Multi-agent and Semantic Web Systems: Web Services: Part 1

Fiona McNeill

School of Informatics

7th March 2013

Antecedents

B2B

Previous attempts at distributed computing (CORBA, Distributed Smalltalk, Java RMI) have yielded systems where the coupling between various components in a system is too tight to be effective for low-overhead, ubiquitous B2B e-business over the Internet. These approaches require too much agreement and shared context among business systems from different organizations to be reliable for open, low-overhead B2B e-business.

`http://www-128.ibm.com/developerworks/webservices/library/w-ovr/`

Web Service Architecture

- Tightly coupled, monolithic systems are brittle:
  - changing the output of one subsystem can cause the whole system to break;
  - software collaboration may unintentionally rely on side effects of a specific implementation.
- Web Service architecture is designed to be loosely coupled;
- applications use service discovery to dynamically bind components to concrete network-available services.

Applications in WS Architecture

Application Design

- Describe capabilities of network services needed to perform a function.
- Describe the 'orchestration' of these collaborators.

Application Execution

- Translate collaborator requirements into query to discovery agent.
- Locate service with right capabilities.
- Orchestrate message-passing to invoke services.

- Desired goal: "Just-in-time" integration of applications.

Concepts and Terminology

- Following terminology about Web Services derived from W3C *Web Services Architecture* (WS-Arch: W3C WG Note, 2004–2-11).

- http://www.w3.org/TR/ws-arch/

Web Services

A Web Service (WS) is:

- a software system

Web Services

A Web Service (WS) is:

- a software system
- designed to support interoperable machine-to-machine interaction over a network;

Web Services

A Web Service (WS) is:

- a software system
- designed to support interoperable machine-to-machine interaction over a network;
- its public interfaces are described in XML (e.g., WSDL);

Web Services

A Web Service (WS) is:

- a software system
- designed to support interoperable machine-to-machine interaction over a network;
- its public interfaces are described in XML (e.g., WSDL);
- other systems can interact with the WS as prescribed by the interface description,

Web Services

A Web Service (WS) is:

- a software system
- designed to support interoperable machine-to-machine interaction over a network;
- its public interfaces are described in XML (e.g., WSDL);
- other systems can interact with the WS as prescribed by the interface description,
- using XML-based (e.g., SOAP) messages.

Web Services

A Web Service (WS) is:

- a software system
- designed to support interoperable machine-to-machine interaction over a network;
- its public interfaces are described in XML (e.g., WSDL);
- other systems can interact with the WS as prescribed by the interface description,
- using XML-based (e.g., SOAP) messages.

WS Standards

WSDL (Web Service Description Language) and SOAP: W3C Recommendations; used widely but not universally.

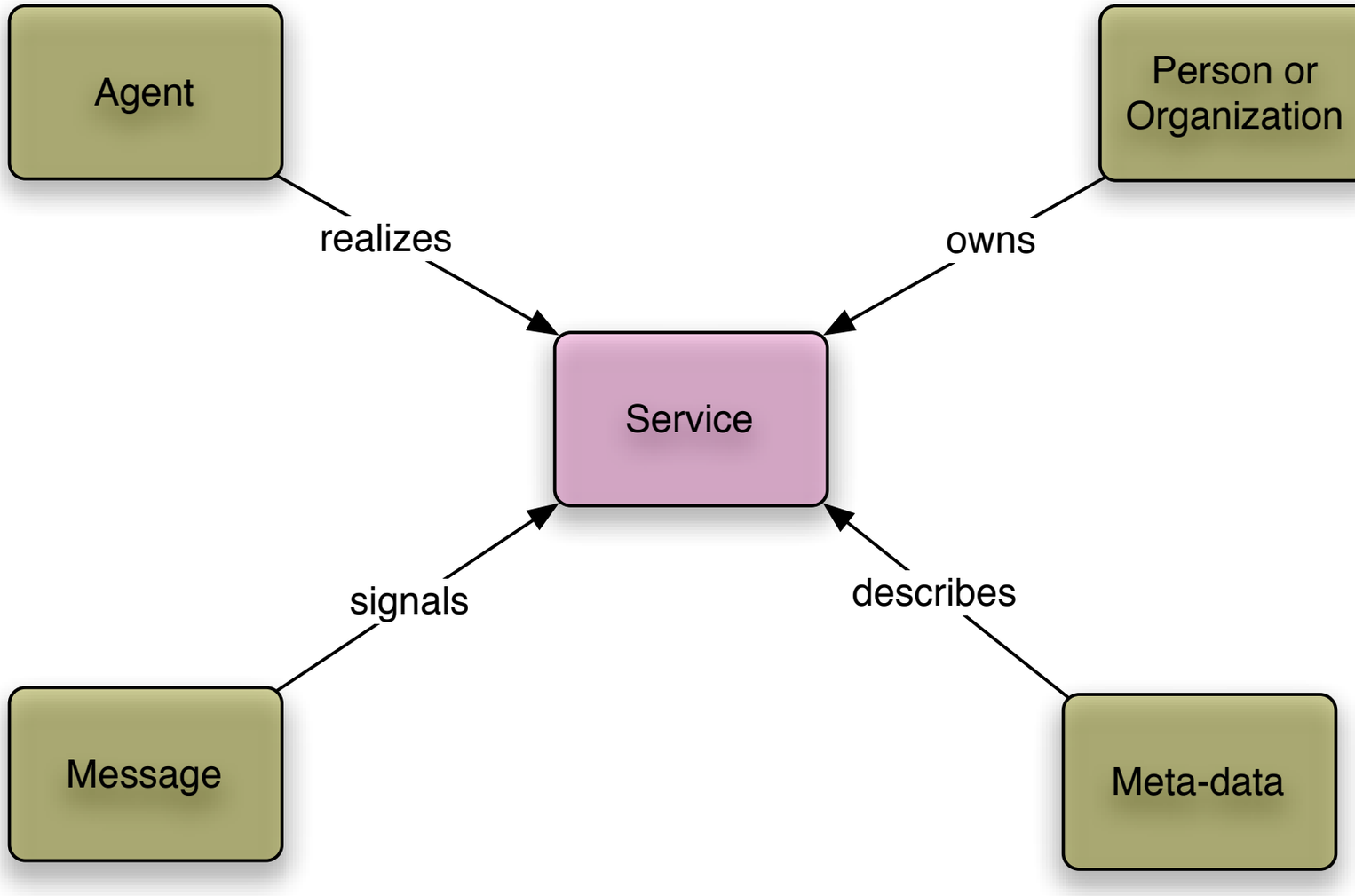
Agents, Entities and Services

- WS is intended to be an abstract notion.

