" </wsdl:input>
 <wsdl:output> <soapbind:body use="literal" </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
```

```
<tModel tModelKey="uuid:49662926-f4a…">
 <name> PurchaseOrderSOAPBinding </name>
 <overviewURL>
    http://supply.com:8080/PurchaseOrderService.wsdl
 <overviewURL>
 <categoryBag>
  <keyedReference
     tModelKey="uuid:d01987…."
     keyName="binding namespace"
     keyValue="http://supply.com/PurchaseService/wsdl"/>
  <keyedReference
     tModelKey="uuid:082b0851…"
     keyName="portType reference"
     keyValue="uuid:e8cf1163…"/>
 </categoryBag>
</tModel>
```

```
<definitions name="PurchaseOrderService"
 targetNamespace=http://supply.com/ PurchaseService/wsdl

<wsdl:portType name="PurchaseOrderPortType">
     <wsdl:operation name="SendPurchase">
          <wsdl:input message="tns:POMessage"/>
          <wsdl:output message="tns:InvMessage"/>
       </wsdl:operation>
    </wsdl:portType>
```

```
<tModel tModelKey="uuid:e8cf1163…">
 <name> PurchaseOrderPortType</name>
   <overviewURL>
    http://supply.com:8080/PurchaseOrderService.wsdl
   <overviewURL>
   <categoryBag>
    <keyedReference  tModelKey="uuid:d01987d1…"
       keyName="portType namespace"
       keyValue="http://supply.com/PurchaseService/wsdl"/>
   </categoryBag>
</tModel>
```

Summary of the UDDI data model

BusinessEntity
businessKey, name, contact, description, identifiers, categories

BusinessService
serviceKey, businessKey, name description, categories

BindingTemplate
bindingKey, serviceKey, description, categories, access point

WSDL document
External web service interface description (located at the service provider)

tModel
name, description, overview document, URL pointer to WSDL

Interaction with an UDDI registry

- UDDI provides application program interfaces (APIs) for access to a UDDI system:
  - **UDDI inquiry:** to locate and find details about entries in a UDDI registry. Supports a number of patterns (browsing, drill-down).
  - **UDDI publication:** to publish and modify information in a UDDI registry.
  - **UDDI security:** for access control to the UDDI registry (token based).
  - **UDDI subscription:** for clients to subscribe to changes of information in the UDDI registry.
  - **UDDI replication:** to perform replication of information across nodes in a UDDI registry.
  - **UDDI custody and ownership transfer:** to change the owner (publisher) of information.
- UDDI also provides a set of APIs for clients of a UDDI system:
  - **UDDI subscription listener:** the client side of the subscription API.
  - **UDDI value set:** used to validate the information provided to a UDDI registry.

UDDI enquiry API

- Requestors can obtain information from a UDDI registry using its enquiry interface.
  - Enquiries enable trading organizations to find businesses, services, or technical characteristics meeting certain criteria.
- *Browse functions* search the registry based on keywords and return summary lists with overview information (key, name, and description) about matching businesses or services.
- *Find qualifiers* are used to sort the results and to control the keyword matching.
  - To minimize the number of requests, find queries can be nested.
- *Drill-down functions* fetch the specific UDDI data structures about particular entries given their key, returned by the Browse functions.

Browse functions
find_business
find_relatedBusinesses
find_service
find_binding
find_tModel

Drill-down functions
get_businessDetail
get_operationalInfo
get_serviceDetail
get_bindingDetail
get_tModelDetail

UDDI Version 3.0 Specification, 19 July 2002

UDDI publishing and security API

- The publishing interface can be used by clients to store and update information contained in a UDDI registry.

- The registry performs *access control* for all publishing functions: information about the entries can only be edited by the owner.

- **Category information** and keyed references associated with the entries are validated before accepting new information into the registry.

- **Save operations** allow a client to add or update information in the UDDI.
  - Each of the primary UDDI data structures has a corresponding save operation.
  - Deletion functions are used to remove entries identified by their key from the registry. Removing a business will remove all services associated with it.

- The same publishing functions are used both to add new information or replace existing information, depending on whether a valid key is passed or not.

Security session management functions
get_authToken, discard_authToken

Publishing functions	Deletion functions
save_business	*delete_business*
save_service	*delete_service*
save_binding	*delete_binding*
save_tModel	*delete_tModel*

Querying the UDDI

