Semantic Web Services

Christoph Bussler, Dieter Fensel,
Terry Payne, and Katia Sycara

Oracle Corporation, VU Amsterdam,
University of Southampton, Carnegie-Mellon

Contents

Introduction

- Web Services: The Vision & Truth
- WSMF
- Mediation & Matchmaking
- DAML-S
  - Advertising
  - Service Description and Invocation

Discussion and Wrap Up

Semantic Web Services

Introduction

Web Services:
The Vision and Truth

1. Vision & Truth

- Web Services will transform the web from a collection of information into a distributed device of computation.
- In order to employ their full potential, appropriate description means for web services need to be developed.

  For this purpose we developed a full-fledged *Web Service Modeling Framework (WSMF)* that provides the appropriate conceptual model for developing and describing web services and their composition.

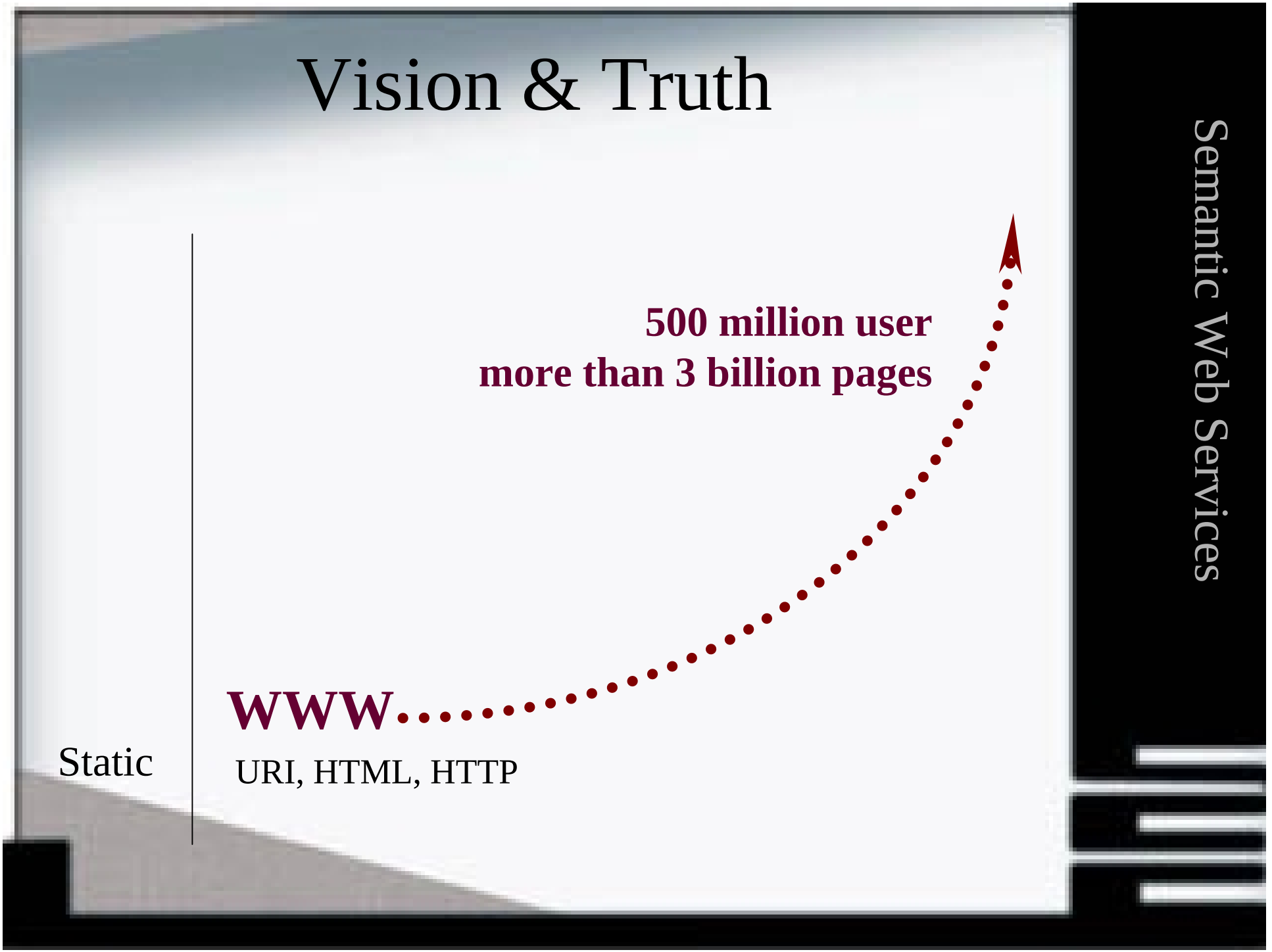
  For this purpose we developed a formal framework for describing web services called *DAML-S* that applies Semantic Web technology for mechanizing service selection and composition.

Vision & Truth

- The philosophy of WSMF is based on the following principle:
  - *maximal de-coupling* complemented by
  - *scalable mediation service*.
- The philosophy of DAML-S is to enrich web service description by machine-processability.

These are pre-requisites for scalable and cost-effective use of web service technology for eWork and eCommerce.

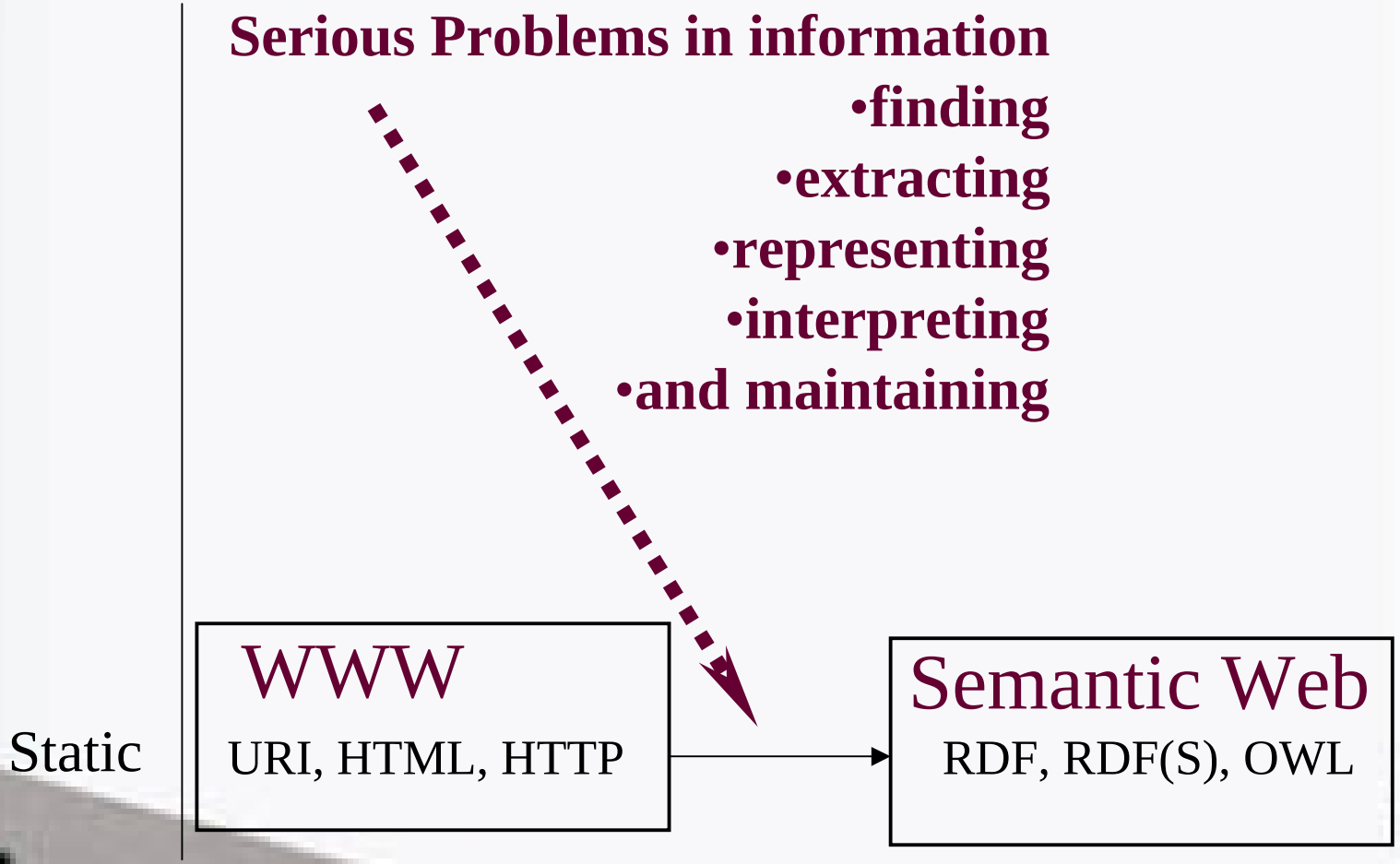
Vision & Truth



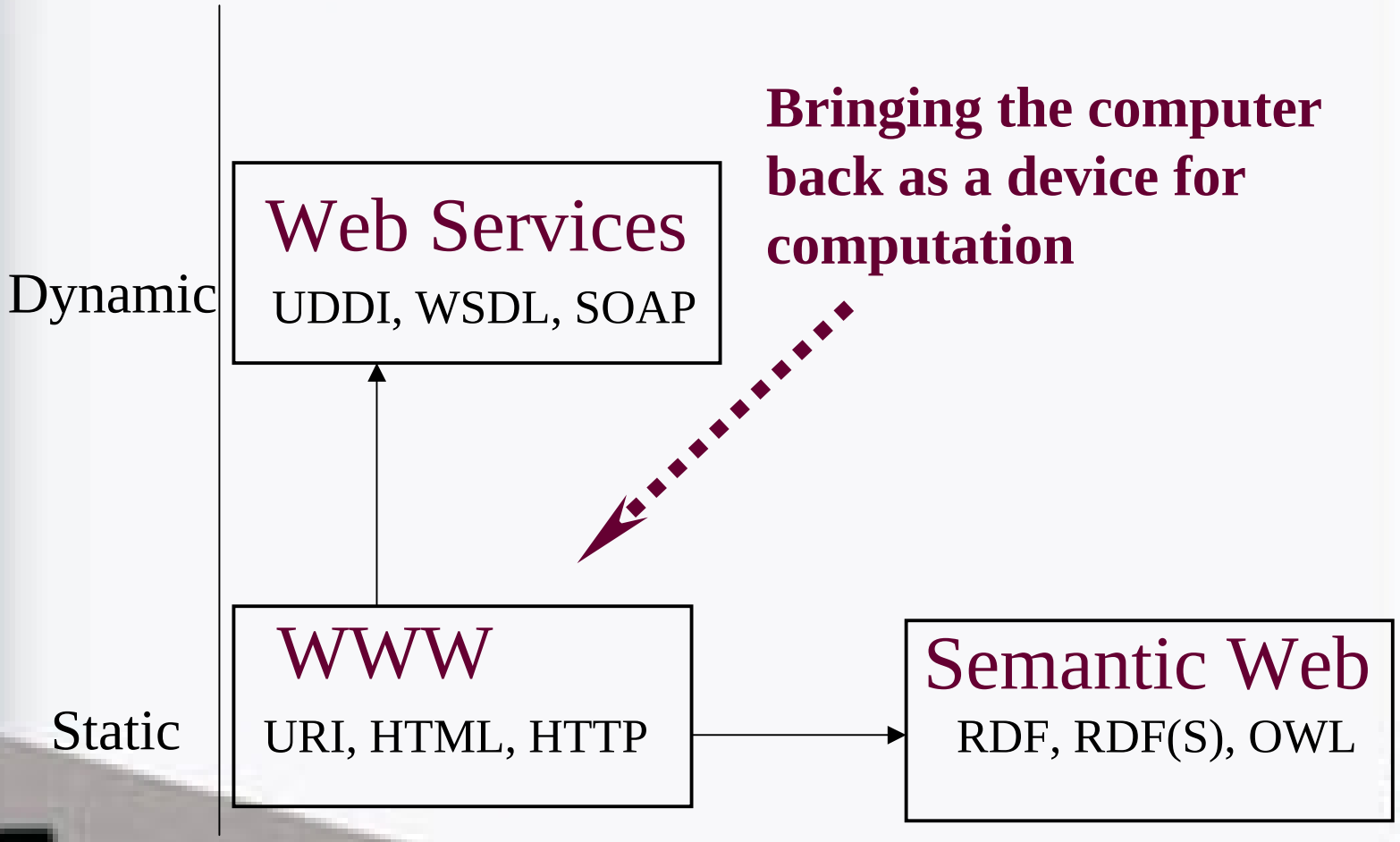
Vision & Truth

Serious Problems in information

- **finding**
- **extracting**
- **representing**
- **interpreting**
- **and maintaining**

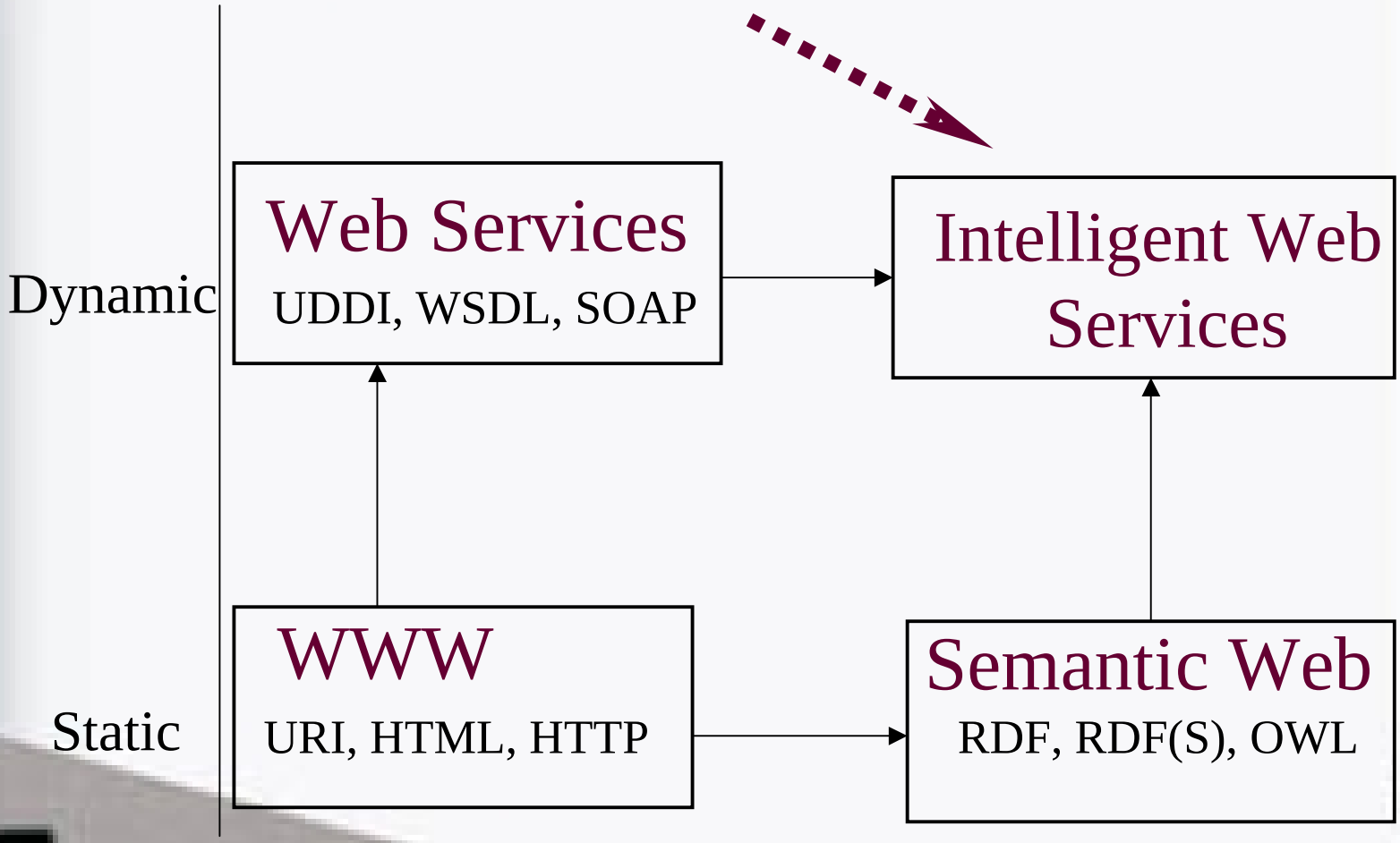


Vision & Truth



Vision & Truth

Bringing the web to its full potential



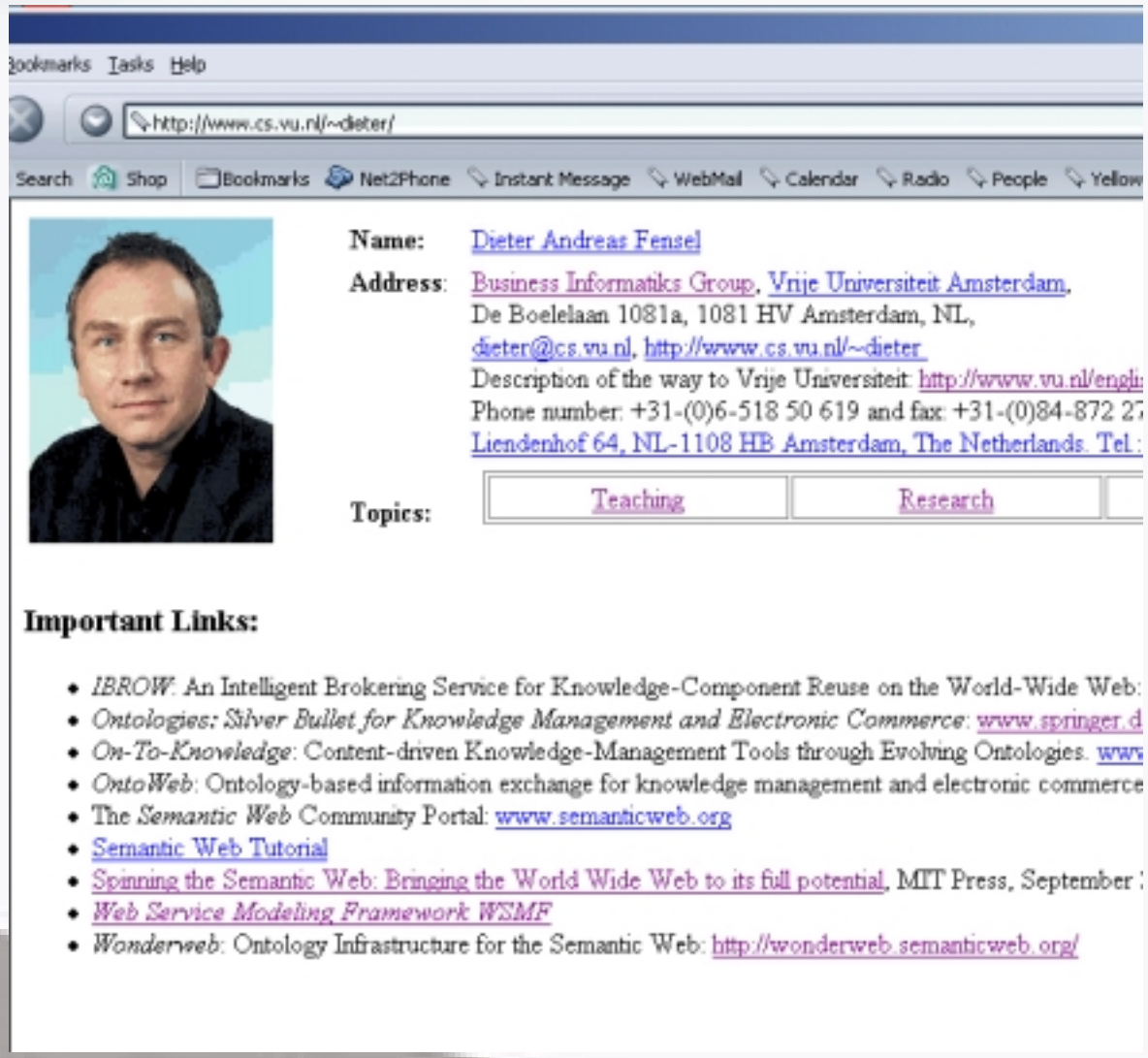
Semantic Web Services

Vision & Truth

What is a web service?