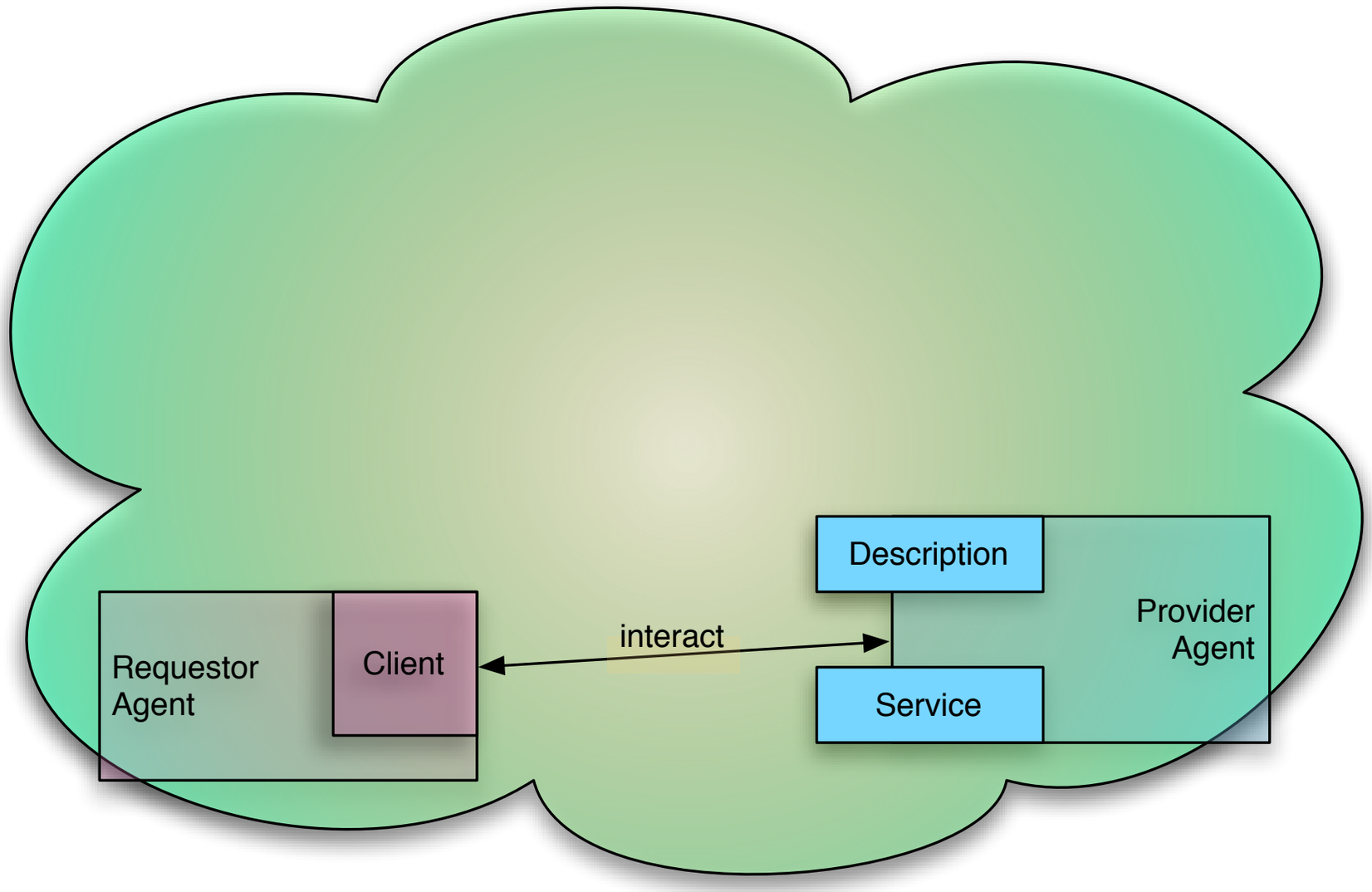
- Must be realised by a concrete piece of software—called an agent by WS-Arch.
  - Agent can send and receive messages.
  - Service is a resource defined by its functionality.
  - Service can stay the same even though agent (i.e., implementation) is changed.

- Entity is individual or organisation that requests or provides a service.

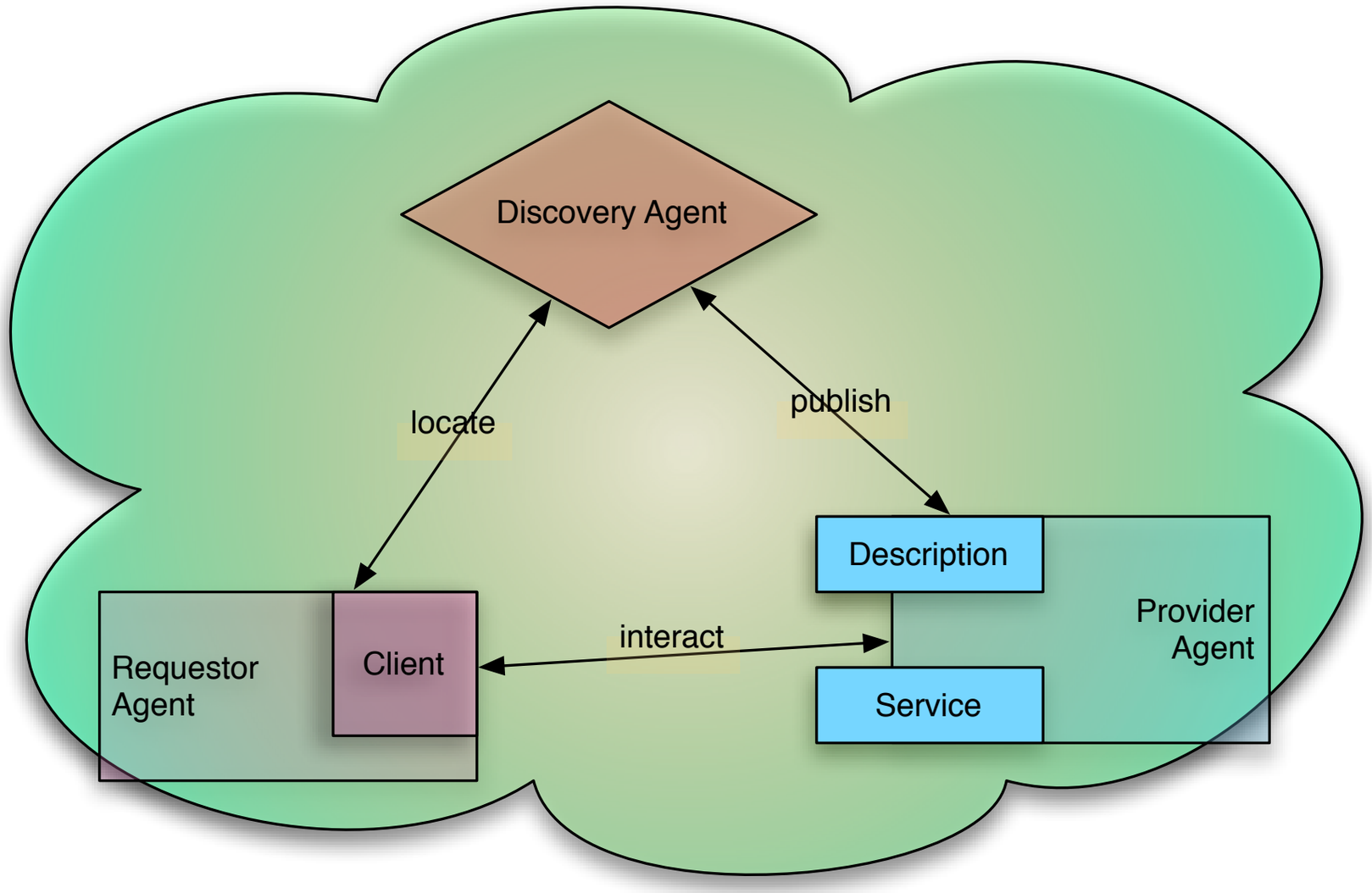
Service Oriented Architecture (WS-Arch)