```
<find_business generic="2.0" xmlns="urn:uddi-org:api_v2">
   <name xml: lang="en"> Automotive Equipment Manufacturing Inc. </name>
   <name xml: lang="en"> Manufacturing Goods Co. </name>
</find_business>
```

Query 1 Finding business entities by name.

```
<find_business generic="2.0" xmlns="urn:uddi-org:api_v2">

<categoryBag>
   <!—North American Industry Classification System (NAICS) -->
   <keyedReference
           keyName="Automotive parts distribution"
           keyValue="…"
           tModelKey="…"/>
     ……
</categoryBag>
</find_business>
```

Query 2 Finding <businessEntity> entries by category.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
   <name> PurchaseOrderPortType </name>
   <categoryBag>
      <keyedReference
         tModelKey="uuid:d01987d1…"
         keyName="portType namespace"
         keyValue="http://supply.com/PurchaseService/wsdl" />
      <keyedReference
         tModelKey="uuid:6e090afa…"
         keyName="WSDL type"
         keyValue="portType" />
   </categoryBag>
</tModel>
```

Query 3 Finding the <tModel> for <portType> name.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
   <categoryBag>
      <keyedReference
         tModelKey="uuid:6e090afa…"
         keyName="WSDL type"
         keyValue="binding" />
      <keyedReference
         tModelKey="uuid:082b0851… "
         keyName="portType reference"
         keyValue="uuid:e8cf1…" />
   </categoryBag>
</tModel>
```

Query 4 Find all <binding tModel> for PurchaseOrderPortType.

Business information provider roles

- **Registry operators:** organizations (operator nodes) that host and operate the UDDI business registry (UBR).
  - These manage and maintain the directory information, cater for replication of business info. and other directory-related functions.
  - The registry works on a "register once, published everywhere" principle.
- **Standard bodies and industry consortia**: these publish descriptions in the form of service type definitions (<tModel>s).
  - These <tModel>s do not contain the actual service definitions, instead they have a URL that points to the location where the service descriptions (usually in WSDL) are stored.
- **Service providers:** commonly implement Web services conforming to service type definitions supported by UDDI.
  - They publish information about their business and services in the UDDI.
  - The published data also contains the end point of Web services.

UDDI deployment possibilities

- **e-marketplace UDDI:** an e-marketplace, a standards body, or a consortium of organizations that participate and compete in the industry can host this private UDDI node.
  - The e-marketplace could run a local version of a UDDI registry with its data shielded from the global UDDI registry.
  - The entries in this private UDDI relate to a particular industry or narrow range of related industries.
- **Business partner UDDI registry:** a private UDDI node hosted behind one of the business partner's firewall and only trusted or vetted partners can access the registry.
  - It also contains Web service description meta-data published by trusted business parties (organizations with which the hosting organization has formal agreements/relationships).
- **Portal UDDI:** this type of deployment is on an enterprise's firewall and is a private UDDI node that contains only meta-data related to the enterprise's Web services.
  - External users are allowed to invoke find operations on the registry; however, a publish operation would be restricted to services internal to the portal.
  - It gives a company ultimate control over how the meta-data describing its Web services is used.
- **Internal UDDI:** this allows applications in different departments of the organization to publish and find services, and is useful for large organizations.
  - The major distinction of this UDDI variant is the potential for a common administrative domain that can dictate standards (for example a fixed set of <tModels> can be used).
  - This type of UDDI deployments is called EAI UDDI, as it allows corporations to deploy and advertise Intranet Web services.