- **Empirical Answer**
- **Three Definitions**

Vision & Truth



Semantic Web Services

Vision & Truth



Vision & Truth



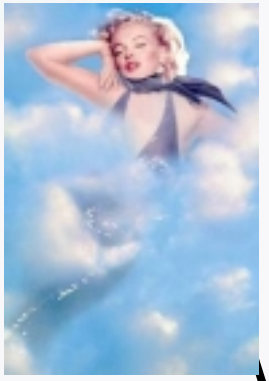
Semantic Web Services

Vision & Truth

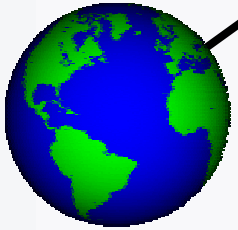


Semantic Web Services

Vision & Truth



Def 2. New concept for
eWork and eCommerce



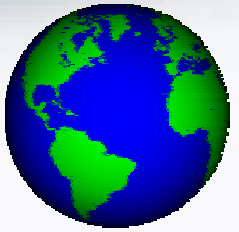
Def 1. Software Architecture

Def 3.
New programming technology



Semantic Web Services

Vision & Truth



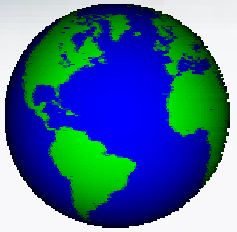
Def 1. Web Services as a Software Architecture

"Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. …

Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service."

IBM web service tutorial

Vision & Truth



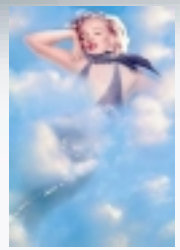
Web Services connect computers and devices with each other using the Internet to exchange data and combine data in new ways.

The key to Web Services is on-the-fly software creation through the use of loosely coupled, reusable software components.

Software can be delivered and paid for as fluid streams of services as opposed to packaged products.

No riscs in terms on investment.

Vision & Truth

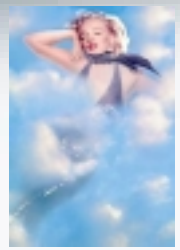


Def 2. Web Services as a new Concept for eWork and eCommerce

„Web Services, are Services accessible via the web“

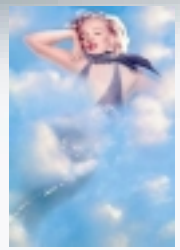
numerious white papers

Vision & Truth



- Business services can be completely decentralized and distributed over the Internet and accessed by a wide variety of communications devices.
- The internet will become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and to provide value-added services.
- The dynamic enterprise and dynamic value chains become achievable and may be even mandatory for competitive advantage.

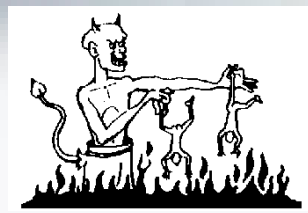
Vision & Truth



- Large companies shrink around their core competencies into small, flexible, and highly profitable units.
- Vica versa, virutal enterprises are set up on the fly reflecting current needs from the market.

eWork and eCommerce will be the two sides of the same coin.

Vision & Truth

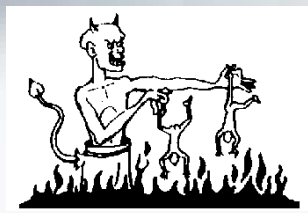


Def 3. Web Services as a programming technology

„Web Services are Remote Procedure Calls (RPC) over HTTP“

current state of the art

Vision & Truth

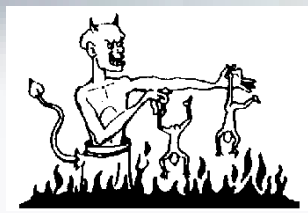


The web is organized around URIs, HTML, and HTTP.

- URIs provide defined ids to refer to elements on the web,
- HTML provides a standardized way to describe document structures (allowing browsers to render information for the human reader), and
- HTTP defines a protocol to retrieve information from the web.

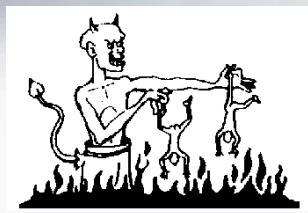
==> Not surprisingly, web services require a similar infrastructure around UDDI, WSDL, and SOAP.

Vision & Truth



UDDI	WSDL	SOAP
↑	↑	↑
URI	HTML	HTTP

Vision & Truth



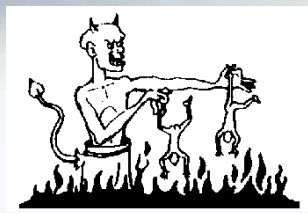
- **UDDI** provides a mechanism for clients to find web services.
- A UDDI registry is similar to a CORBA trader, or it can be thought of as a DNS service for business applications.
- White pages: Who is the service provider?
- Yellow pages: What is the service provding?
- Green pages: How can I make use of the service?

Vision & Truth



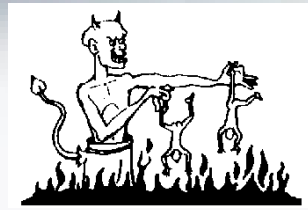
- **WSDL** defines services as collections of network endpoints or *ports*.
- The abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings.
- The concrete protocol and data format specifications for a particular port type constitute a *binding*.
- A port is defined by associating a network address with a binding; a collection of ports define a ***service***.

Vision & Truth



- **SOAP** is a message layout specification that defines a uniform way of passing XML-encoded data.
- In also defines a way to bind to HTTP as the underlying communication protocol.
- SOAP is basically a technology to allow for “RPC *over the web*”.

Vision & Truth

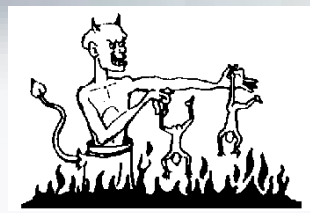


- UDDI, WSDL, and SOAP are important steps into the direction of a web populated by services.
- However, they only address part of the overall stack that needs to be available in order to achieve the above vision eventually.
- There are many layer requires to achieve automatic web service discovery, selection, mediation and composition into complex services.

Vision & Truth

- There are important steps to take to bring web services and fully enabled E-commerce to reality.
- Bringing E-commerce to its full potential requires a Peer-to-Peer (P2P) approach. Anybody must be able to trade and negotiate with everybody else.
- However, such an open and flexible E-commerce has to deal with many obstacles before it becomes reality!
- The issue is ***scalability*** and ***economy in price.***

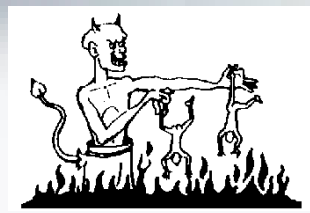
Vision & Truth



Layer / Standard	EDI	RosettaNet	ebXML	SOAP	OAGIS	
Document type	X	X			X	
Semantics	X	X				
Process		X	X			
Exchange Sequence		X	X			
Packaging		X	X	X		
Transport binding		X	X	X		

Semantic Web Services

Vision & Truth



- Many organizations had the insight that message definition and exchange are not sufficient to build an expressive web services infrastructure.
- In addition to UDDI, WSDL and SOAP, standards are proposed such as WSFL, XLANG, ebXML, BPSS, BPML and WSCL.

Vision & Truth

- Still, there are important features missing in all of the mentioned frameworks. Very important is to reflect the *loose coupling* and *scalable mediation* of web services in an appropriate modeling framework.

==> Therefore, we developed a full-fledged **Web Service Modeling Framework (WSMF)**. It provides a rich conceptual model for the development and the description of web services.

Vision & Truth

The WSMF consists of four main different elements:

- *ontologies* that provide the terminology used by other elements;
- *goal repositories* that define the problems that should be solved by web services;
- *web services* descriptions that define various aspects of a web service;
- and *mediators* which bypass interoperability problems.



Vision & Truth

- Mechanized support is needed in finding and comparing vendors and their offers. ***Machine processable semantics*** of information allows to mechanize these tasks.
- Mechanized support is needed in dealing with numerous and heterogeneous data formats. ***Ontology technology*** is required to define such standards better and to map between them.
- Mechanized support is needed in dealing with numerous and heterogeneous ***business logics***. Mediation is needed to compensate these differences, allowing partners to cooperate properly.

Vision & Truth

- Web services can be accessed and executed via the web.
- However, all these service descriptions are based on semi-formal natural language descriptions.

  Therefore, the human programmer need be kept in the loop and scalability as well as economy of web services are limited.

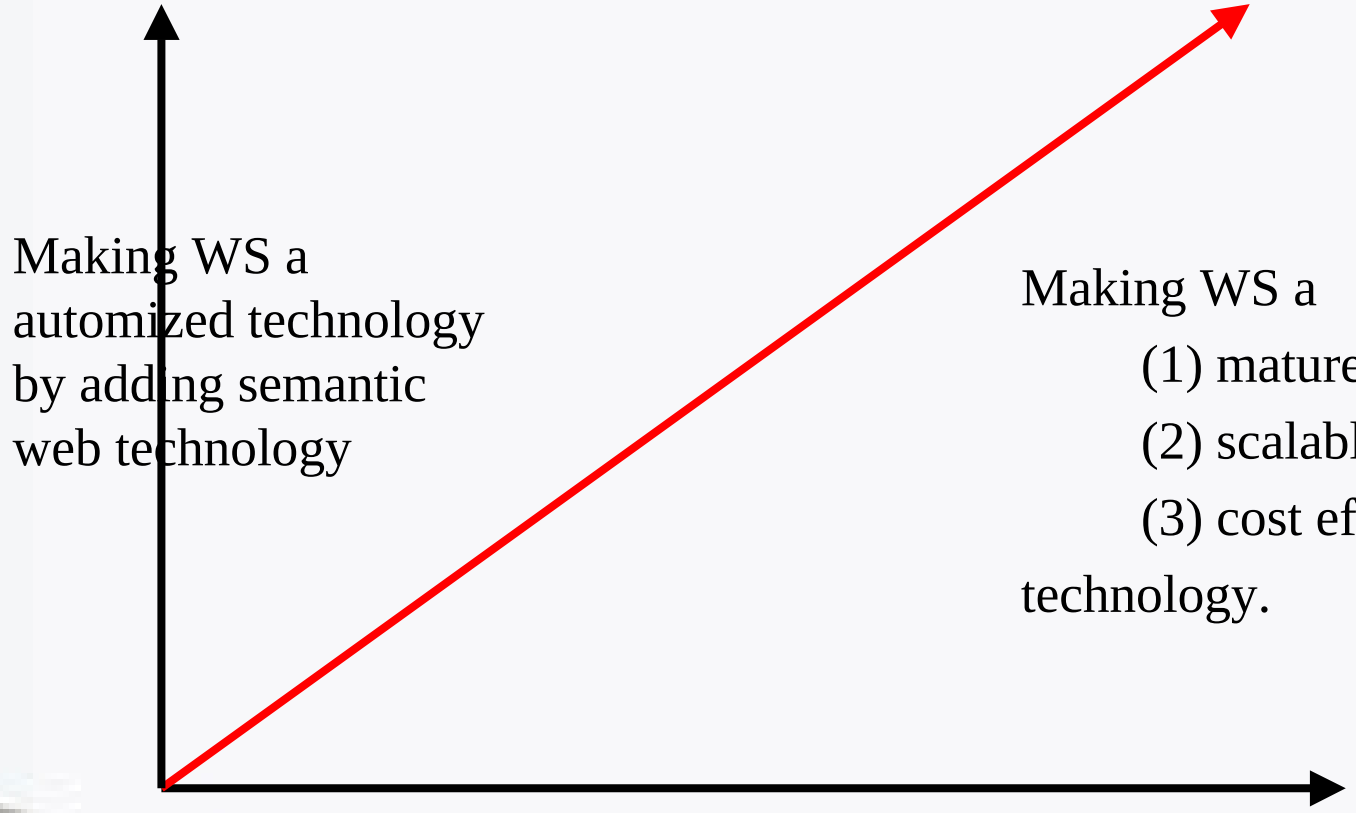
Vision & Truth

- This **semantic web** will provide intelligent access to heterogeneous, distributed information, enabling software products to mediate between user needs and the information sources available.
- Bringing web services to their full potential requires their combination with semantic web technology.
- It will provide mechanization in service identification, configuration, comparison, and combination**.**
- **Semantic Web Services** have the potential to change our life in a much higher degree as the current web already did.

Vision & Truth

- DAML-S will provide support for the following elements necessary to enable efficient interenterprise execution:
  - Public process description and advertisement;
  - discovery of services;
  - selection of services;
  - composition of services;
  - and delivery, monitoring and contract negotiation.
- Without mechanization of these processes, internet-based eCommerce will not be able to provide its full potential in economic extensions of trading relationships.

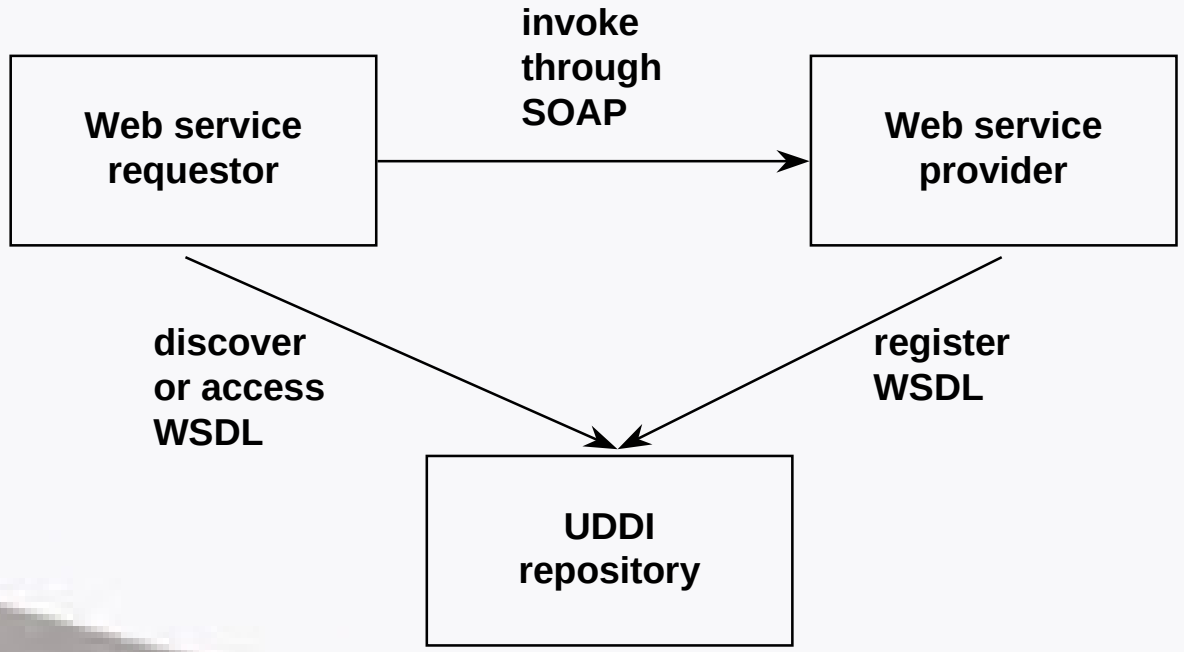
Vision & Truth



Questions?

Truth & Vision

- Web services (SOAP, UDDI, WSDL)
  - Data exchange between two programs in XML format
  - Operate on syntactic level: web services infrastructures do not access data content



Truth & Vision

- Detour: web service infrastructures
  - Application servers
    - Like Oracle's 9iAS, IBM's WebSphere or BEA's Weblogic
    - Provide infrastructure to create SOAP messages, initiate SOAP invocations, and receive SOAP invocations
    - Provide WSDL generation and interpretation functionality
    - Provide UDDI connectivity
  - Non-application server implementations
    - Example: CapeClear (www.capeclear.com)
  - Web service definition and implementation (i. e. web services logic) done by programmer in context of a web service infrastructure
- End detour

Truth & Vision

- Invocation model
  - One way invocation
  - Request/reply invocation
  - Solicit/response invocation
- Invoked entity (service provider)
  - publishes WSDL operation with input and output message
- Invoker (service requester): no concept
  - Especially not a “subroutine” call a la RPC with appropriate stack operations or stub generation

Vision & Truth

- Web Services Interoperability
  - Web Services Interoperability Organization
  - Define interoperable standards versions
  - Provide tools for interoperability testing
  - www.ws-i.org

Truth & Vision

- Example of WSDL
  - Christmas tree
    - h: height of the tree
    - r: radius of the tree
    - l: radius of the flare of the lights
    - Returns number of lights in the tree


xmethods_xmas_iii.pdf

Truth & Vision

- Missing concepts in web services
  - Data definition
    - XML Schema is definition language for input and output message
    - No domain specific data definitions
  - Invocation behavior
    - No operation sequence definition
    - All operations are equal wrt. to behavior. Any restriction to be known (by magic) by invoker
  - Mediation
    - No mediation of data
    - No mediation of behavior

Vision & Truth

- Missing elements in web services (continued)
  - Composition
    - No concepts for composition
  - Trading partner management
    - Web services recognize URIs as endpoints and do not incorporate trading partner management
  - Service level guarantees
    - Web services do not contain any service level agreements
    - Emerging work
      - Web Services Security
        - » www-106.ibm.com/developerworks/library/ws-secure
      - Business transactions (OASIS)
        - » www.oasis-open.org/committees/business-transactions

Vision & Truth

- WSMF (Web Services Modeling Framework)
  - Addresses all deficiencies of web services by providing a complete set of concepts
  - Session 2 of this tutorial will describe WSMF in detail
  - Rest of this session will discuss related work

Truth & Vision

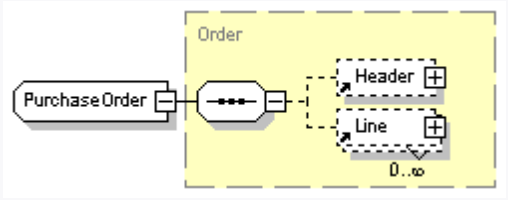
- Related work
  - Data definition
  - Invocation behavior
  - Mediation
  - Composition
  - Trading partner management
  - Service level guarantees