Service Oriented Architecture: interact



Service Oriented Architecture: interact/publish/locate



WS Use Case

http://www.w3.org/2002/06/ws-example

- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.

WS Use Case

http://www.w3.org/2002/06/ws-example

- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.
- Organisations offer WS that allow user to query services and make reservations.

WS Use Case

http://www.w3.org/2002/06/ws-example

- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.
- Organisations offer WS that allow user to query services and make reservations.
- Credit card agency provides WS to guarantee payment by consumer.

WS Use Case

http://www.w3.org/2002/06/ws-example

- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.
- Organisations offer WS that allow user to query services and make reservations.
- Credit card agency provides WS to guarantee payment by consumer.
- Travel Agent doesn't have/need *a priori* agreements with service providers.

WS Use Case

http://www.w3.org/2002/06/ws-example

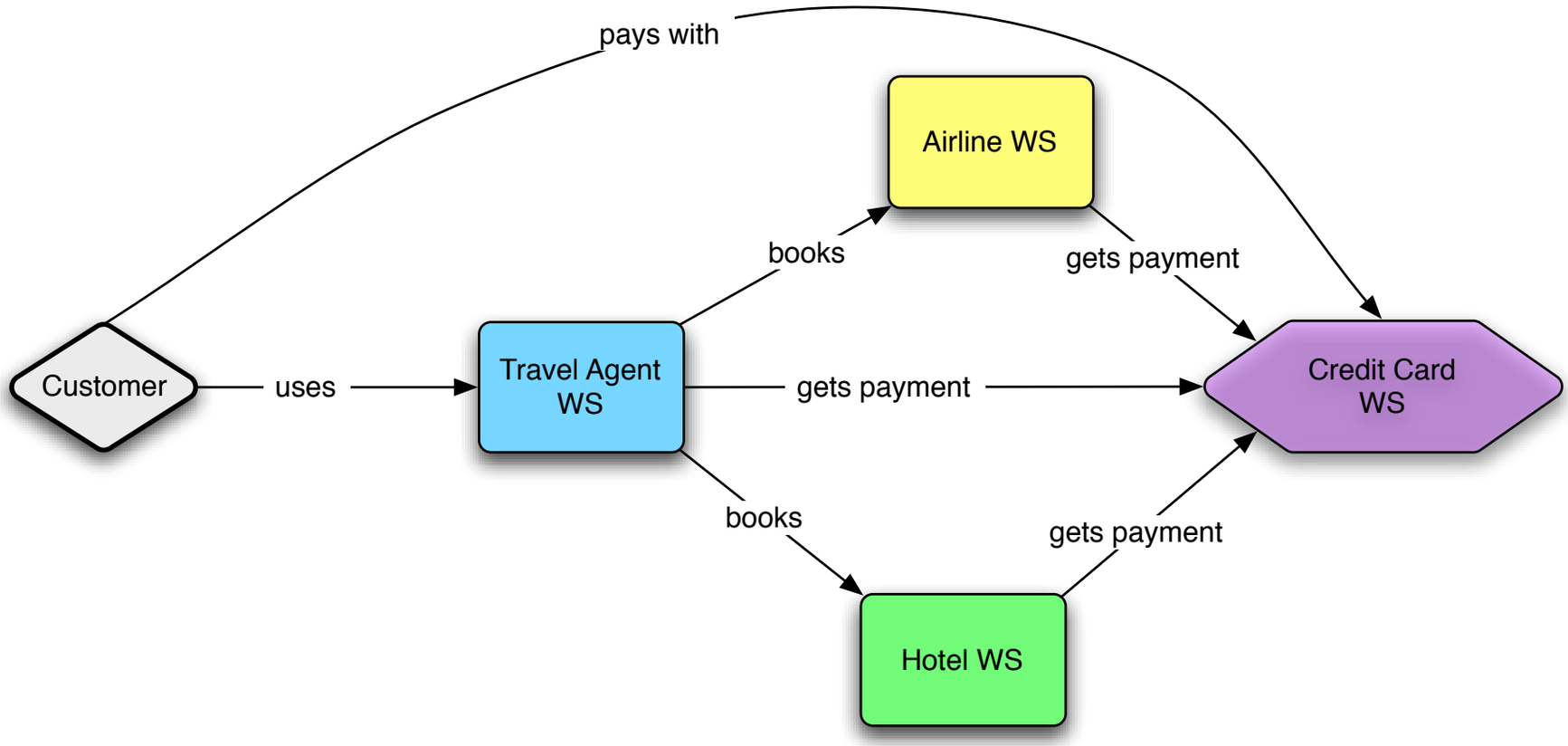
- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.
- Organisations offer WS that allow user to query services and make reservations.
- Credit card agency provides WS to guarantee payment by consumer.
- Travel Agent doesn't have/need *a priori* agreements with service providers.
- Only the vacationer is human; all other services are software agents.