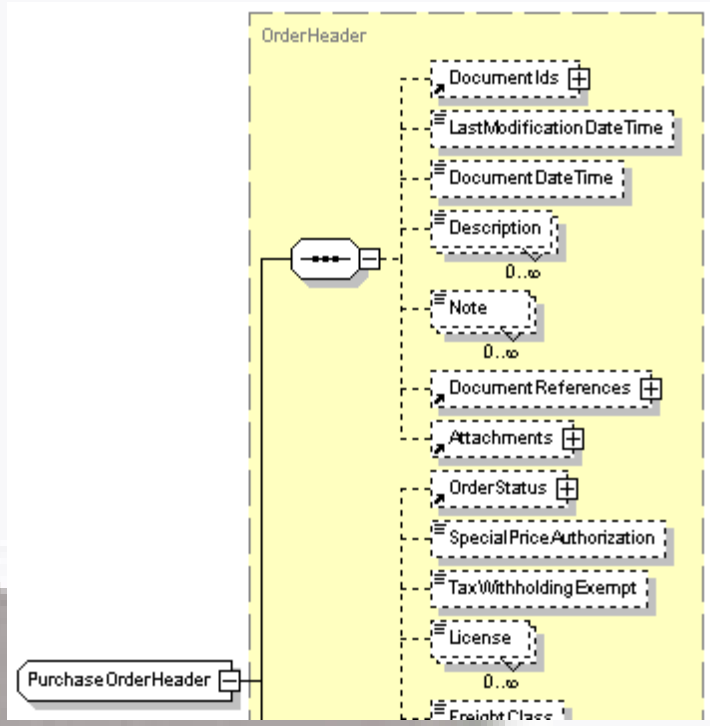
Truth & Vision

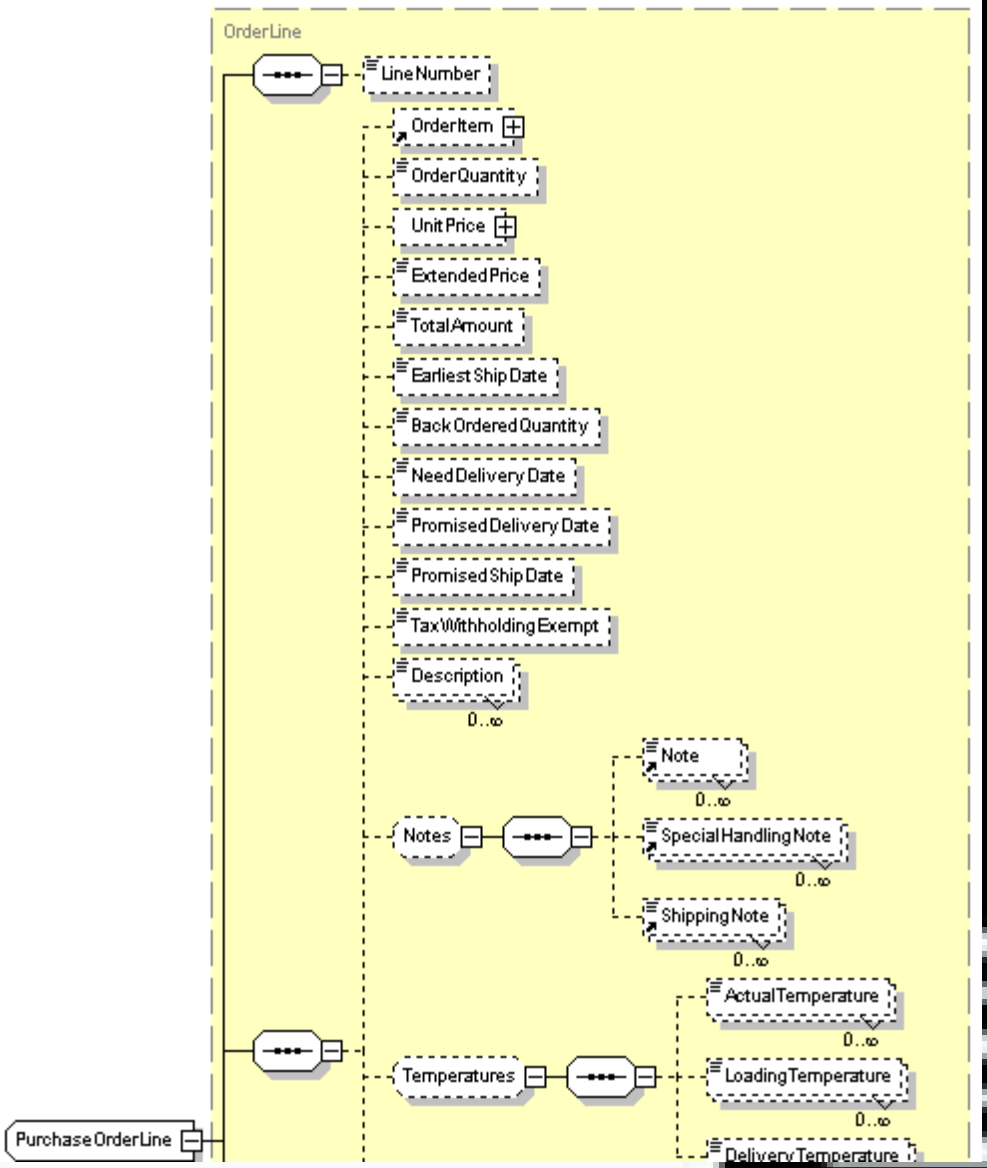
- Data definitions
  - Open Applications Group (www.openapplications.org)
  - RosettaNet (www.rosettanet.com)
  - EDI (www.x12.org)
  - SWIFT (www.swift.com)
  - UBL (www.oasis-open.org/committees/ubl)
  - Many, many more in all major application domains
    - See The XML Cover Pages: www.oasis-open.org/cover/sgml-xml.html
  - Not yet using ontology languages, but XML schema or equivalent syntax to define documents

Truth & Vision

- Example OAGIS PO







Truth & Vision

- Example EDI 850 (purchase order)

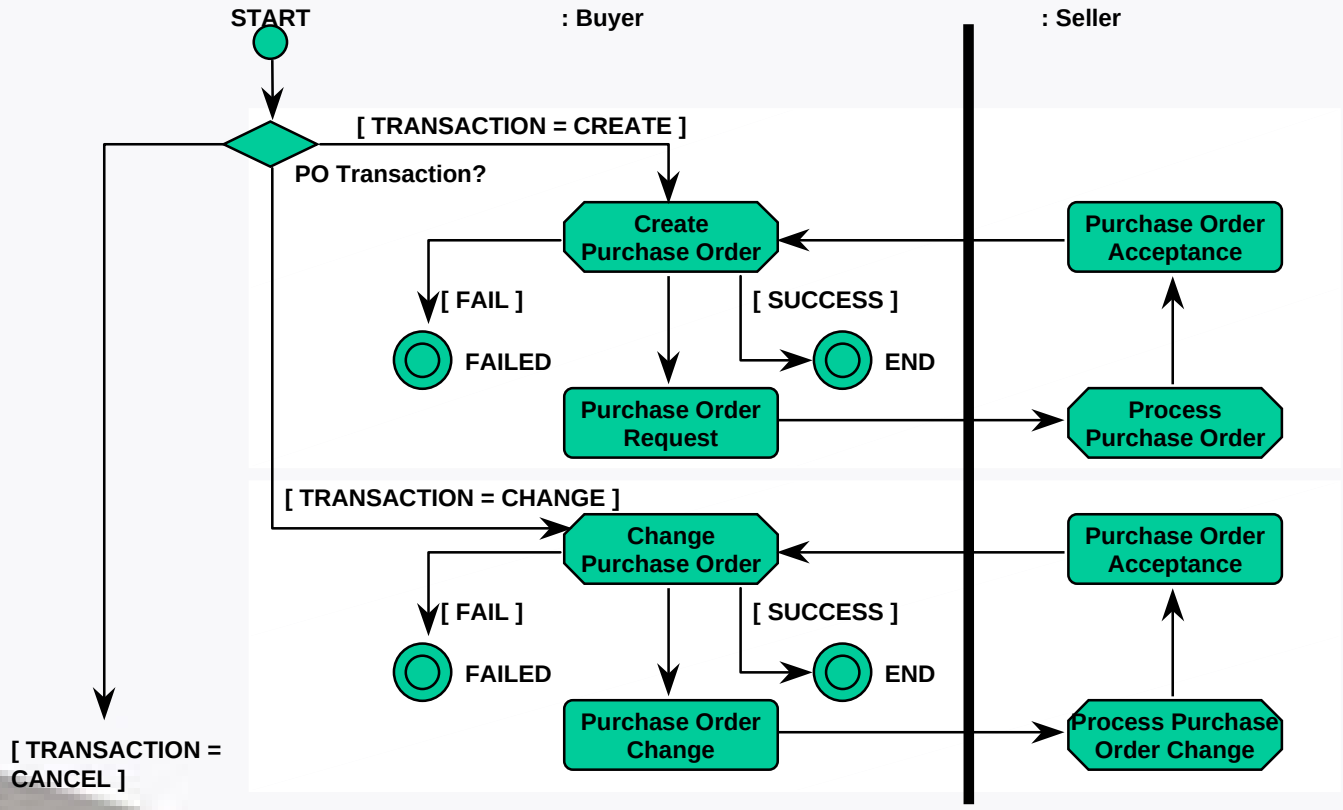
GS~PO~ERPTEST~IMPEXP~20000512~1352~142~X~004010
ST~850~0001
BEG~00~NE~616000000096~~20000420
PER~BD~JACKSON, DEBBIE
FOB~DF~ZZ~Road Freight
N1~SU~SUPPLIER NAME INC~92~999888
N2~SUPPLIER REP
PO1~0001~2~EA~~~PN~BOEING PART NUMBER 19
CTP~~~0~2~EA
PID~F~~~~Burns Part
TD5~~GA~RD
TXI~LS~~100~CD~46020106~~~~178 000 010
SCH~2~EA~~~002~20000420
N9~55~0000
N9~TX~R&D EX
N9~C7~0000

Vision & Truth

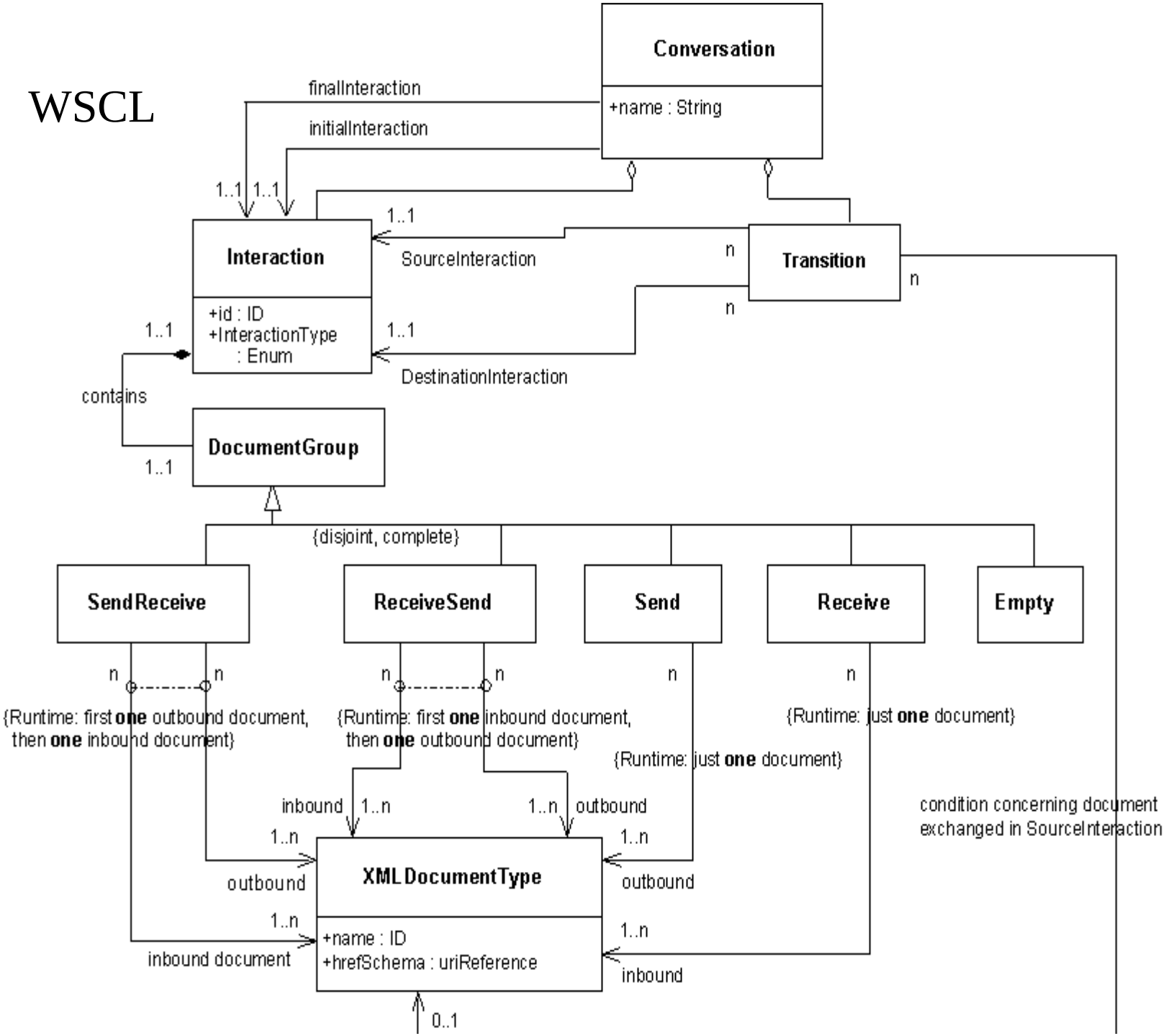
- Invocation behavior
  - No related work for synchronous invocations
    - RPC would be a stretch
  - P2P approach for asynchronous invocations
    - RosettaNet (www.rosettanet.org)
      - Partner Interface Processes (PIPs) defining the behavior of both interaction trading partners
      - Domain specific behavior definition
    - Web services conversation language (WSCL)
      - www.w3.org/TR/wscl10
      - Specification language for behavior

Vision & Truth

- Example RosettaNet PIP 3A4



WSCL


Conversation
+name : String
finalInteraction
initialInteraction
1..1
1..1
Interaction
+id : ID
+InteractionType
: Enum
1..1
SourceInteraction
1..1
DestinationInteraction
n
Transition
n
n
1..1
contains
1..1
DocumentGroup
{disjoint, complete}
SendReceive
ReceiveSend
Send
Receive
Empty
n
n
{Runtime: first one outbound document,
then one inbound document}
{Runtime: first one inbound document,
then one outbound document}
{Runtime: just one document}
{Runtime: just one document}
condition concerning document
exchanged in SourceInteraction
inbound
1..n
1..n
outbound
1..n
outbound
1..n
XMLDocumentType
1..n
outbound
1..n
+name : ID
+hrefSchema : uriReference
inbound document
1..n
inbound
0..1

Vision & Truth

- Mediation
  - No related work at all
  - Problem definition:
    - Matching interal and external
      - Data definitions
      - Event exchange behavior
    - Data definition example
      - EDI purchase order to be matched with RosettaNet purchase order
    - Behavior example
      - EDI behavior (no acknowledgments) to be matched with RosettaNet partner interface processes (with acknowledgments)

Vision & Truth

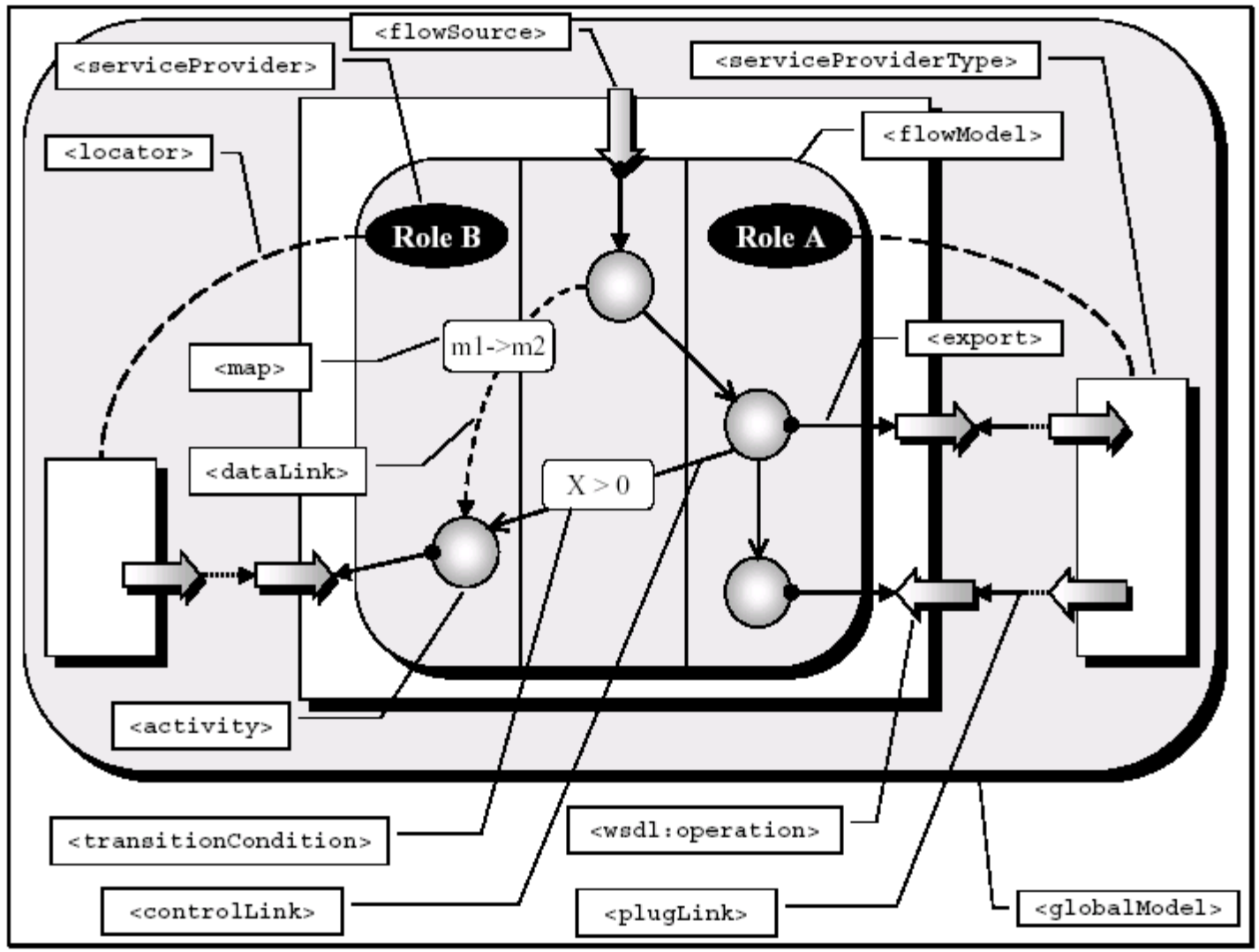
- Composition
  - So far unclear definition of “composition”:
    - Composition in the part-of sense, i. e. larger part encapsulates web services and exposes itself as a web service
      - Analogy: method invocations as part of method definition
    - Composition in the sequencing sence, i. e. definition of the invocation order of web services
      - Behavior as discussed earlier
  - Proposed languages for “composition”
    - WSFL (Web services flow language)
    - BPML (Business process modeling language)
    - ebXML BPSS (Business process specification schema)
    - XLANG (“Web Services for Business Process Design”)

Vision & Truth

- Composition (continued)
  - WSFL
    - www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
    - Message definitions
    - Port types
      - Set of operations with their input and output messages
    - Service provider
      - Set of port types
    - Flow model
      - Flow model for each service provider. Defines invocation sequence of operations of port types
    - Global model
      - Relates operations of all service providers
  - Page 85, ticket order example gives good insight into the workings of WSFL

Vision & Truth

WSFL Example



Vision & Truth

- Composition (continued)
  - BPML (www.bpmi.org)
    - BPML is a workflow definition language with no reference to web services or their composition
    - Data format is XML since process language contains XPATH expressions
    - Very elaborate process model that includes concepts for
      - Inter-workflow communication (message exchange between ongoing workflow instances)
      - Participants
      - Closed and open transactions
      - Compensation
      - Recovery

Vision & Truth

- BPML example

```
<process name="TrackTrouble">
<supports abstract="Customer"/>
<message name="troubleReportInput" type="request">
        <xsd:element name="service" type="Service"/>
        <xsd:element name="trouble" type="xsd:string"/>
</message>
<message name="troubleReportOutput" type="response">
        <xsd:element name="cookie" type="TrackTrouble"/>
</message>
<message name="getStatusInput" type="request"/>
<message name="getStatusOutput" type="response"> … </message>

<sequence name="reportAndTrack">
        <operation name="reportTrouble">
                <participant name="reportTroubleForm"/>
                <input name="troubleReportInput"/>
                <output name="troubleReportOutput">
                        <assign target="cookie" select="TrackTrouble/text()"/>
                </output>
        </operation>
```

Vision & Truth

- BPML example (continued)

```
<operation name="findProvider">
        <participant select="troubleReportInput/service"/>
        <output message="getProviderInput"/>
        <input message="getProviderOutput"/>
</operation>
<operation name="createTicket">
        <participant select="getProviderOutput/provider"/>
        <output message="openTicketInput">
                <assign select="troubleReportInput/trouble"/>
                <assign select="TrackTrouble/text()" target="customer"/>
        </output>
        <input message="openTicketOutput"/>
</operation>
<consume name="notifyCustomer">
        <input message="ticketClosed"/>
</consume>
</sequence>
</process>
```

Vision & Truth

- Composition (continued)
  - ebXML BPSS (www.ebxml.org)
    - Look under “Specifications”
  - ebXML BPSS is a process specification language
  - Specific emphasis on document exchange
    - Business data messages
    - Acknowledgment messages
  - Message activities
    - Non-repudiation
    - Confidential
    - Encrypted

Vision & Truth

- ebXML BPSS example
  - Please refer to the specification; language (XML) is chatty ;-)

Vision & Truth

- Composition (continued)
  - XLANG
    - www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
  - Extension of WSDL for behavior definition
  - Main constructs (block structured)
    - Activation operation, i. e. WSDL operation that starts the behavior
    - Operation, delayFor, delayUntil, raise
    - Empty, Sequence, Switch, While, All, Pick
    - Correlation
    - Context

Truth & Vision

- XLANG Example

```
<?xml version="1.0"?>
<definitions name="StockQuoteProvider"...>
<service>
        <xlang:behavior>
                <xlang:body>
                        <xlang:sequence>
                        <xlang:action operation="AskLastTradePrice"
                                port="pGetRequest" activation="true"/>
                        <xlang:action operation="SendLastTradePrice"
                                port="pSendResponse"/>
                        </xlang:sequence>
                </xlang:body>
        </xlang:behavior>
        </service>
</definitions>
```

Vision & Truth

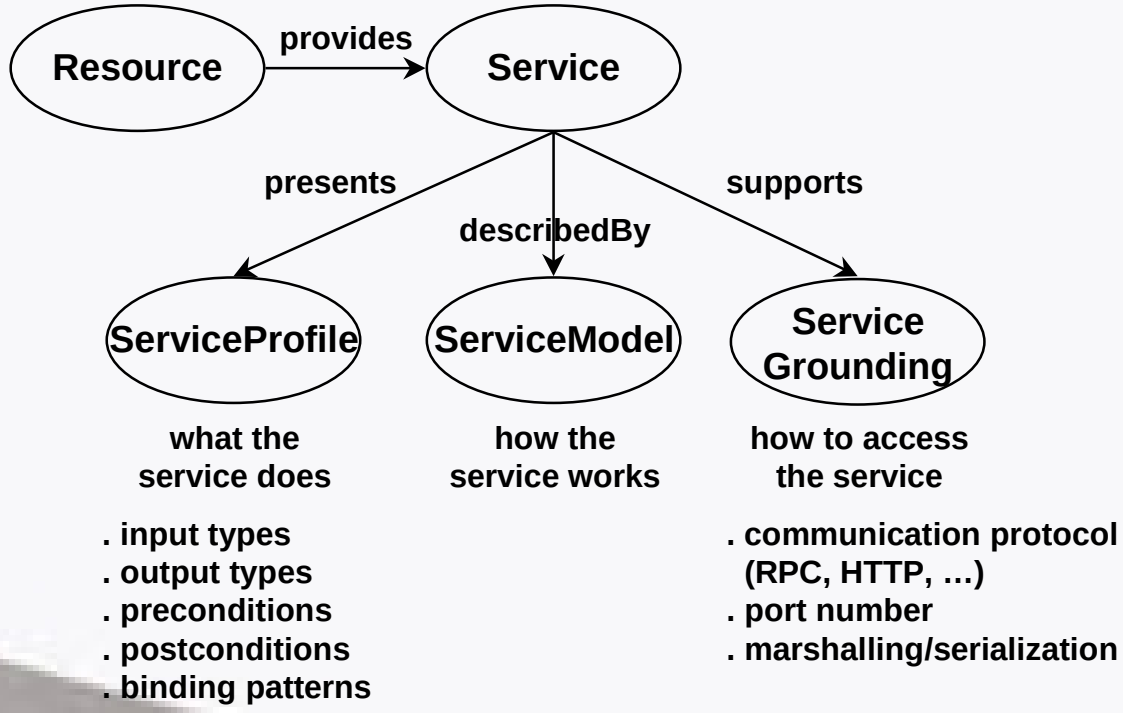
- Trading partner management
  - ebXML
    - CPP: Collaboration partner profile. Properties of collaboration partners
    - CPA: Collaboration partner agreement. Agreement between collaboration partners about rules of engagement
  - EDI
    - Document type 838 that allows the communication of trading partner attributes
  - ERPs
    - ERP internal management of trading partner information that is available and accessible

Vision & Truth

- Service level guarantees
  - Reliable message transmission over unreliable network
    - RosettaNet
      - RNIF (RosettaNet Interaction Framework)
      - Time-outs for expected delays in responses (“time to perform”)
      - Retry counter
      - Resending of messages
      - Agreement in which state interaction considered failure or success, no explicit message sent to indicate failed or succeeded behavior
    - Emerging: business transactions
  - Security
    - Signatures, encryption, non-repudiation
    - Emerging: web services security (see earlier)

Vision & Truth

- DAML-S - An overview
  - DAML (DARPA Agent Markup Language)
  - DAML-S: Upper ontology of web services



Truth & Vision

- DAML-S Service Model
  - www.daml.org/services
  - Subclass of ServiceModel: ProcessModel
    - Two main components
      - Process (defined in Process Ontology)
      - Process control (defined in Process Control Ontology)
    - Process Ontology
      - Process
        - » Atomic, simple process and composite process
      - Control constructs
        - » Sequence, Spit, Unordered, Split+Join, Choice, If-Then-Else, Iterate, Repeat-Until
    - Process Control Ontology, Time, Resources
  - Section 2 of the tutorial will introduce DAML-S in more detail

Questions?

Semantic Web Services

WSMF

Web Service Modeling Framework

Introduction

- WSMF provides framework for peer-to-peer communication between any number of endpoints
  - Peer-to-peer (P2P): direct communication between communication partners without intermediaries
  - Endpoints: trading partners, application systems, web services, i. e. “anything that can be invoked and responds”
- WSMF principles
  - Strong de-coupling
    - Each endpoint is fully autonomous in its message exchange behavior and message content
  - Strong mediation
    - Extensive support for bridging data and process differences resulting from strong de-coupling (“compensating strong de-coupling”)

Introduction

- Technical report describing WSMF in detail
  - swws.semanticweb.org under “resources”
- This tutorial gives only an overview of WSMF
  - I encourage everybody of you to check out swws.semanticweb.org for all the important details

Basic Elements of P2P Communication

- Document types
  - E. g. purchase order, shipment notice
- Semantics
  - E. g. shipment date is date when shipper took over the goods for shipping, NOT the date the shipper was contacted first or NOT the date when the goods are expected to be delivered
  - E. g. 9/6/14 means 6/9/2002 means 9.6.2002 :-)
- Transports
  - E. g. http, smtp, ftp, VAN, TCP/IP, beep, EDIINT
- Exchange sequence definition
  - E. g. exactly once semantics achieved over unreliable networks through combination of retry, time-out, re-sending, duplicate detection and upper retry limit

Basic Elements of P2P Communication

- Process definition
  - E. g. the sending of a purchase order requires the return of one or more purchase order acknowledgments acknowledging all line items
- Security
  - Encryption
  - Signatures
  - Non-repudiation
- Syntax
  - XML
  - Non-xml like comma-delimited or character delimited
  - Binary

Basic Elements meet Endpoint Autonomy

- The autonomy of endpoints leads to different specifications and uses of basic communication elements
- The consequence is in general that mismatches occur in communication between endpoints
  - Strong de-coupling
- This requires mediation to overcome mismatches
  - Data mismatches
    - Data mediation through data transformation
  - Behavior mismatches
    - Process mediation through process transformation

Main Elements of WSMF

- Ontologies
- Goal repositories
- Web services
- Mediators

Ontologies

- Definition in context of basic communication elements
  - Document types
    - Structure of data communicated
  - Semantics
    - Meaning of data communicated
  - Both are defined by endpoints due to their autonomy
- Definition in context of WSMF
  - Goal and pre-/post-condition support
    - Meaning of goal web service provider addresses
    - Meaning of goal web service client has in mind
  - This is defined by endpoints due to their autonomy

Goal Repositories

- Goals
  - Client can state goal it has, i. e. what it tries to accomplish
    - E. g. “book a complete round-trip travel”
  - Endpoint can state goal it addresses, i. e. for what goal it provides service
    - E. g. “book most expensive hotels”
    - E. g. “book affordable hotels” :-)
  - Ontology
    - E. g. definition of round-trip travel
      - Trip between two cities whereby transportation is between the two cities. One destination city and one origination city. Complete means flight, hotel and car whereby flight is between the two cities and hotel and car are booked in the destination city.
      - City means the city boundaries. Hotel in this city means that the hotel has to be in the city boundaries (Trip to LA? :-)).
      - …

Goal Repositories

- Pre-conditions
  - Input that web service requires to function properly
    - E. g. definition of “affordable” or “most expensive”
  - Ontology
    - What does “affordable” mean?
      - Total charge per night per room (including taxes and any applicable fees) is in the range of USD 49 to USD 99
    - “Most expensive” is clear :-)
      - Select the hotel in the city with the highest price per room per night

Goal Repositories

- Post-condition
  - Output web service provides
    - E. g. a complete trip is booked or not at all
  - Ontology
    - What does “booked” mean?
      - Flight in status OK, hotel guaranteed with credit card for late arrival, car guaranteed with credit card.
    - What does “not at all” mean?
      - No flight booked or reserved, no hotel booked or reserved, no car booked or reserved and no financial transaction or obligation existent (or any more, if cancellation fees have been paid)

Web Services

- The naïve approach
  - Web services are either simple or complex
    - In WSDL terminology: operation
  - Complex web services are composition of simple web services and appear themselves as simple web services
  - Complex web services are black boxes that do not exhibit internals to web service requester
    - Hence they appear as simple web services that either succeed successfully or fail completely
- For real-world applications, life is not that simple, its more complex

  Comprehensive WSMF model of web services

Web Services

- Clear distinction
  - Interface definition of web service
  - Implementation of interface of web service

  In the following we focus on the interface definitions

Web Service Definition

- Name
- Reference to goal and pre-/post-conditions
  - Need for mediation
    - Between terminology of goals or pre-/post-conditions and the web service
    - Between the strength of the pre-/post-conditions
  - Web service might strengthen pre-conditions or weaken post-conditions
    - Mediation has to make sure invoker is fine with the restriction
  - Web service might weaken pre-conditions or strengthen post-conditions

Web Service Definition

- Input/output data
  - Formal parameter called ports
  - Parameter can be set/read concurrently with web service execution
  - E. g. making results immediately available to invoker
  - E. g. supplying critical data after the web service invocation
- Error data
  - Error ports
  - Also concurrently accessible during web service execution

Web Service Definition

<detour>

- Concurrent parameter access
  - Failure handling. Web service indicates failure without finishing. Invoker can try to resolve failure and have web service continue.
    - E. g. “No hotel found”. Invoker can provide hotel alternative for web service to try again
    - E. g. non-transactional web service. If hotel/car booking does not succeed, but flight was booked, manual flight cancellation is necessary before web service can end normally
  - Simple web service unavailable during complex web service execution
    - E. g. invoker might select another simple web service and have the complex web service try again

</detour>

Web Service Definition

- Invoked web service proxy
  - Goal definition
    - Selection criteria to determine web service dynamically through e. g. lookup in e. g. UDDI
  - Name, pre-/post-conditions, input data, output data
    - Fixed binding to web service
  - Name passed through input port of complex web service
    - Dynamic binding to web service
  - Input/output ports
  - Error ports

Web Service Definition

- Invoked web service proxy (continued)
  - Web service proxy is important because it acts as a placeholder at definition time for a real web service invocation at execution time
  - Placeholder used later on to sequence the web service invocation
    - E. g. flight booked first, then hotel, then car

Web Service Definition

- Data flow
  - Define data flow between ports of complex web service and its web service proxies
  - E. g. complex web service “book trip” which consists of three proxies “book flight”, “book hotel” and “book car”
    - Destination city is “data flown” to all three proxies
    - Preferred airline is data flown to “book flight”
    - Preferred hotel group is data flown to “book hotel”
    - Preferred car rental agency is data flown to “book car”
  - Data flow constructs
    - Direct flow
    - Split, join
    - Typecast, to achieve data transformation (mediation, see later)

Web Service Definition

- Control flow
  - Data flow might result in parallel proxy execution
  - Additional constructs for further proxy invocation sequencing
    - Sequence
    - Conditional branching
    - For-loops
    - While-loop
    - Parallel execution
- Data flow and control flow together define the final execution sequence of web services defined by proxies

Web Service Definition

- Exception handling
  - Explicit declaration of exception handling within complex web service to handle error situations
    - Exception handling might change binding to another web service for retrying invocation with alternative web service
    - Exception handling might just plainly retry the invocation of the same web service after some time in case web service is down
    - If retry or alternate is successful, fine
    - If retry or alternate is not successful, error code returned through error port
      - Invoker tries to handle error
      - Invoker can abort complex web service
      - Exception handling can start compensation (see next)

Web Service Definition

- Compensation
  - Undoing the effects of web service until point of failure
  - Assumptions
    - Web service invocation produced side effects
      - E. g. ticket booked for flight, hotel reservation made
  - Compensation logically undoes effects
    - E. g. ticket returned, hotel reservation canceled
  - Compensation might cause further side effects
    - E. g. ticket return fee is USD 100
    - E. g. hotel reservation cancellation fee equaling one night if made within 24 hours before expected arrival, otherwise no cancellation fee
  - Compensation invoked by complex web service at the point of no possible successful failure handling

Mediation

- Autonomy of web service providers lead to heterogeneity in at least the following areas
  - Document types (data structures)
    - Data definitions of data communicated
  - Semantics
    - Meaning of data exchanged

    Data mediation

  - Process definition
    - Message interaction patterns, i. e. message exchange sequence

    Process mediation

Data Mediation

- Transformation between different syntax, structure and semantics of messages of interacting endpoints
- Types of mismatches
  - Data-complete match
    - Messages are identical, a copy operator is sufficient
  - Data-complete mismatch
    - Messages can be transformed into each other without semantic loss
    - Might require syntax and/or structure and/or semantic transformation

Data Mediation

- Types of mismatched (continued)
  - Data-overcomplete mismatch
    - Requester sends more than provider requires
    - Provider sends more then requester requires
    - Unnecessary data for provider have to be kept around since the return message from provider (if it exists) might have to be augmented with unnecessary data to construct message for requester (and vice versa for second case)
  - Data-incomplete mismatch
    - Requester does not send enough data for provider
    - Provider does not send enough for requester
    - More data has to be obtained from provider or requester to accomplish transformation
      - If impossible, mediation fails and communication is impossible

Data Mediation

- One-step transformation approach
  - Difficult since transformation rules have to deal with all aspects (syntax, structure, semantics) at the same time
- Multi-step transformation approach
  - Extract data from native source message syntax into data mediation syntax and structure
  - Transform concepts based on ontology description from source message ontology into target message ontology
    - Within data mediation syntax
    - Semantic preserving transformation
  - Write data to native target message syntax from data mediation syntax and structure

Data Mediation

- Direct transformation
  - n * (n-1) transformations necessary between n endpoints for one message type
- Indirect transformation
  - Intermediate conceptualization
    - Ontology superset of endpoints
  - Only 2n transformations necessary
  - Pays off for n >= 3

Process Mediation

<detour>

- Basic communication patterns
  - Client/Server
    - Client invokes server and has to adjust to server defined interface
  - Peer-to-peer
    - Both peers agree on peer protocol and both might have to adjust internally to agreed upon peer protocol
  - Hub
    - Interacting endpoints communicate through hub (third party)
    - Third party provides different interfaces so that endpoints do not have to adjust at all or only slightly

</detour>

Process Mediation

- Types of mismatches
  - Precise match
    - Messages sent by sender are expected by receiver and vice versa in correct order
  - Unresolvable mismatch
    - Messages sent by sender are not expected by receiver
    - Messages expected by receiver are not sent by sender
    - If messages cannot be consumed, merged, re-arranged or artificially produced, an unresolvable mismatch happens
  - Resolvable mismatch
    - Mismatch scenarios as before
    - BUT, message can be consumed, merged or re-arranged so that sender's or receiver's message exchange behavior is achieved

Data and Process Mediation

- Data mediation and process mediation happen concurrently
- E. g. once a message is received from a sender
  - It is first transformed into the receivers semantics
  - And then merged, consumed or re-arranged to fit the receivers patterns
    - Or is just perfectly fits, the happy case:-)
- Proxies, data flow and control flow together provide the concepts to provide mediation for the discussed mediation types

Complex Web Service Composition

- WSMF provides concepts to compose complex web services addressing many limitations in current proposals
- WSMF allows to follow different composition approaches
  - Provider-based composition
  - Client-based composition
  - Declarative specification of composition

Complex Web Service Composition

- Complex web service composition approaches
  - Provider-based composition
    - Provider has web services and provider combines them itself into a complex web service
    - Requester does not realize that the composition took place and accesses the complex web service like any other web service
  - Client-based composition
    - Requester combines various web services of one or several web service providers
    - Requester has to “get it right” since provider does not know about composition at all and hence cannot supervise execution
      - Correct web services to invoke
      - Correct order of invocation
      - Correct data and process mediation
      - Correct error handling

Complex Web Service Composition

- Complex web service composition approaches (continued)
  - Advanced case: declarative specification of composition
    - Provider provides a declarative specification of correct complex service composition without implementing it
    - Requester follows declarative specification
      - And “gets it right” automatically by construction
    - Provider assured that correct invocation sequence and mediation is followed

Summary

- WSMF provides concepts to implement semantic web services
- WSMF designed to address the most complex real-world scenarios
- WSMF is based on
  - Strong de-coupling
  - Strong mediation

    to overcome heterogeneity due to endpoint autonomy

Semantic Web Services

Mediation & Matchmaking

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Semantic Web Services and Open Multi Agent Systems

- Web Services on the SW are similar to Open Multi-Agent Systems:
  - Service availability is dynamic
    - Available services are continually changing and evolving
    - Alternative or novel services appearing due to the emergence of new businesses
    - Services may be withdrawn if unprofitable, or if businesses providing the service fail
  - Agents may enter or leave a MAS at will
    - No guarantees of availability
    - Registries required to find available agents with given capabilities

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Semantic Web Services and Open Multi Agent Systems

- Web Services on the SW are similar to Open Multi-Agent Systems:
  - Services are heterogeneous in that:
    - Their capabilities
    - The ontologies they use to describe the service
    - The exchanged information
  - Agents are heterogeneous
    - Agents may differ in functionality, representation, reasoning methods, and software architecture

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation, Middle Agents and Multi Agent Systems

- Middle Agents are used to:
  - Provide basic mediation services
  - Coordinate mediation services according to a given set of protocols, conventions and policies
  - Ensure reliable service mediation in terms of QoS and trust management within and across heterogeneous MAS.