WS Use Case

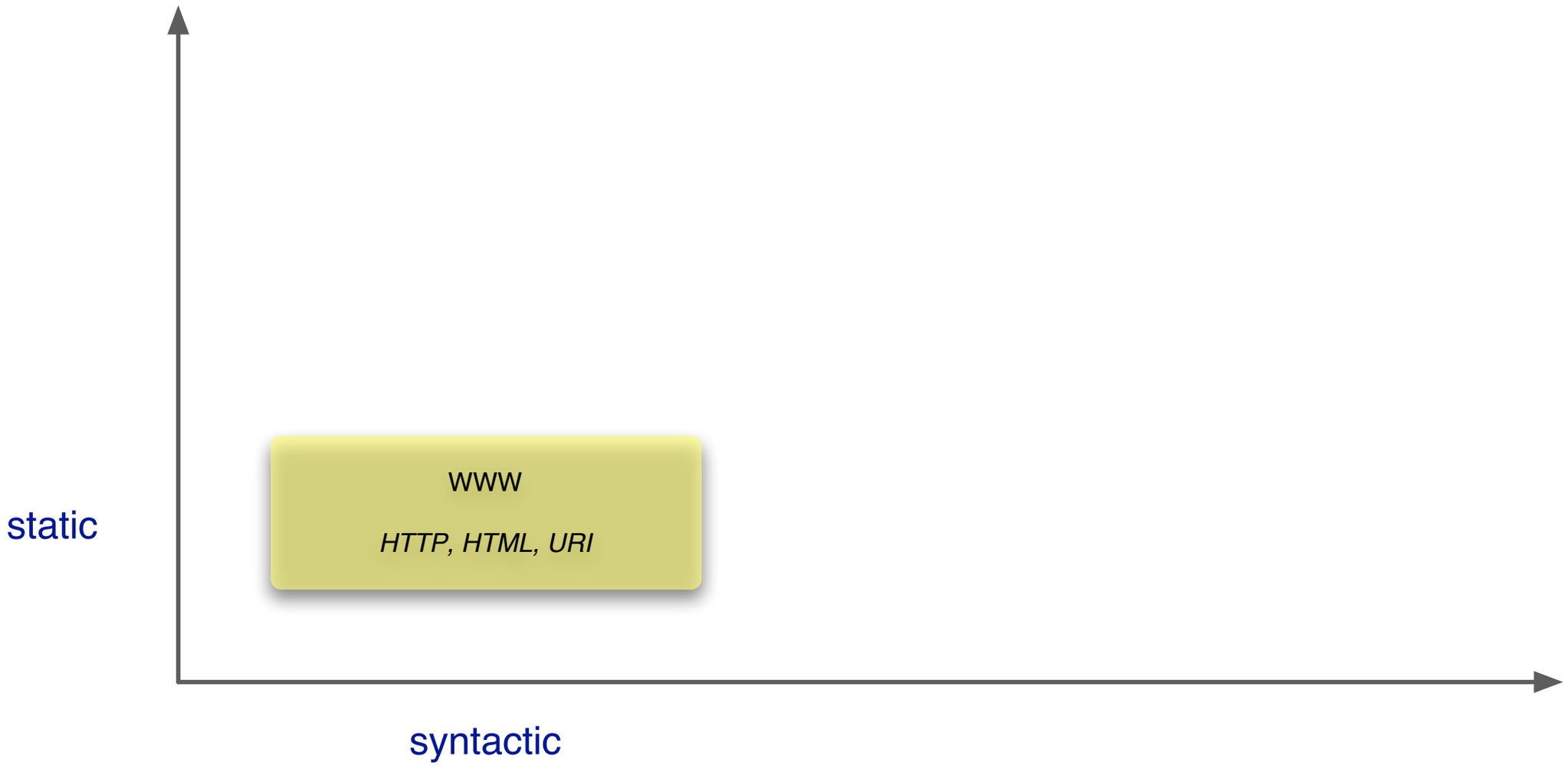
http://www.w3.org/2002/06/ws-example

- Travel Agent offers customers ability to book complete vacation package, e.g., plane tickets, hotel, car rental at destination, excursions, etc.
- Organisations offer WS that allow user to query services and make reservations.
- Credit card agency provides WS to guarantee payment by consumer.
- Travel Agent doesn't have/need *a priori* agreements with service providers.
- Only the vacationer is human; all other services are software agents.
- Assumes that agents share common concepts about Flight, EconomyClass, etc.