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation via Middle Agents

- Middle Agents provide mediation for:
  - Processing agent capabilities and service descriptions
    - e.g. matching requests with advertisements
  - Semantic Interoperability between Agents & Services
    - e.g. resolving semantic mismatches between capability descriptions, or identifying articulations between ontologies
  - Management of Data and Knowledge
    - e.g. storing and managing registered advertisements; adapting and refining matching algorithms.

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Coordinated Mediation between Agents

- Middle Agents may mediate communication between agents:
  - Agents may use different communication languages
    - e.g. SOAP vs. KQML
  - Agent transactions may assume different protocols and policies
    - e.g. virtual marketplaces may assume different auction policies
  - Semantic mismatch of knowledge may require intermediary to translate between ontologies

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Reliable Mediation

- Quality of Service
  - Middle agent should comply with data requirements, standards and policies regarding knowledge and data stored
- Trust Management
  - Provide guarantees that service providers provide the service they advertise
  - Prevent abuse of shared, private information
    - such as selling contact information or preferences
  - Provide Security Authorization and Verification services

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Service Discovery & Mediation

- Why the need for service discovery?
  - Internet is a dynamic environment
    - Available services are continually changing and evolving due to competition
    - Alternative or novel services appearing due to the emergence of new businesses
    - Services may be withdrawn if unprofitable, or if businesses providing the service fail
  - Scalability
    - In recent years, there has been a huge increase in the number of:
      - (Human-oriented) B2C services on the web
      - B2B services available for eCommerce

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Types of Middle Agents

- Middle Agents can be categorized by what knowledge is shared between provider and requester.
  - Providers have capabilities
  - Requesters have preferences

Preferences initially known by	**Capabilities initially known by**		
	Provider	*Provider & Middle Agent*	*Provider, Middle Agent & Requester*
Requester	***Broadcaster***	***Front-Agent***	***Matchmaker/ Yellow Pages***
Requester & Middle Agent	***Anonymizer***	***Broker/Facilitator***	***Personal Assistant/ Recommender***
Requester, Middle Agent & Provider	***Blackboard***	***Introducer/ Bodyguard***	***Arbitrater***

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Service Requests and Service Queries

- Service Requests
  - Description of a desired service, often in terms of relevant inputs and expected outputs.
  - A list of services or service capabilities are returned.
    - Find me a service that can book a flight reservation to Italy
- Service Queries
  - Desired invocation of a service, including attribute values
  - A service is invoked and the results are returned
    - Book a flight to Rome, Italy from Pittsburgh on the 5th June

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation, Requests and Queries

- The type of mediation used can affect the types of interaction between agents
  - Agents requesting services from Middle Agents receive list of candidate services
  - Agent then selects a service provider from the candidate list and submits a query
- However, it would be meaningless to submit capability requests to mediators that do not release capability descriptions
  - e.g. Brokers

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation and Scalability

- Several approaches to distributed middle agents
  - Share information between Middle Agents to provide reliable, robust service in case of failure
  - Propagate requests across a web of Middle Agents
  - Create domain specific Middle Agents, to focus and optimize search & matching

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation and Scalability

- Middle Agents providing capability descriptions scale well
  - Can send requests to several middle agents simultaneously
    - Can then select a service from the aggregated results
    - Can benefit from disjoint repositories
  - However, requesters are responsible for coordination between agents
    - May result in overloading one provider whilst another is lightly loaded

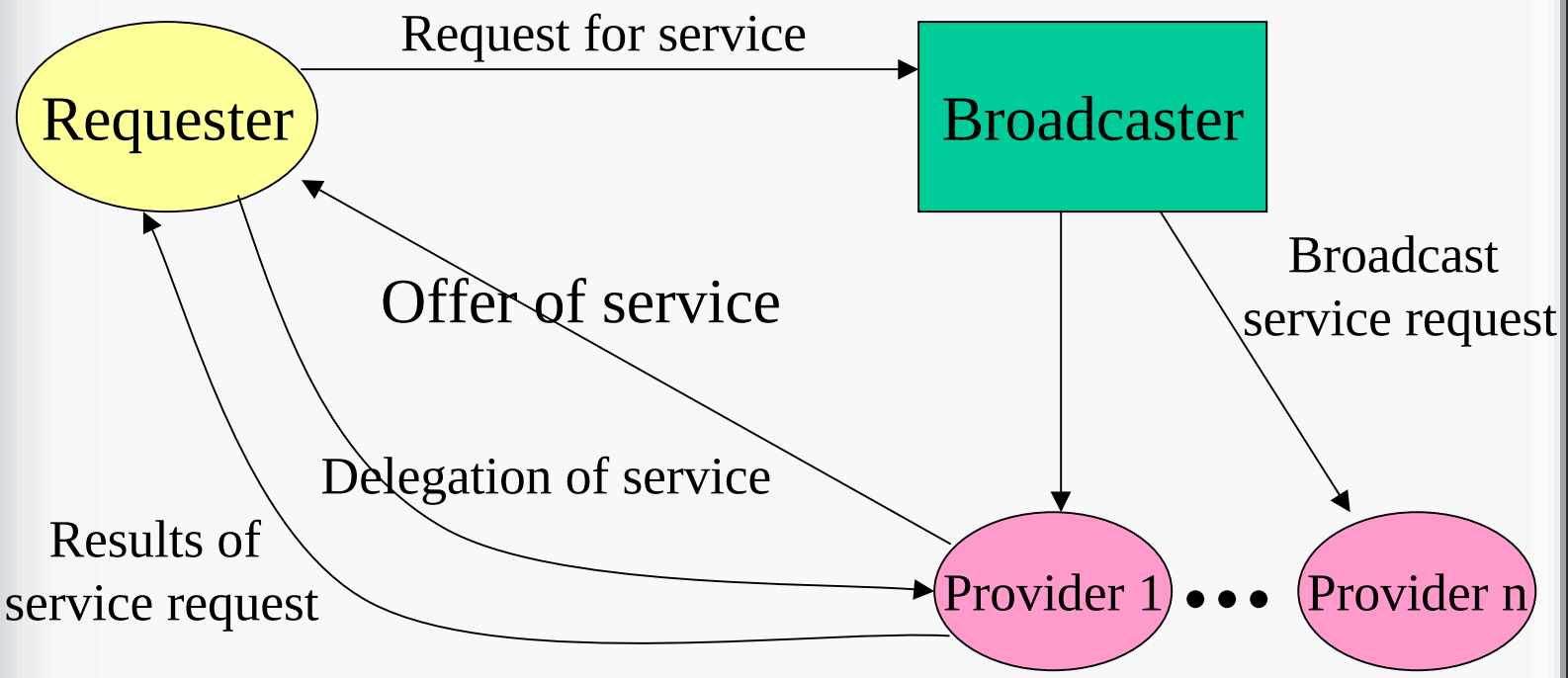
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Mediation and Scalability

- Middle Agents providing communication mediation scale poorly
  - Single Point of Failure
    - If more than one query sent, then more than one service invoked.
    - Undesirable if services charge for invocation
  - Middle Agents can coordinate based on load, availability etc.
    - Mediation can result in load-balancing, resulting in improved response and higher QoS

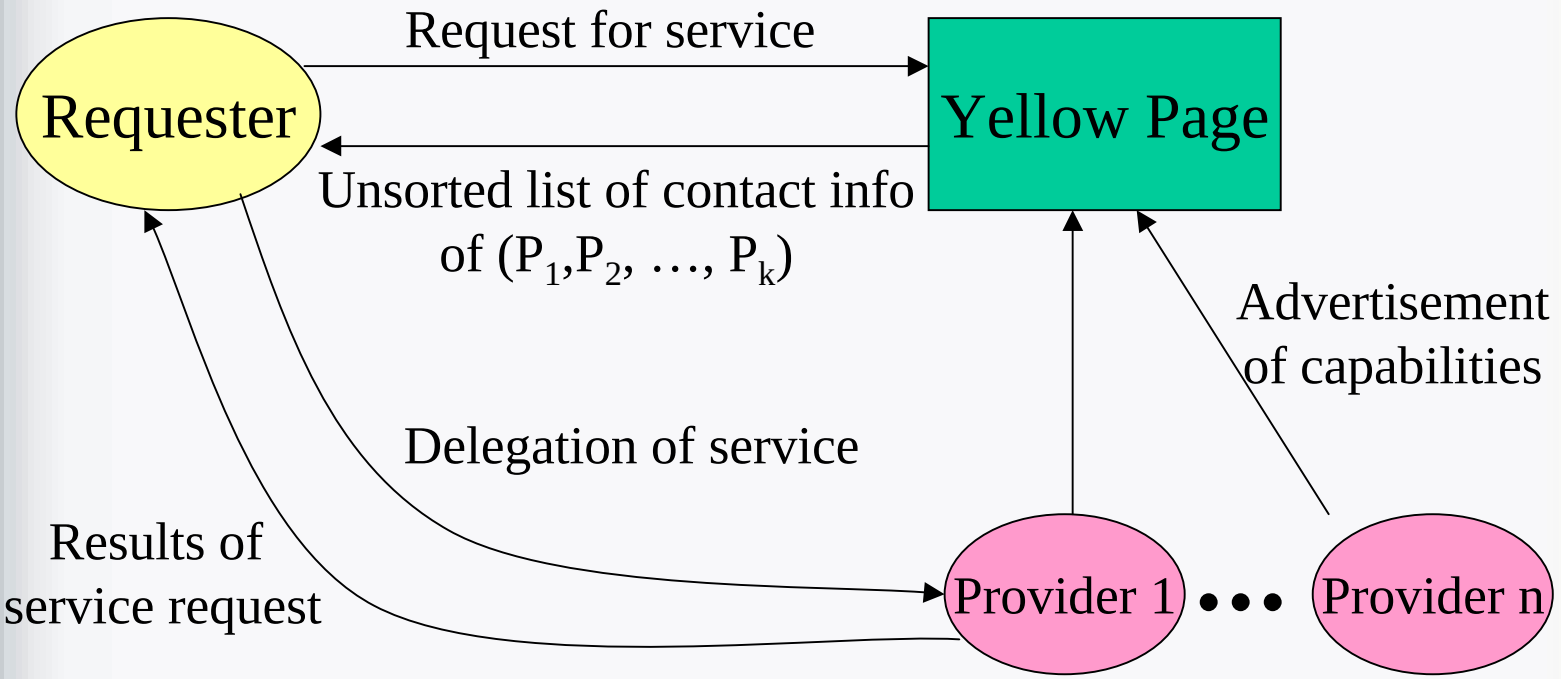
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Broadcaster Middle Agents



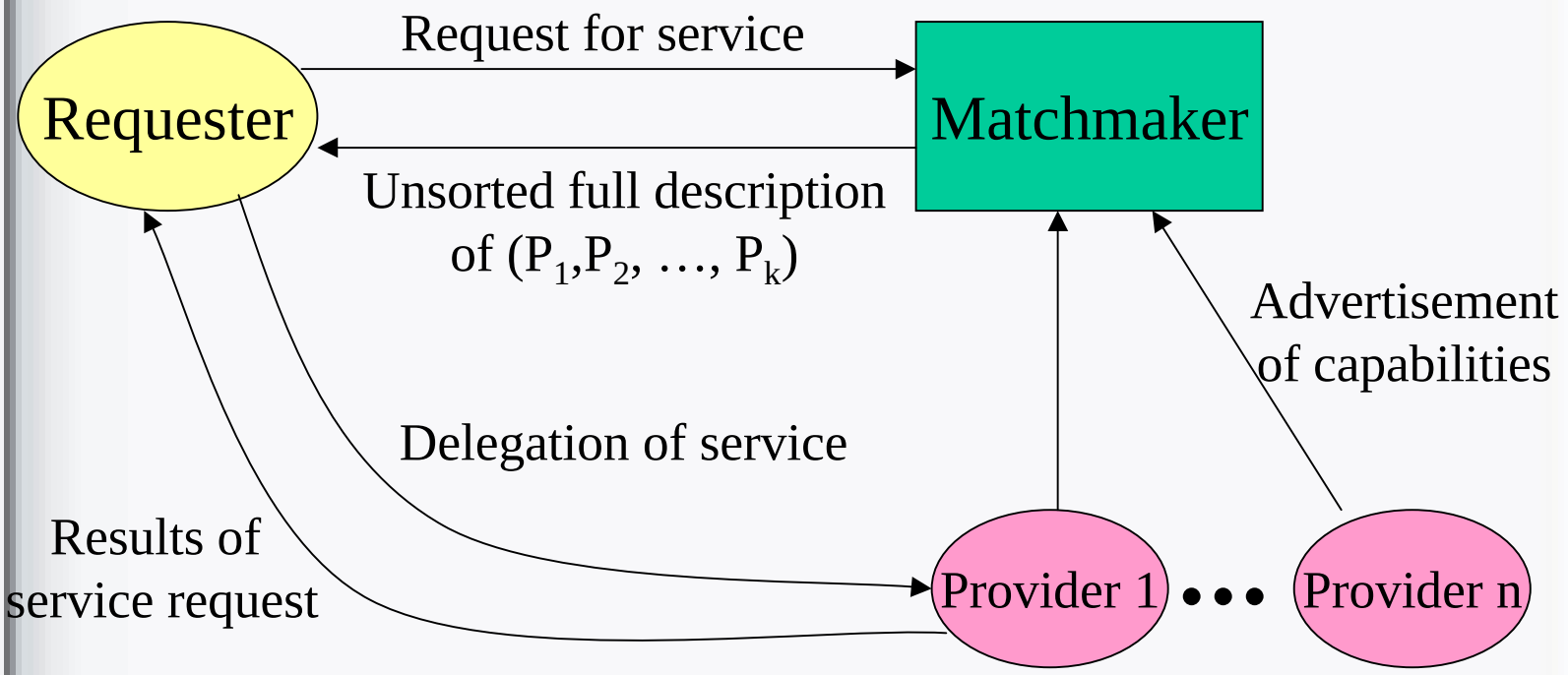
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Yellow Pages Middle Agents



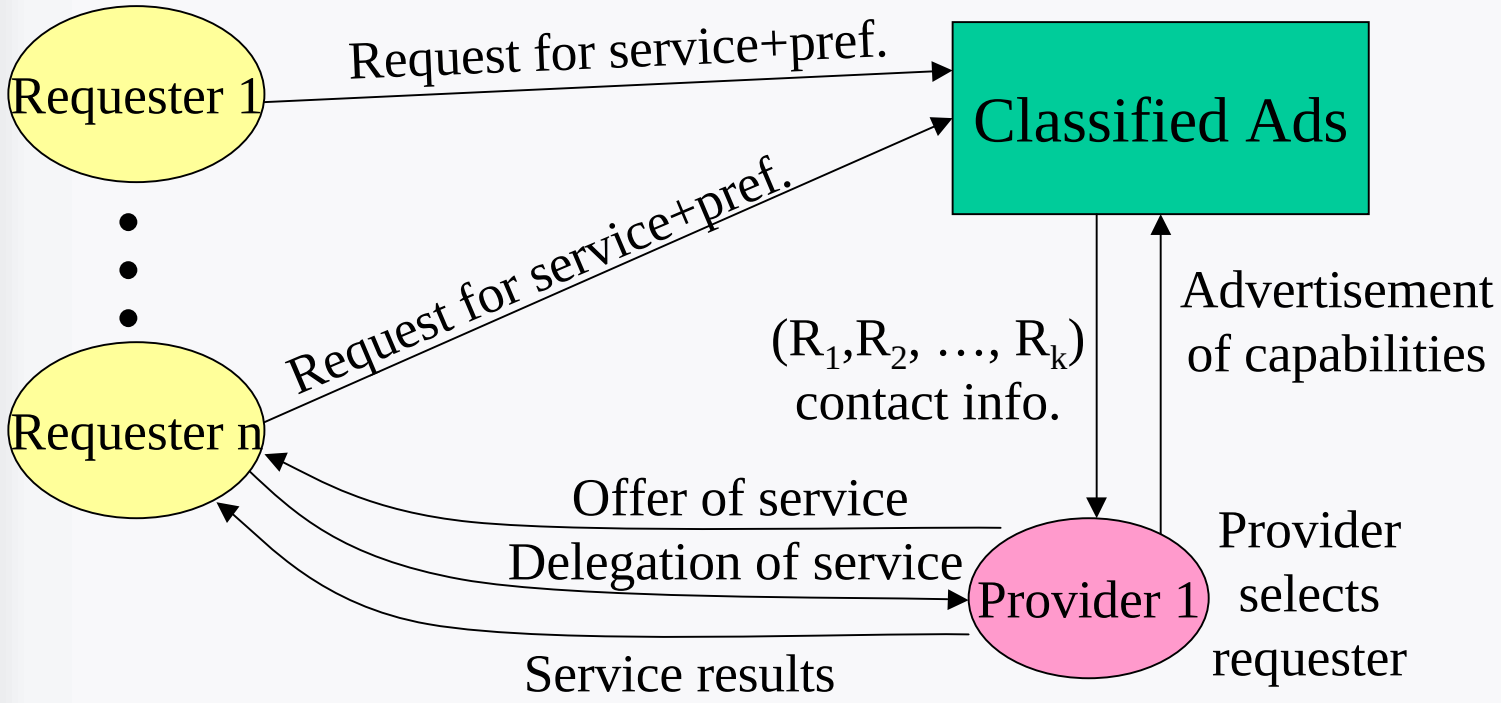
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Matchmaker Middle Agent



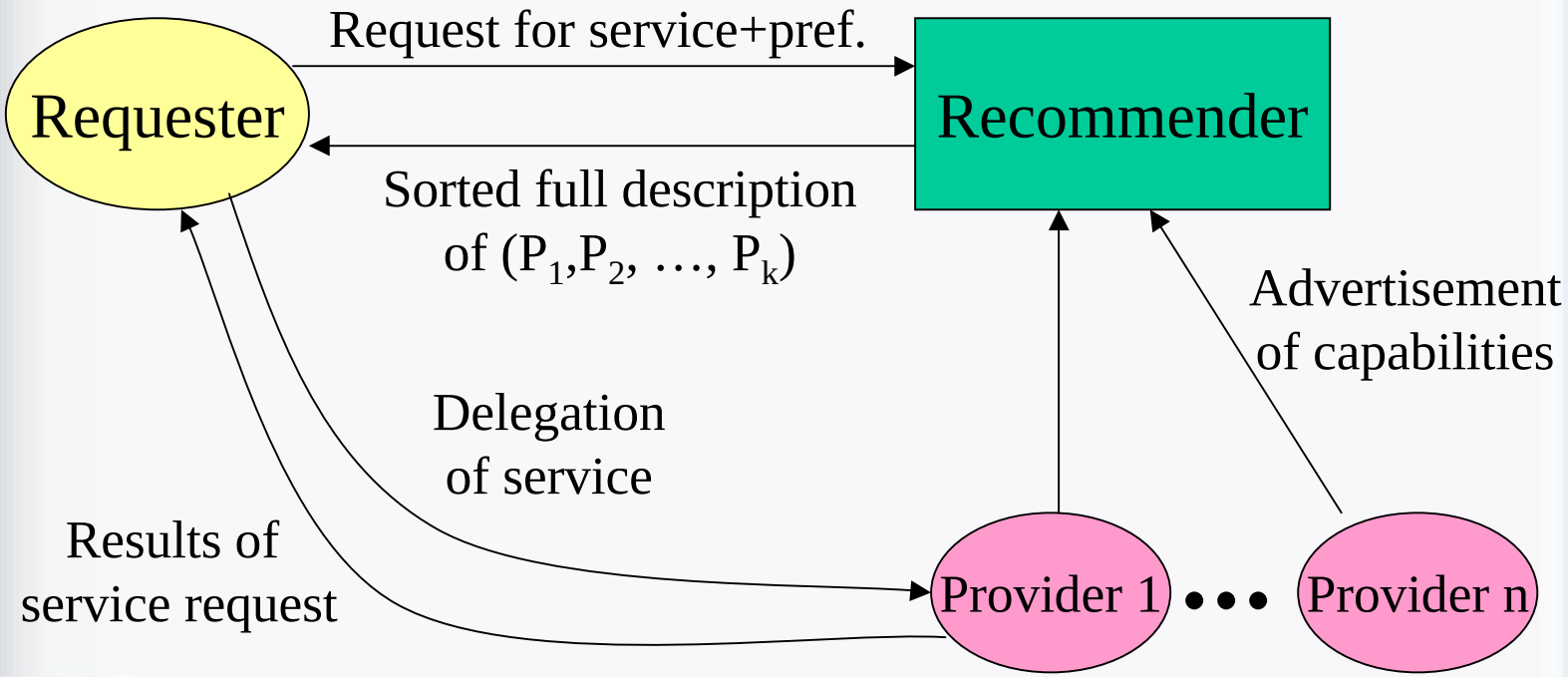
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Classified Ads Middle Agent



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

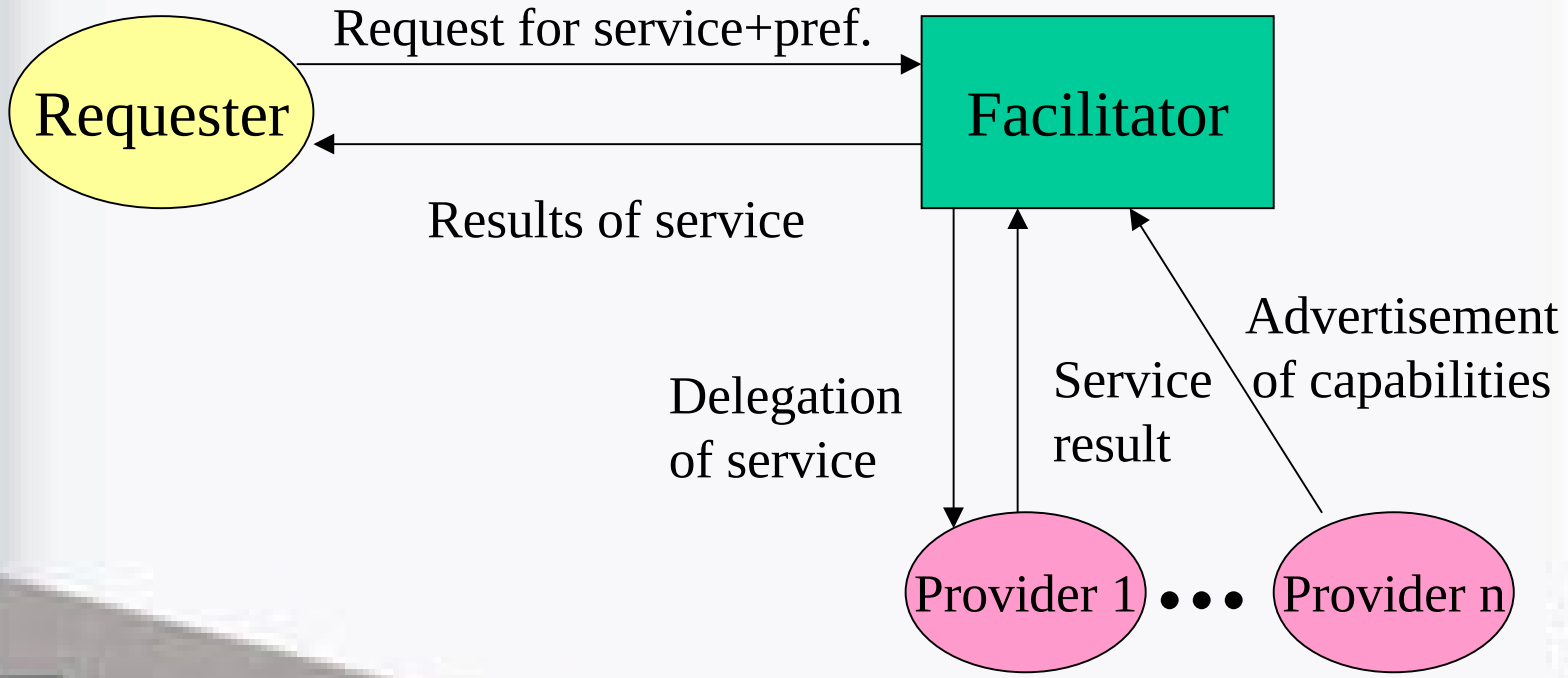
Recommender Middle Agent



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

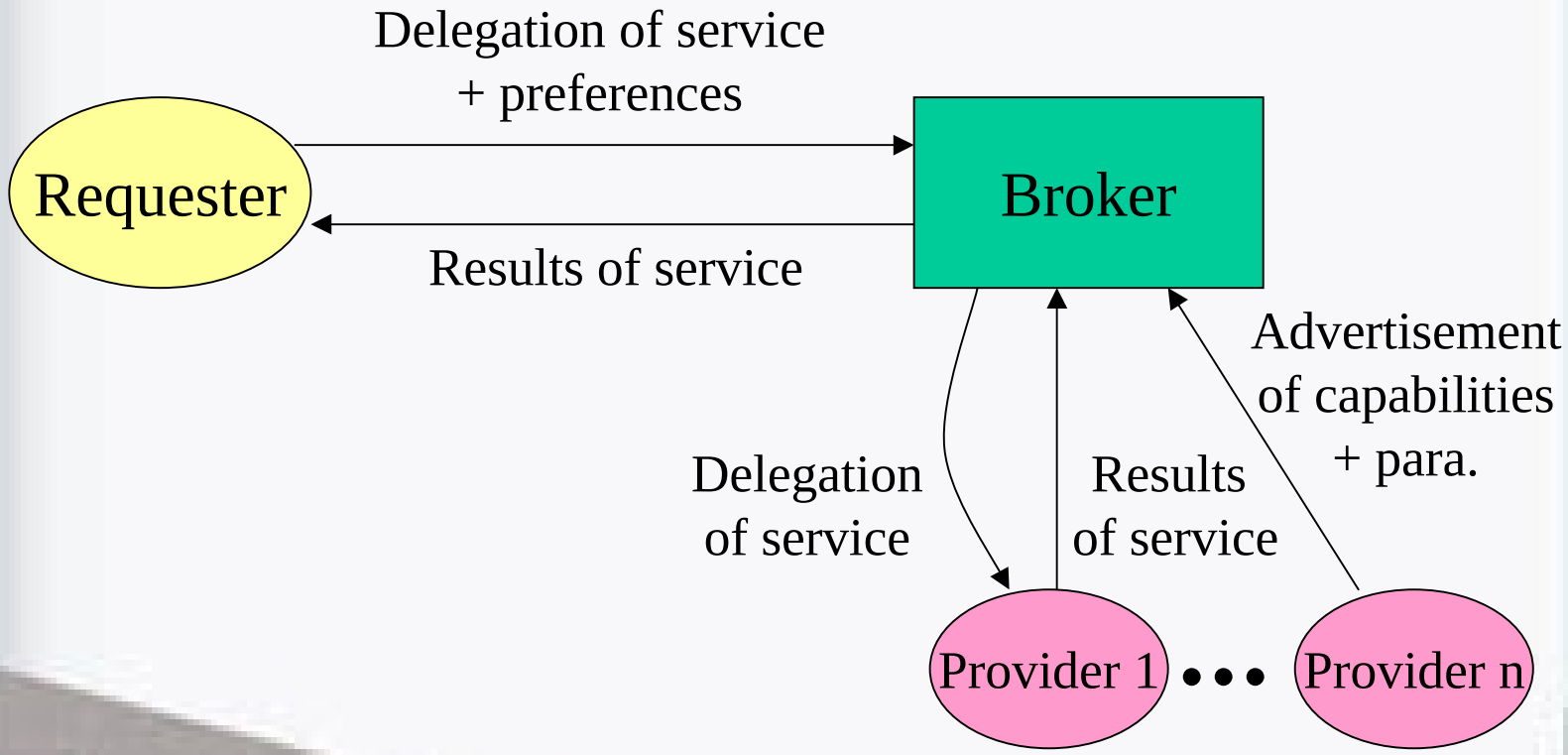
Facilitator Middle Agent

Combines Agent Location and Transaction Phases



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

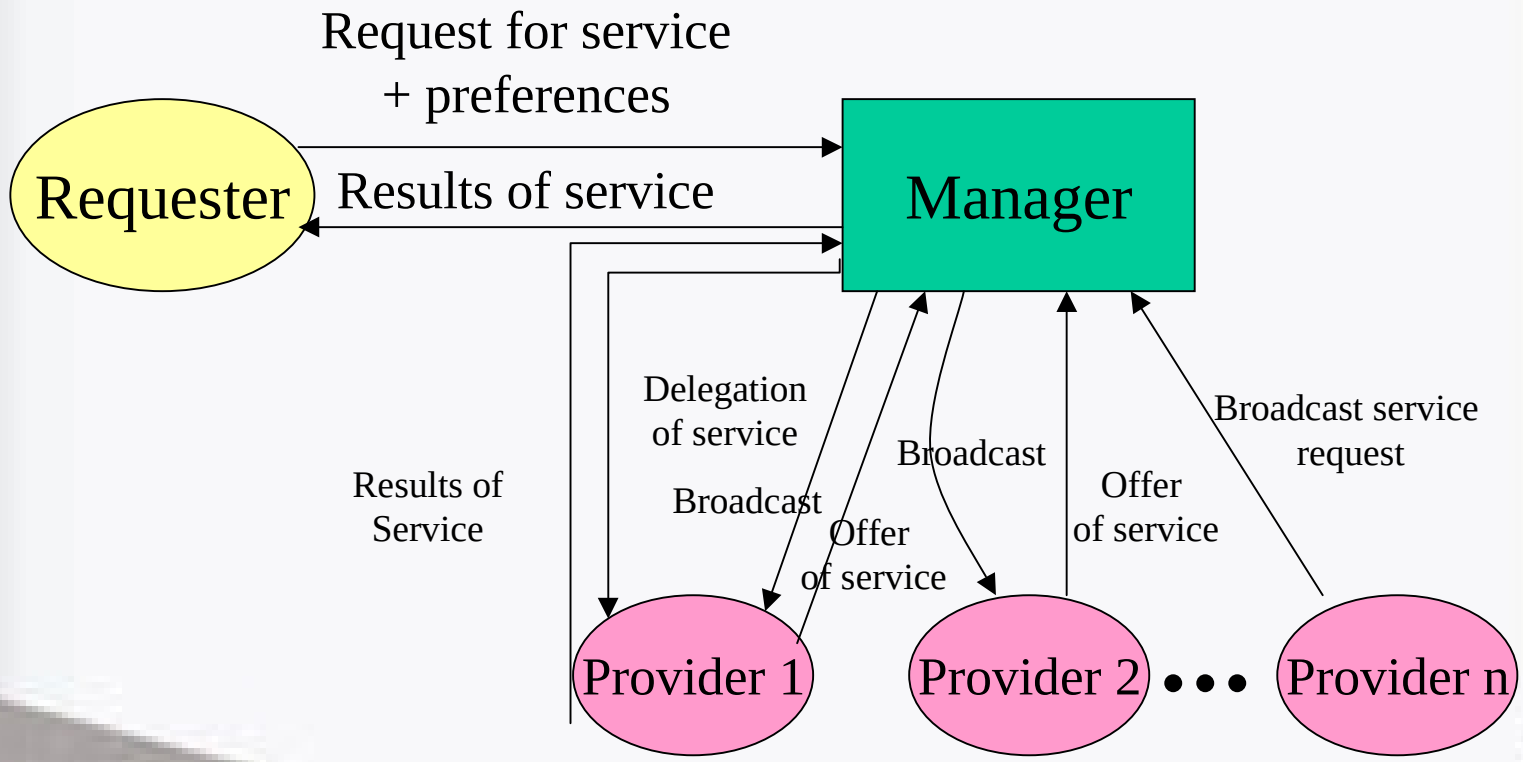
Broker Middle Agent



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Contract Net Protocol

(Composed of Broadcaster and Broker Middle Agents)



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Capability Descriptions and Mediation Services

- Several MAS use different capability descriptions and mediation policies
  - InfoSleuth
    - MCC, Austin, Texas
  - IMPACT
    - *Interactive Maryland Platform for Agents Collaborating Together* - University of Maryland
  - LARKS
    - *Language for Advertising and Request for Knowledge Sharing* – Carnegie Mellon University
  - OAA
    - *Open Agent Architecture* – SRI International

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

InfoSleuth - Middle Agents

- Agents advertise themselves with a Matchmaker (known as "*The Broker*")
  - Performs syntactic and Semantic matching
- Ontology Agents collectively maintain KB of the ontologies used, and return ontological information on demand.

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

InfoSleuth – Capability Description

- Consists of:
  - Agent conversations used to communicate about the service
  - Interface to the service
  - Information a service operates over
  - Semantics of what the service does

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

InfoSleuth – Capability Description

Advertisement

Capability env-subscription
  ontology conversation
  class conversation
    slot conversation-type **in** {*subscribe*}
    slot language **in** {*kqml*}
  …
  ontology environment
  class site
    slot containment
    slot remedial-response
    slot city
    slot state **in** {*Texas*}

Query

Capability subscribe-to-capability
  ontology conversation
  class conversation
    slot conversation-type **in** {*subscribe*}
    slot language **in** {*kqml*}
  …
  ontology environment
  class site
    slot containment
    slot city **in** {*Austin*}
    slot state **in** {*Texas*}

Semantic Web Services

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

InfoSleuth – Capability Matching

- Matching occurs at different levels:
  - Syntactic Matching
    - Based on the syntax of the incoming messages that wrap the query
  - Semantic Matching
    - Matching the internal data structures of the request and capability descriptions
  - Pragmatic Matching
    - Considerations include performance/load of the agent, security requirements, etc.

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

IMPACT - Middle Agents

- Uses a collection of four servers:
  - Yellow Pages Server
    - Basic matchmaking based on weighted verb & noun hierarchies of synonyms
  - Registration Server
    - For agent registration and maintaining agent service index used by Yellow Pages Server
  - Type Server
    - Maintains hierarchical ontology of standard data types & concepts
  - Thesaurus / Human Interface
    - Provides access to the above servers

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

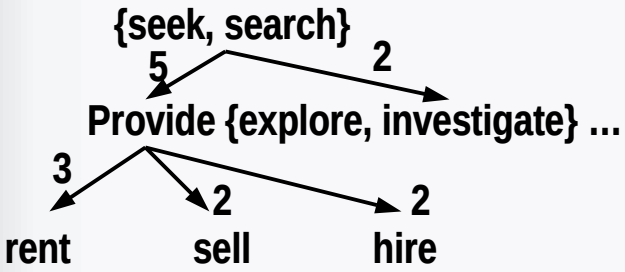
IMPACT – Capability Description

- Consists of:
  - Service name in terms of verb-noun expression
    - *e.g. rent:car(Japanese)*
  - Typed input & output variables
  - Service Attributes
    - *e.g. usage cost, average response time, etc.*

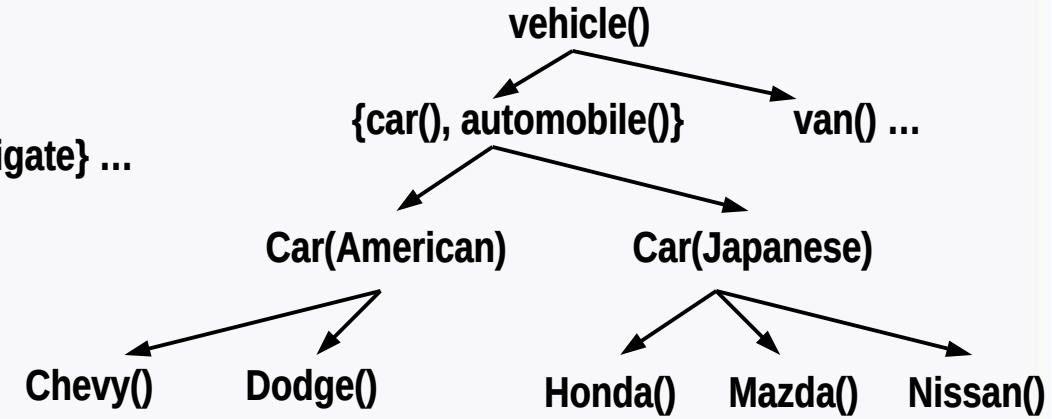
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

IMPACT – Hierarchies & Thesaurus

Verb Hierarchy



Noun Hierarchy



Thesaurus

Word	Synonyms
vehicle()	car, carrier, van …
car(American)	Chevy, Neon, Dodge
car(Japanese)	Honda, Mazda, Nissan…
seek	search, lookup
provide	supply, hand-over

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

IMPACT – Capability Matching

- Services requested using:
  - k-nearest neighbor request
    - *e.g. find me the 6 closest matches to my request*
  - d-range search request
    - *e.g. find me all matches within a distance of 0.95*
- Matching algorithm relies on weighted verb and noun hierarchies.
  - Similarity values for verbs and nouns based on paths of weighted edges in the hierarchies
  - If verb/noun cannot be found in the hierarchy, then a synonym is sought for in the thesaurus
  - Distance function combines similarity values

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

LARKS - Middle Agent

- Matchmaker Middle Agent supports three general types of capability matching:
  - Relaxed Matching
    - Least accurate – returns survey advertisements that may be relevant.
  - Plug-In Matching
    - More accurate match – returns advertisements that have more general capabilities than required by the requestor
  - Exact Match
    - Most restrictive match – where both descriptions are equal (although variables may be renamed)

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

LARKS - Knowledge Bases

- Larks maintains several data/knowledge bases:
  - Local Partial Ontology (Concept Database)
    - Built from combining local ontologies (defined in ITL) found in the advertisement's *ConcDescription* frame
    - Uses a bottom-up approach to find common concepts
  - Advertisement Database
    - Store of the capability advertisements
  - Word Distance Database
    - Maintains Term statistics (e.g. word frequencies) for the profile filter