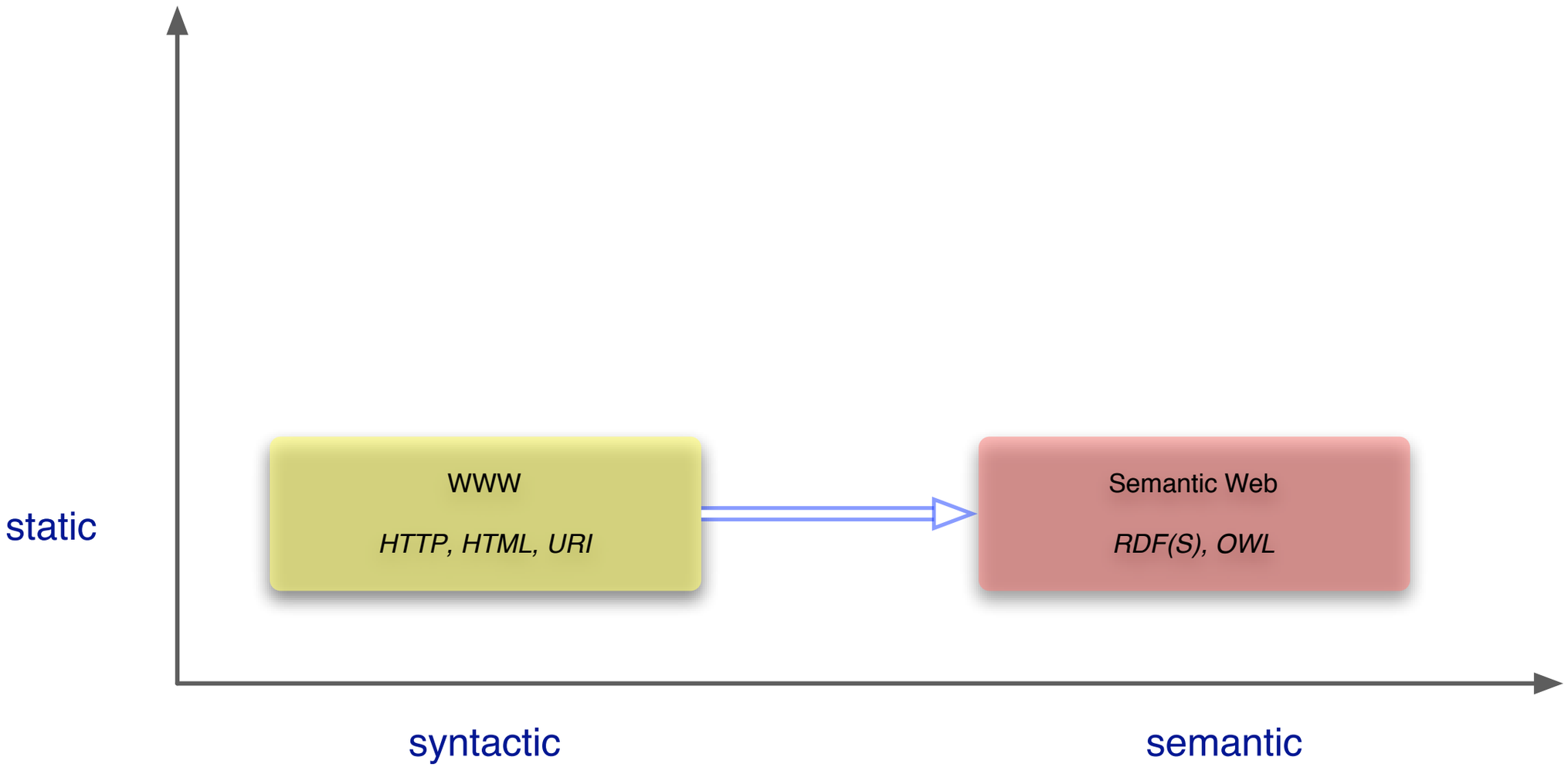
Travel Agent Use Case



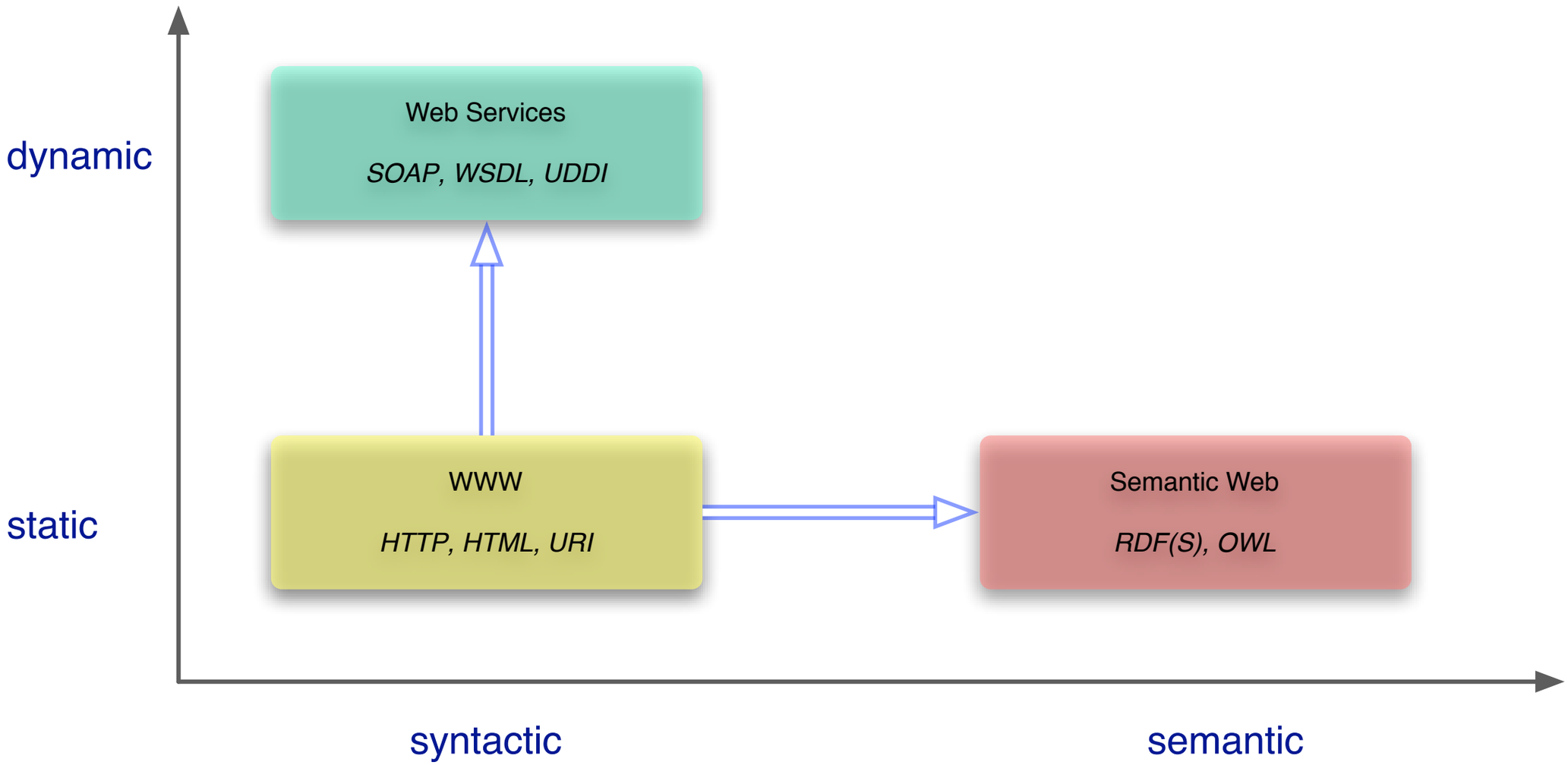
Evolution of the WWW



Evolution of the WWW



Evolution of the WWW



The Appeal of Web Services

- A means of building distributed system across the internet.

- Virtualisation: independent of programming language, OS, development environment.

- Based on well-understood underlying transport mechanisms (e.g., HTTP).

- Components can be developed and upgraded independently.

- Fairly decentralised (though issues about discovery, composition).

- Probably not appropriate where a high level of fine-grained interaction is required.