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

LARKS - Capability Description

- ***Context***
  - Keywords denoting the ontological domain
- ***Data Types***
  - User defined data types found in the interface description
- ***Input & Output***
  - Interface (signature) description in terms of inputs and outputs
- ***InConstraints & OutConstraints***
  - Logical constraints on Input/Output variables (i.e pre & post conditions)
- ***ConcDescriptions***
  - Descriptions of disambiguating terms defined in the concept language ITL
- ***TextDescription***
  - Free Text description of the agent's capabilities

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

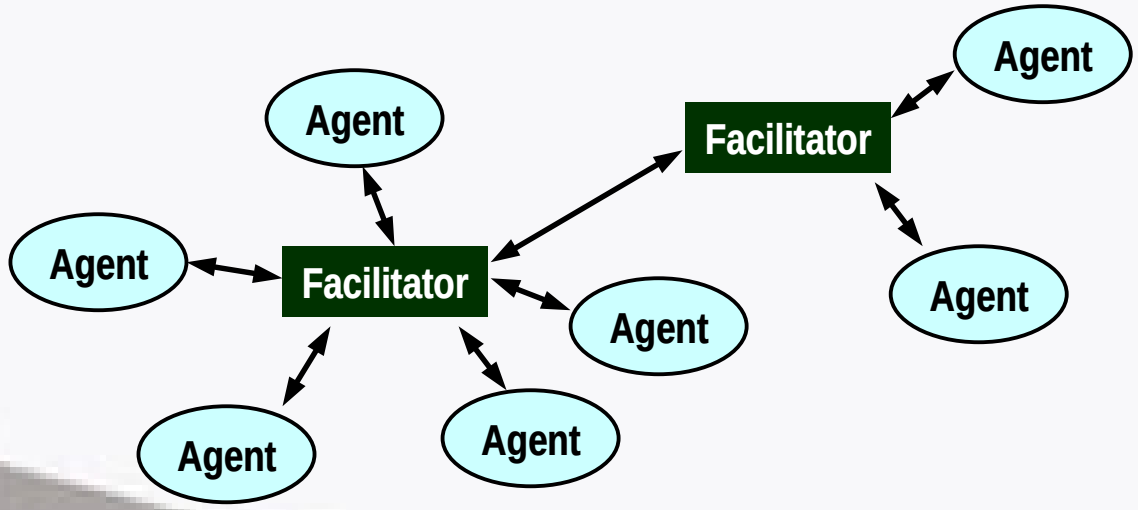
LARKS - Capability Matching

- Context
  - Compares the context specification of two advertisements.
- Profile
  - Uses Information Retrieval metrics to compare advertisements.
- Similarity
  - Compares pairwise inputs, outputs, input & output constraints.
- Signature
  - Checks pairwise signatures according to a given set of data type inferencing rules
    - *e.g integer is subtype of real*
- Constraint
  - Compares pairwise signatures of Input/Output declarations
    - Determines if Provider inputs logically imply Requester inputs
    - Determines if Requester outputs logically imply Provider outputs

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

OAA - Middle Agent

- One or more Domain Independent Facilitators
  - May be organized within a hierarchy
- One or more Domain Dependent Meta-Agents
  - Assist the facilitator in coordinating the activities of the agents



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

OAA - Capability Description

- A PROLOG based Solvable, containing:
  - A Goal Template
    - Logical representation of the service
  - Parameters
    - Options that refine the semantics associated with the solvable
      - e.g. *type* identifies a *data* or *procedure* solvable
  - Permissions
    - Controls access to the service

```
solvable (last_message(email, -MessageID)
          [type(data),single_value(true)],
          [write(true)])
```

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

OAA - Capability Matching

- Agents submit *goals* to the Facilitator
  - (Partially instantiated goals can be thought of as *queries*)
- Prolog unification is used to match *goals* with advertised *solvables*
  - Solvable parameters and request parameters also influence this process
- The *goals* are then sent to the corresponding providers to satisfy (and instantiate if needed).
- Instantiated goals are then returned to the Facilitator, and in turn to the service requester.

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Questions?

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Semantic Web Services

Semantic Web-Enabled Web Services: DAML-S

Acknowledgements to the **DAML-S Web Services Coalition**

CMU: Anupriya, Ankolekar, Massimo Paolucci,
Terry Payne, Katia Sycara

BBN: Mark Burstein

Nokia: Ora Lassila

Stanford KSL: Sheila McIlraith, Honglei Zeng

SRI: Jerry Hobbs, David Martin, Srini Narayanan

[Yale: Drew McDermott & **Manchester:** Ian Horrocks]

Several of these slides courtesy of Sheila McIlraith, Stanford KSL, & David Martin, SRI

Introduction

- **DAML-S**: A **D**ARPA **A**gent **M**arkup **L**anguage for **S**ervices
  - DAML+OIL Ontology for (Web) services
- AI-inspired markup language:
  - tailored to the representational needs of Services
  - expressive power
  - well-defined semantics
  - ontologies support reuse, mapping, succinct markup, ...

http://www.daml.org/services/

DAML-S Objectives

- Provides:
  - An upper ontology for describing properties & capabilities of agents & (Web) services in an unambiguous, computer interpretable markup language.
  - Supporting ontologies for describing service types, security definitions, execution parameters, access and invocation characteristics.
  - Guidelines for advertising, modeling and requesting services.
  - Infrastructure to support the location and invocation of services.

DAML-S Objectives

- Desiderata:
  - an ontology of Web services
  - ease of expressiveness
  - enables automation of service use by agents
  - enables reasoning about service properties and capabilities

DAML-S Objectives

- Model web services using DAML syntax and description-logics
- Achieve semantic interoperability through tight coupling with Semantic Web standards.
- Automate:
  - WS Discovery
  - WS Invocation
  - WS Selection, Composition & Interoperation
  - WS Execution Monitoring

Automating WS Discovery & Selection

- Support tasks such as
  - Find me an airline that can fly me to Sardinia
- Issues:
  - How are service capabilities defined in the advertisement/request?
    - How are constraints specified?
  - How do mediation/discovery services affect the format of the request specification?

Automating WS Invocation

- Support tasks such as
  - Book flight tickets from Alitalia to arrive on the 9th June.
- Issues:
  - What is the format of the messages sent between service provider and service requester?
  - How are values in incoming messages validated to ensure successful invocation?

Automating WS Composition & Interoperation

- Support tasks such as:
  - Arrange travel from Pittsburgh to Civitavecchia via Rome.
- Issues:
  - How are composite tasks defined?
  - How are multi-party interactions mediated?
  - How are tasks decomposed
    - Matching with pre-defined composite models?
    - Planning based on known or discovered services?

Automating WS Execution Monitoring

- Support tasks such as
  - Have the ferry tickets been reserved yet??
- Issues:
  - How does a service monitor and share its current execution status?
  - How are execution monitoring queries defined?
  - How are failures and aborts handled?

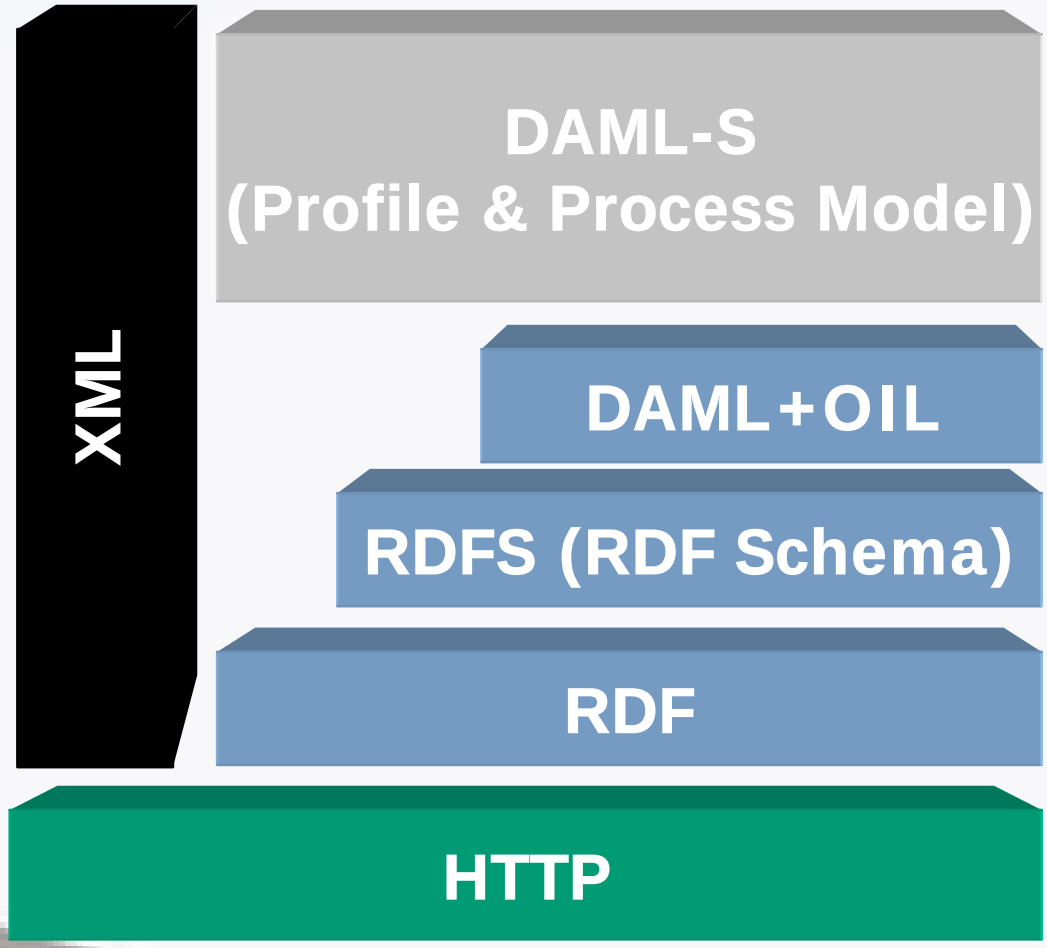
Main Elements of DAML-S

- Utilizing Semantic Web Standards
- Building upon and Augmenting Existing Web Services Standards
- Ontologies
- Supporting Infrastructure
  - Semantic Service Registries
    - DAML-S Matchmaker
  - DAML-S API

Main Elements of DAML-S

- **Utilizing Semantic Web Standards**
- Building upon and Augmenting Existing Web Services Standards
- Ontologies
- Supporting Infrastructure
  - Semantic Service Registries
    - DAML-S Matchmaker
  - DAML-S API

Building on Existing Semantic Web Standards



Semantic Web Services

RDF: Resource Description Framework

- RDF built using
  - XML syntax
  - qualified Universal Resource Identifiers (URI)
  - Namespaces etc.
- Models Meta-Data about resources on the Web using *subject/predicate/object* triples
  - Triples define the relationship or *predicate* between two entities (the *subject* and *object*)

RDF Concepts

- Three fundamental concepts:
  - Resources
    - All entities described by RDF expressions, named by a qualified URI:

      rdf:resource="http://www.cs.cmu.edu/~terryp/index.html"

      rdf:resource="http://www.tac.org/2001event.rdf#PainInNEC"
  - Properties
    - Define characteristic of a resource

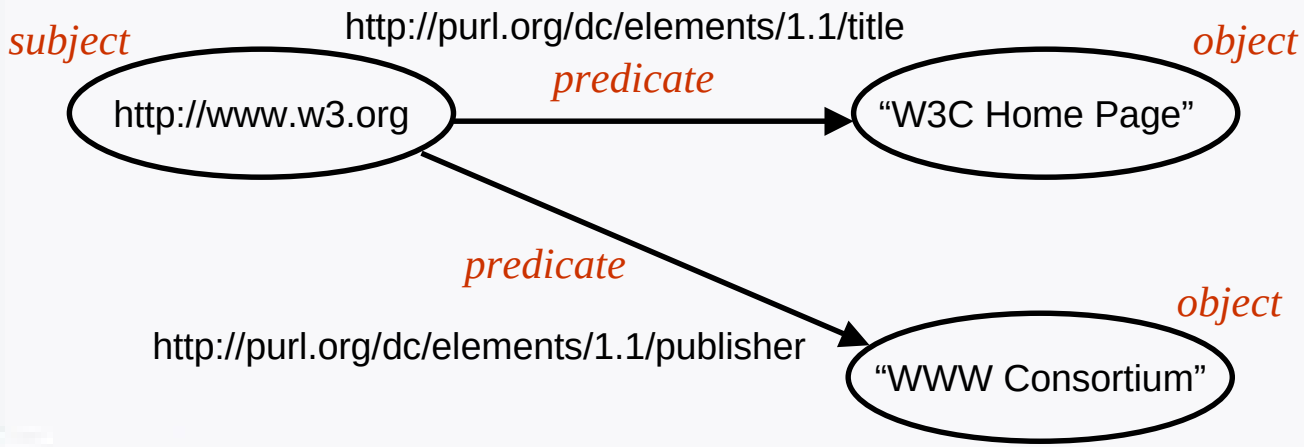
      <rdf:Description about="http://www.w3.org">

      <dc:title>W3C Home Page</s:title>

      </rdf:Description>
  - Statements
    - Resources that reify *subject/predicate/object* triples

RDF Graphs

- The *subject/predicate/object* triples found in an RDF document form a graph:



RDF Schema

- RDF Schema defines new RDF vocabulary, extending beyond definitions of triples. For example:
  - rdfs:Class
    - resources denoting a set of resources, by means of the property rdfs:type
  - rdfs:subClassOf
    - Used to define class hierarchies.
  - rdfs:domain & rdfs:range
    - Define restrictions on the resources that have a given property (domain) and the set of valid values for that property (range)

RDF Schema (example)

```
<rdfs:Class
      rdf:about="http://xmlns.com/foaf/0.1/Person"
      rdfs:label="person"
      rdfs:comment="an foaf subclass of wordnet person">

      <rdfs:subClassOf
      rdf:resource="http://xmlns.com/wordnet/1.6/Person" />
      <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/" />
</rdfs:Class>
<rdf:Property
      rdf:about="http://xmlns.com/foaf/0.1/surname"
      rdfs:label="Surname"
      rdfs:comment="The surname of some agent.">

      <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person" />
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal" />
      <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/" />
 </rdf:Property>
```

From RDF to DAML+OIL

- DAML+OIL extends RDF statements to provide a rich descriptive logic language
  - Provides restrictions and additional notations on properties
    - Cardinality restrictions
    - Notations include inverseOf, Transitivity, etc
  - Provides additional properties for class definitions
    - Disjoint-with, complement-Of, intersectionOf, etc

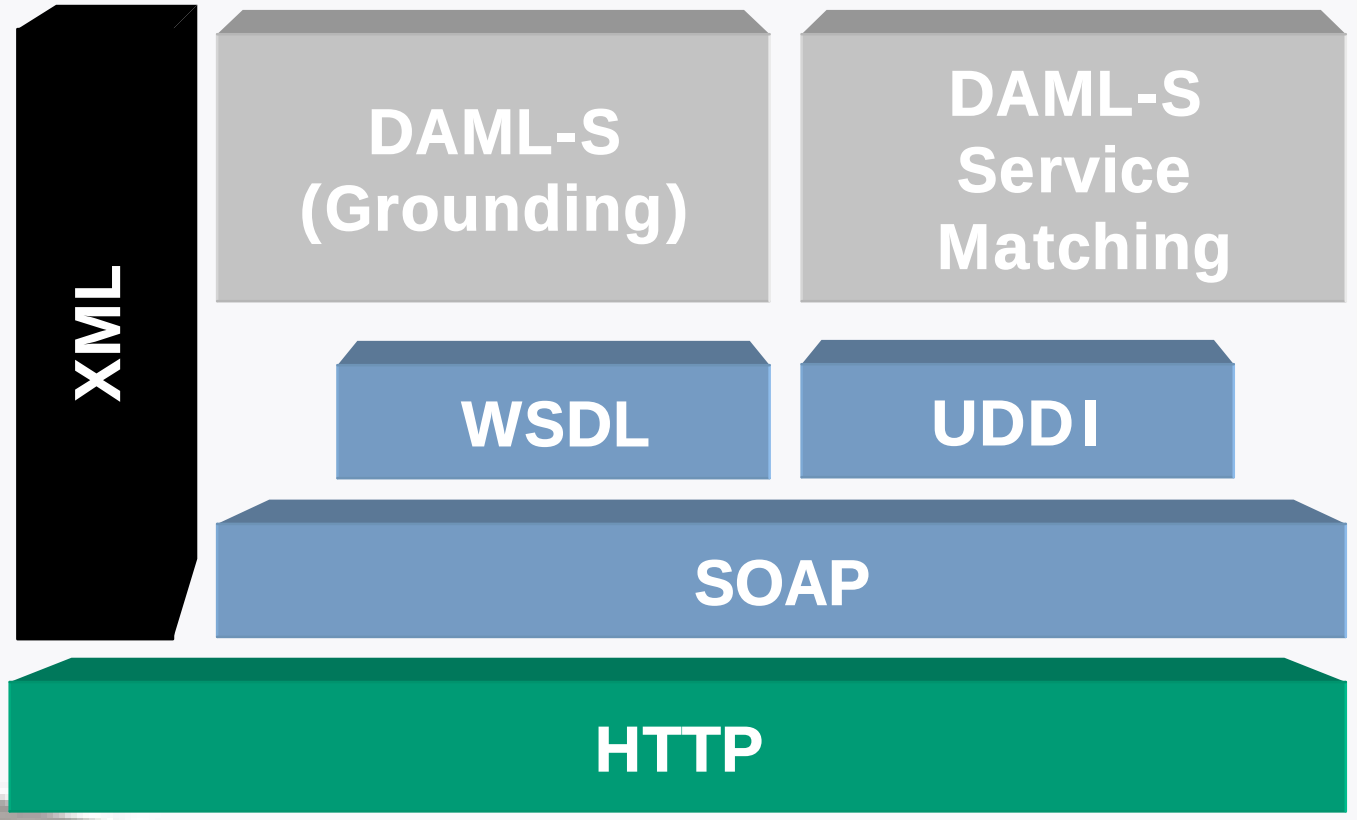
DAML+OIL will be succeeded by the emerging Web Ontology language OWL

http://www.w3.org/2001/sw/WebOnt/

Main Elements of DAML-S

- Utilizing Semantic Web Standards
- **Building upon and Augmenting Existing Web Services Standards**
- Ontologies
- Supporting Infrastructure
  - Semantic Service Registries
    - DAML-S Matchmaker
  - DAML-S API