Perspectives

- A variety of different views on what's happening (not mutually exclusive):

Perspectives

- A variety of different views on what's happening (not mutually exclusive):
  - remote procedure call (RPC)

Perspectives

- A variety of different views on what's happening (not mutually exclusive):
  - remote procedure call (RPC)
  - business process within a workflow

Perspectives

- A variety of different views on what's happening (not mutually exclusive):
  - remote procedure call (RPC)
  - business process within a workflow
  - dialogue in multi-agent system

RPC Concepts

- RPC is a protocol to allow agent on one host to cause execution of code on remote host.

- Uses client-server model of distributed computation:
  - Client sends message to server
    - [execute] procedure *P* with arguments $a1, \ldots, an$
  - Server executes *P*, and sends message back to client
    - result [of executing $P(a1, \ldots, an)$]

Protocols and Endpoints

Protocol

Convention that govern syntax, semantics, and synchronization of communication between two computing 'endpoints'. Enables/controls connection, communication, data transfer.

Protocols and Endpoints

Protocol

Convention that govern syntax, semantics, and synchronization of communication between two computing 'endpoints'. Enables/controls connection, communication, data transfer.

Endpoint

Endpoint is "an entity, processor, or resource to which …messages can be addressed".

Protocols and Endpoints

Protocol

Convention that govern syntax, semantics, and synchronization of communication between two computing 'endpoints'. Enables/controls connection, communication, data transfer.

Endpoint

Endpoint is "an entity, processor, or resource to which …messages can be addressed".

Endpoint Reference

Conveys the information needed to address an endpoint.
Interactions may create new service instances, hence a need to dynamically create new endpoint references.
(cf. `http://www.w3.org/TR/ws-addr-core`)

RPC Example

- Assume Hotel Splendide is making room reservations available as a WS.

RPC Example

- Assume Hotel Splendide is making room reservations available as a WS.

- It should expose a function `checkAvailability` which takes the check-in and check-out dates and room type as input parameters, and returns the price in US$ as a floating point number.

RPC Example

- Assume Hotel Splendide is making room reservations available as a WS.

- It should expose a function `checkAvailability` which takes the check-in and check-out dates and room type as input parameters, and returns the price in US$ as a floating point number.

Example Function

```
public float checkAvailability(Date checkinDate,
                               Date checkoutDate,
                               String roomType) {
    if (roomAvailable) {
       return roomRateInUSD;
    } else {
       return 0.0;
    }
}
```

WS Metadata

Two kinds of metadata about services:

Non-operational

- service category (e.g., hotel room booking)
- informal description
- information about provider entity (name, contact details)

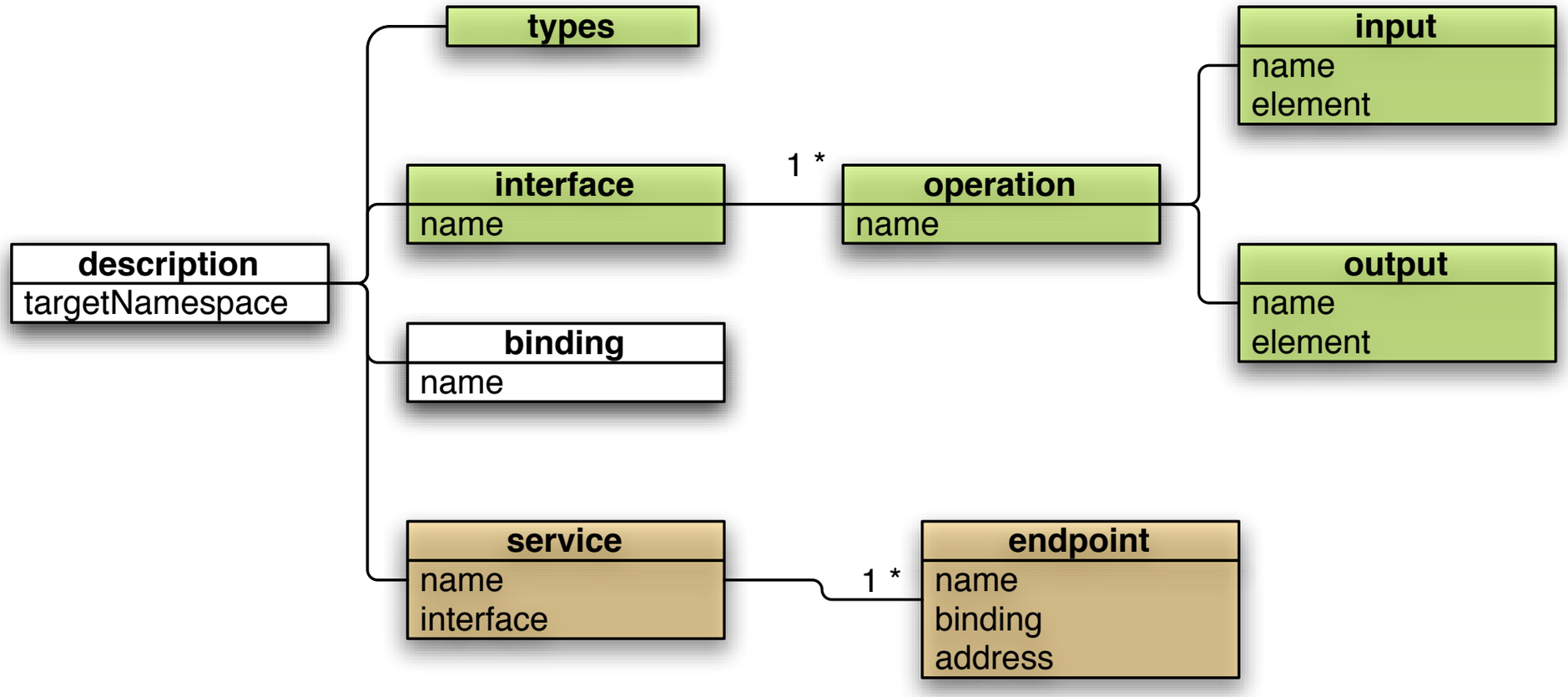
Operational

- service interface
- communication protocol
- service endpoint

- Operational metadata is standardly expressed using Web Service Description Language (WSDL)

- http://www.w3.org/TR/wsdl20-primer

WSDL 2.0 Structure



WSDL Interface

Example of WSDL Interface

```
<interface name = "reservationInterface" >
  <operation name="checkAvailability"">
    <input messageLabel="In"
           element="CheckAvailability"/>
    <output messageLabel="Out"
           element="CheckAvailabilityResponse"/>
  </operation>
</interface>
```

WSDL Interface

Example of WSDL Interface

```
<interface name = "reservationInterface" >
  <operation name="checkAvailability"">
    <input messageLabel="In"
           element="CheckAvailability"/>
    <output messageLabel="Out"
           element="CheckAvailabilityResponse"/>
  </operation>
</interface>
```

WSDL Interface

Example of WSDL Interface

```
<interface name = "reservationInterface" >
  <operation name="checkAvailability"">
    <input messageLabel="In"
           element="CheckAvailability"/>
    <output messageLabel="Out"
           element="CheckAvailabilityResponse"/>
  </operation>
</interface>
```

- element=”CheckAvailability” specifies the message type.

WSDL Interface

Example of WSDL Interface

```
<interface name = "reservationInterface" >
  <operation name="checkAvailability"">
    <input messageLabel="In"
           element="CheckAvailability"/>
    <output messageLabel="Out"
           element="CheckAvailabilityResponse"/>
  </operation>
</interface>
```

- element=”CheckAvailability” specifies the message type.
- Where does this get defined?

WSDL Interface

Example of WSDL Interface

```
<interface name = "reservationInterface" >
  <operation name="checkAvailability"">
    <input messageLabel="In"
           element="CheckAvailability"/>
    <output messageLabel="Out"
           element="CheckAvailabilityResponse"/>
  </operation>
</interface>
```

- element=”CheckAvailability” specifies the message type.
- Where does this get defined?
- In the types section of the WSDL document.

WSDL Types

- Use XML Schema to define types; should be supported by all WSDL 2.0 processors.

WSDL Types

- Use XML Schema to define types; should be supported by all WSDL 2.0 processors.

- But WSDL 2.0 allows the use of other schema definition languages.

WSDL Types

- Use XML Schema to define types; should be supported by all WSDL 2.0 processors.

- But WSDL 2.0 allows the use of other schema definition languages.

Example of WSDL Type Definition

```
<types>
 <xs:element name="checkAvailability" type="tCheckAvailability"/>
 <xs:complexType name="tCheckAvailability">
   <xs:sequence>
     <xs:element  name="checkInDate" type="xs:date"/>
     <xs:element  name="checkOutDate" type="xs:date"/>
     <xs:element  name="roomType" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
 <xs:element name="checkAvailabilityResponse" type="xs:double"/>
</types>
```

Service Discovery

- The 'standard' solution uses UDDI Repositories (supported by OASIS)
- UDDI = Universal Description, Discovery, and Integration
- Original vision was a universal yellow pages for e-Business services.
- Services are categorised using a flattish taxonomy; search is by category and keyword.
- But take-up has been low, and focus has moved to supporting private registries.

UDDI.org White Paper: The Evolution of UDDI

... most of today's web service applications are not intended for public use, but rather inside organizations or among existing, trusted business partners.

Composition

- Recall just-in-time integration of applications.

- Automatic composition of service-based applications is more vision than reality.

- Some tool support for manual or semi-manual composition.

- Composition raises issues about discovery and description.

- Next slides: manual composition using a scientific workflow tool.

myGrid, I (http://www.mygrid.org.uk/)

- Large scale, multi-site project in UK e-Science framework
- Concerned with building Grid oriented middleware for molecular biology research;
- *in silico* discovery by combining results and data from local and remote sources.
- Started in 2001 prior to establishment of BPEL, and developed own tools for service composition:
  - Taverna 'workflow' workbench,
  - Scufl language (composition operators).
  - Freefluo enactment engine.
- Specifically designed for use by biologist and bio-informatics users.

myGrid, 2

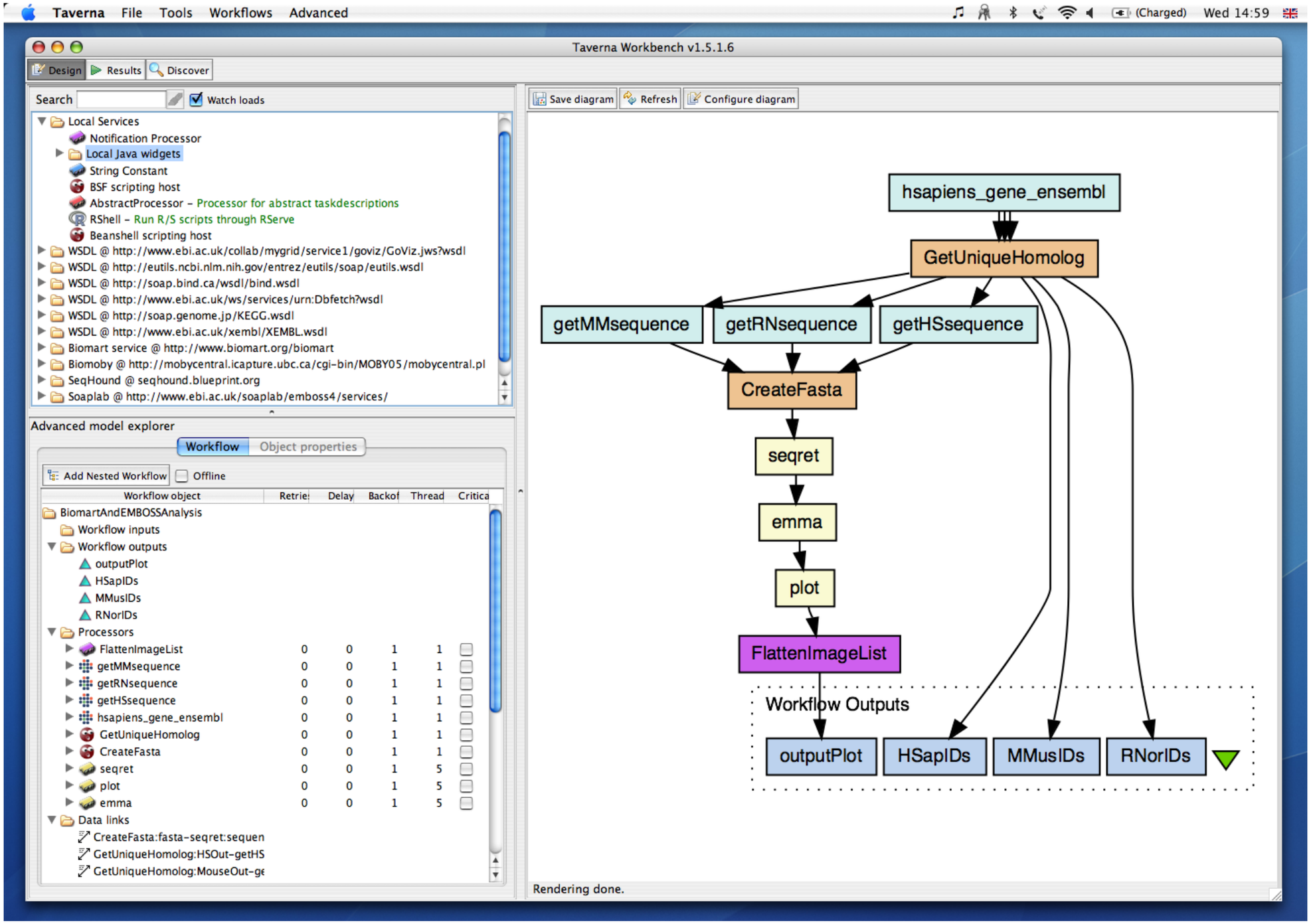
- Intended to provide uniform access to wide variety (> 1000) of services:
  - sequence comparison, protein databases, protein visualization tools, model simulations, etc.

- These are increasingly available as Web Services.

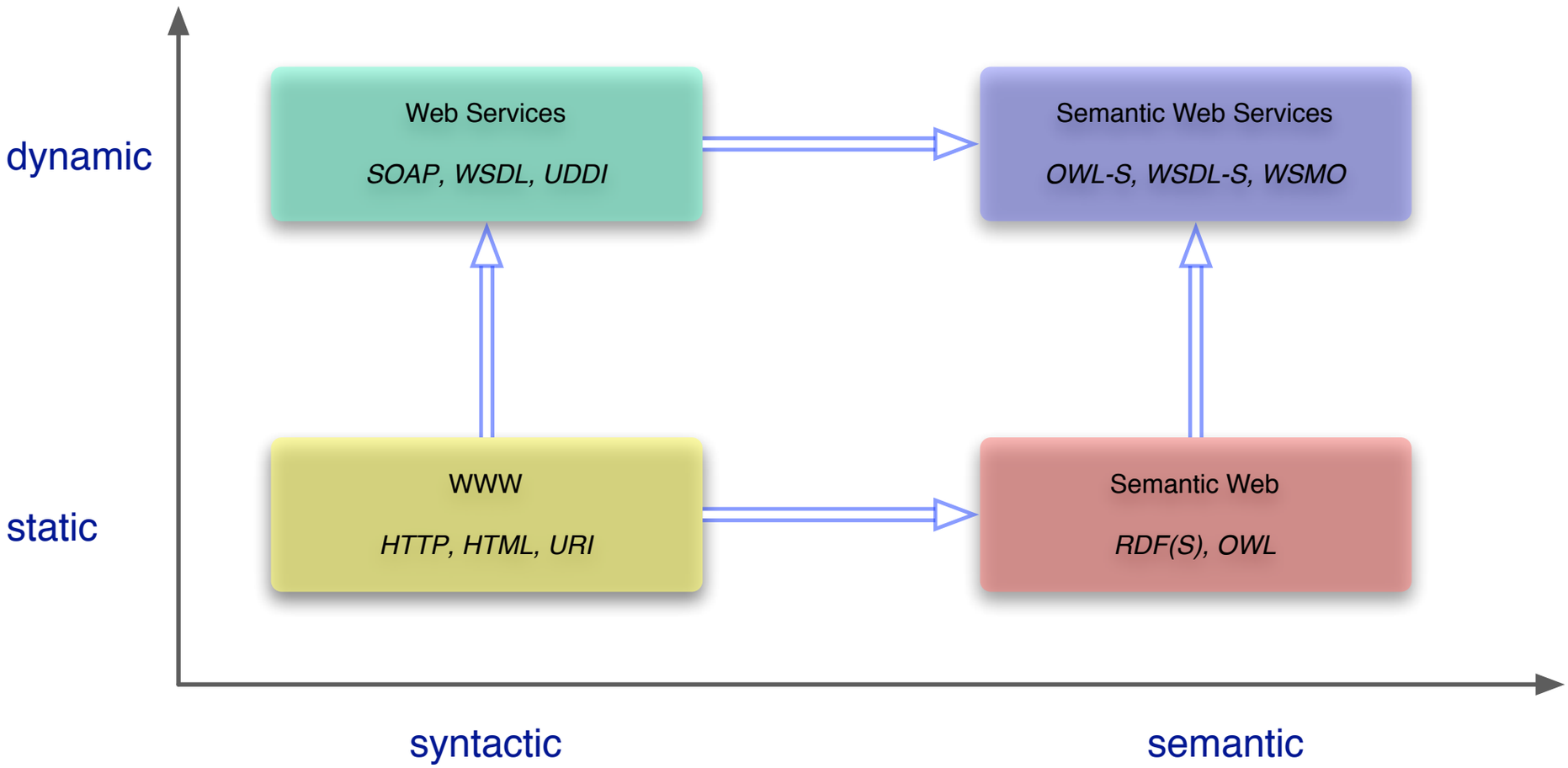
- Workflows need to be easy to create for one-off experiments, but also available for re-use, adaptation, and incorporation in other workflows.

- Tries to be non-prescriptive about data formats.

Taverna Workflow



Evolution of the WWW



Key idea: use richer RDFS / OWL classes instead of XML Schema types.

Conclusion, 1

- SOA can be seen as evolution of Object Oriented approach.

- Web Services are 'big business': lots of commercial involvement, lots of standardisation activity.

- But little deployment to date of SOA across the Internet. WS are primarily used within organisations:
  - Commercial organisations (maybe with trusted partners)
  - Virtual organisations for Grid computing and e-Science / e-Research.

- Other WS tend to be 'one-shot'; cf. Amazon, Google, etc.

Conclusion, 2

- WSDL has been promoted as standard for describing services:
  - `interface` and associated `operations` give an abstract specification of the service;
  - `binding` and `service endpoint` show how to invoke the concrete service.
- WSDL adopts an RPC view of service, in terms of input and output types of the constituent operations.

DIY

- Many WS are not WSDL/SOAP based.
- You've already accessed Last.fm as a WS.
- Even with SOAP-based services, there are easy-to-use client libraries.
- Writing WSDL interfaces and publishing WS is harder.

Reading

- Read Chapter 8 of Passin (but talks about WSDL 1.0 — some syntactic differences with WSDL 2.0).

- Online tutorial: `http://www.w3schools.com/WSDL/`

- `http://www.xmethods.net/` lists some publicly available WS.