Building on Existing Web Services Standards



Semantic Web Services

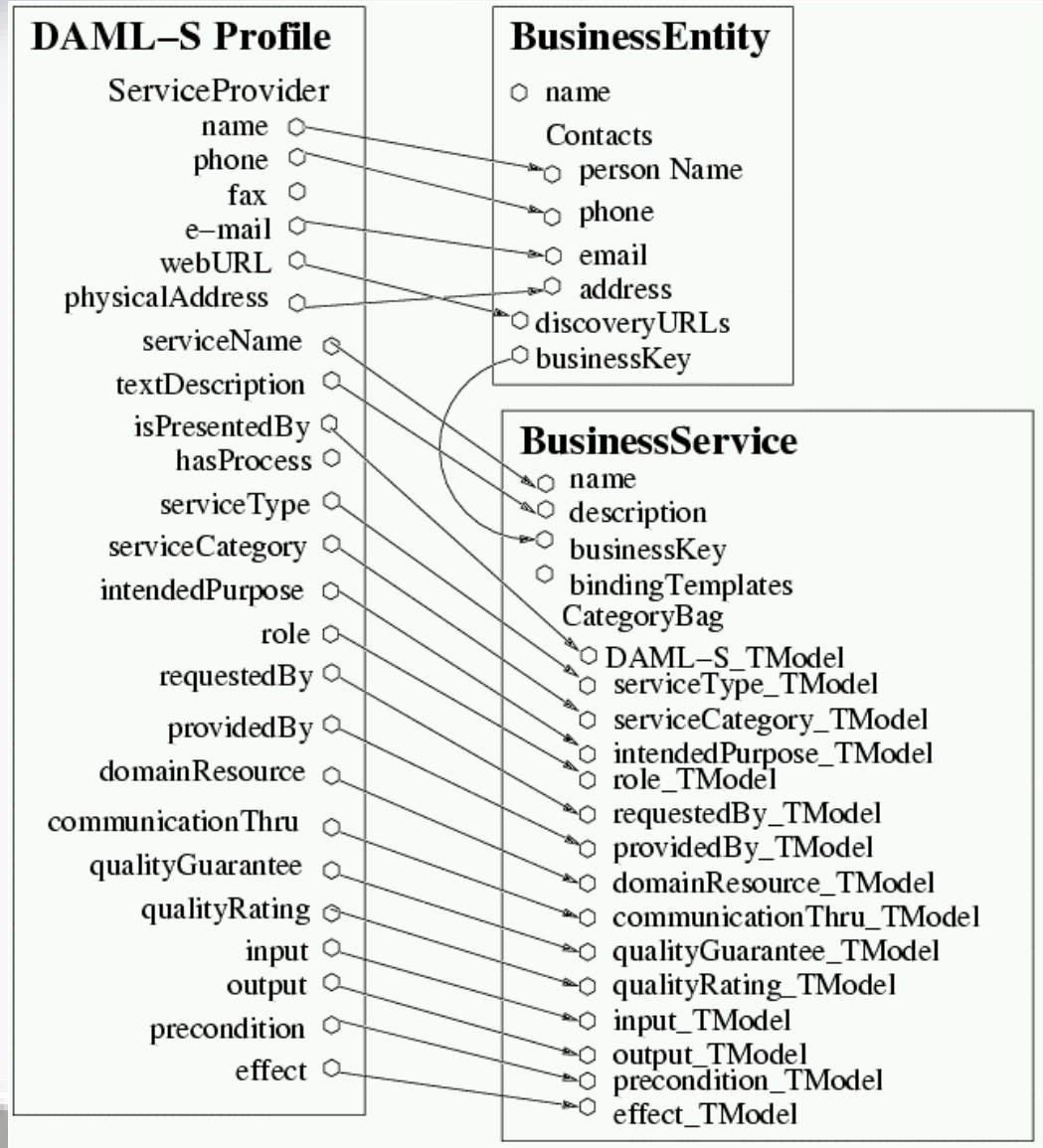
Web Services Description Language

- Extensible XML based description of web based services and how they are invoked
- Allow for several interfaces using different communication languages
  - E.g. http, SOAP, etc.
- Supports simple transactions (operations)
  - E.g. request-response, solicit-response, etc.
- **Used within DAML-S to describe interface and simple transaction protocol for atomic services.**

UDDI: Universal Discovery, Description and Integration

- Provides registries and simple lookup services for:
  - White Pages
  - Yellow Pages
  - Green Pages
- **UDDI T-models have been defined to support UDDI-based DAML-S Profile descriptions**
- **DAML-S Matchmaker extends UDDI server to provide semantic service-capability requests.**

DAML-S T-Models for UDDI



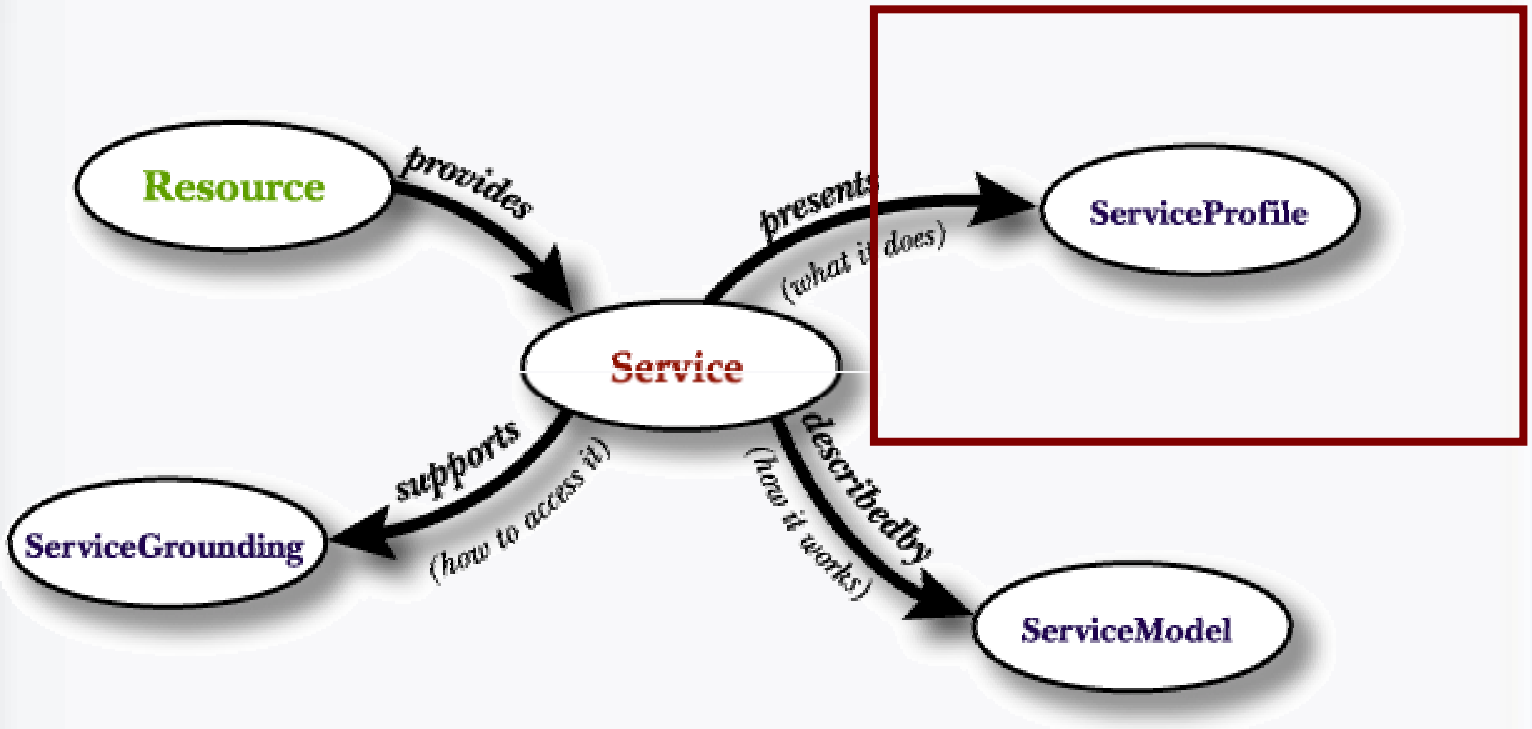
Main Elements of DAML-S

- Utilizing Semantic Web Standards
- Building upon and Augmenting Existing Web Services Standards
- **Ontologies**
- Supporting Infrastructure
  - Semantic Service Registries
    - DAML-S Matchmaker
  - DAML-S API

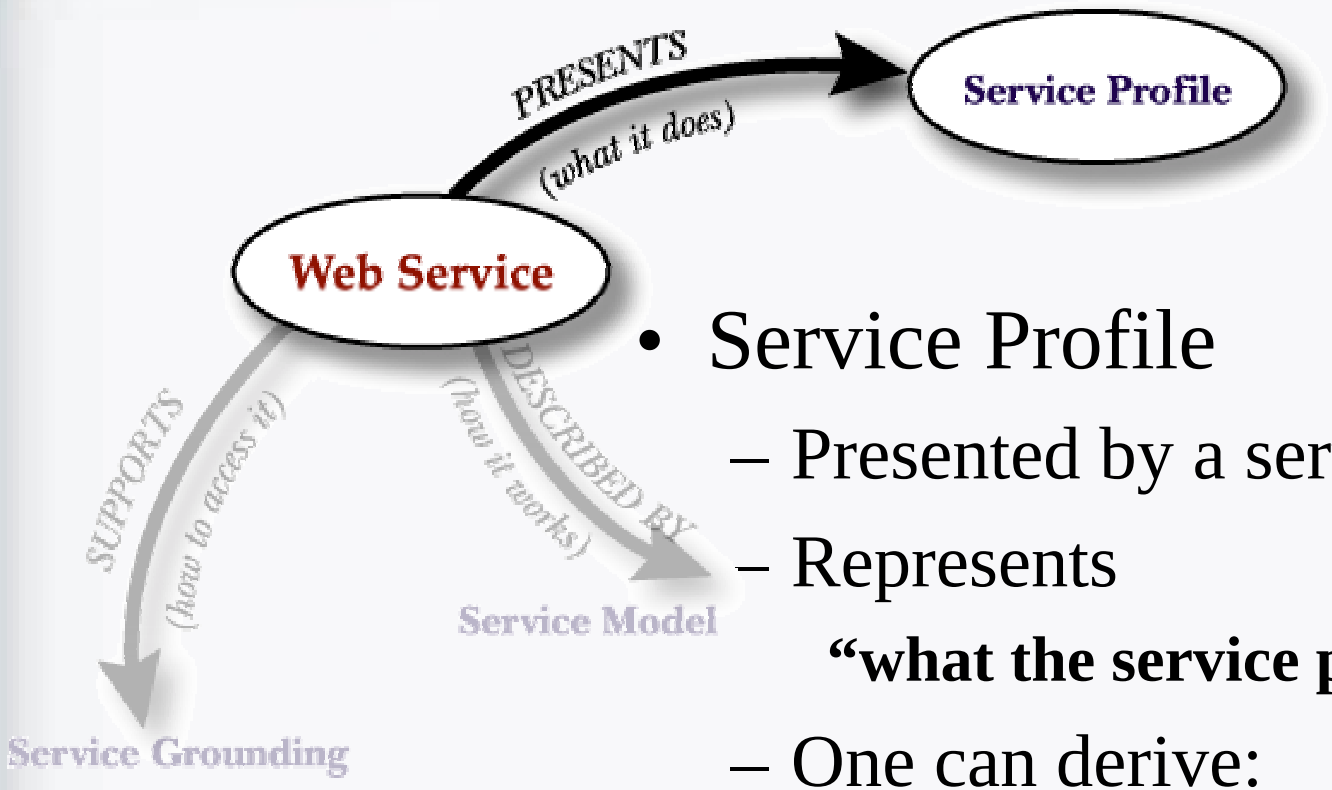
DAML-S Ontologies

- DAML-S defines ontologies for the construction of service models:
  - Service Profiles
  - Process Models
  - Service Grounding
- DAML-S defines ontologies to support and augment Service Profiles
- Ontologies (e.g. service ontologies) and classification schemes (e.g.NAICS) are used to construct Service Profiles and Process Models

DAML-S Service Models



Presenting Service Profiles



- Service Profile
  - Presented by a service.
  - Represents

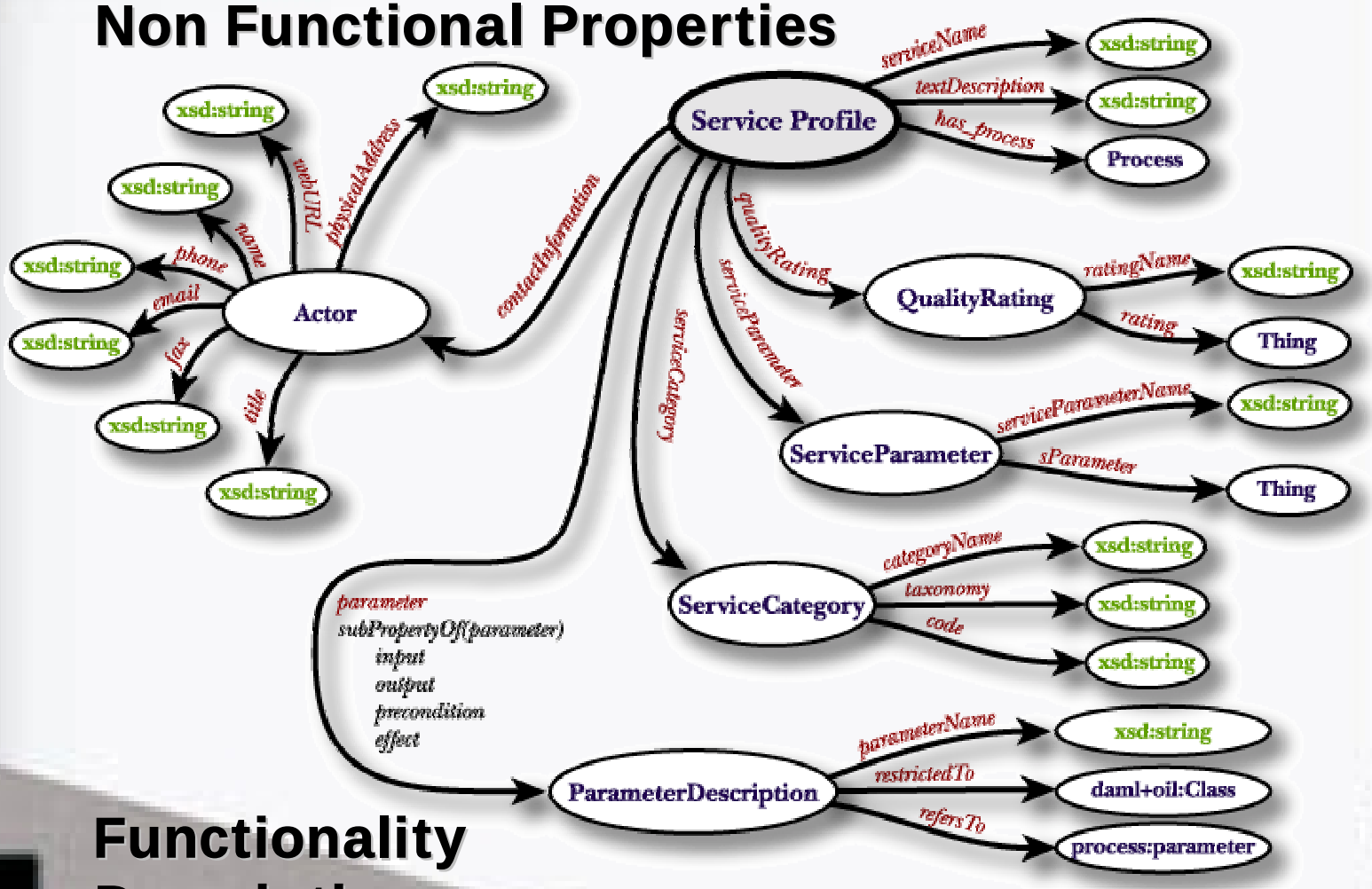
    “what the service provides”
  - One can derive:
    - **Service Advertisements**
    - **Service Requests**

DAML-S Service Profile (Overview)

- High-level description of a service and its provider
  - description of service (human readable)
  - specification of functionalities service provides
  - functional attributes (requirements and capabilities)
- Profile used for
  - populating service registries
  - automated service discovery
  - Matching of capability advertisements to requests

DAML-S Service Profile

Non Functional Properties

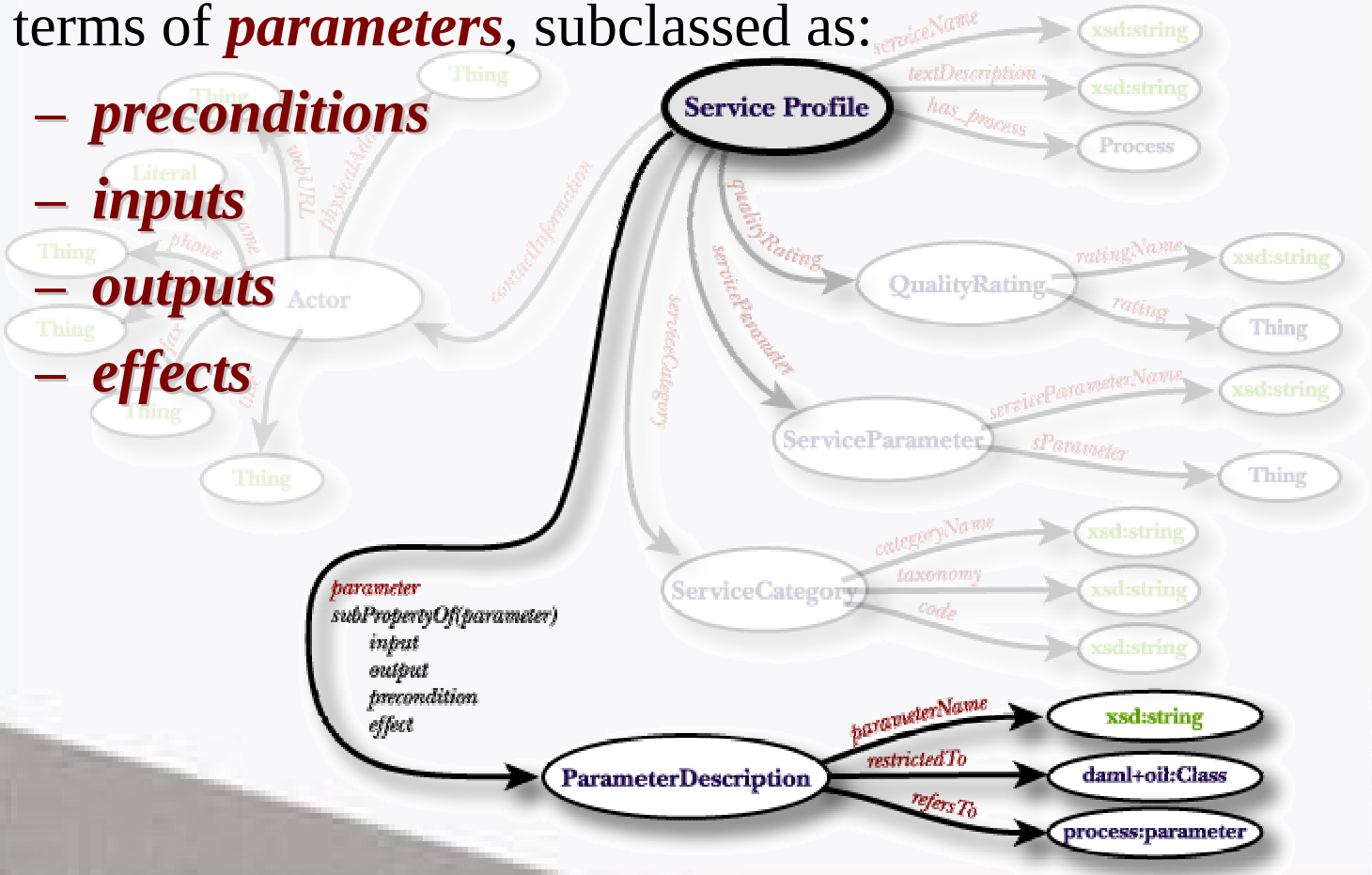


Functionality Description

DAML-S Service Profile

Functionality Description

- Functional Specification of what the service provides in terms of ***parameters***, subclassed as:
  - ***preconditions***
  - ***inputs***
  - ***outputs***
  - ***effects***



DAML-S Service Profile
Functionality Description

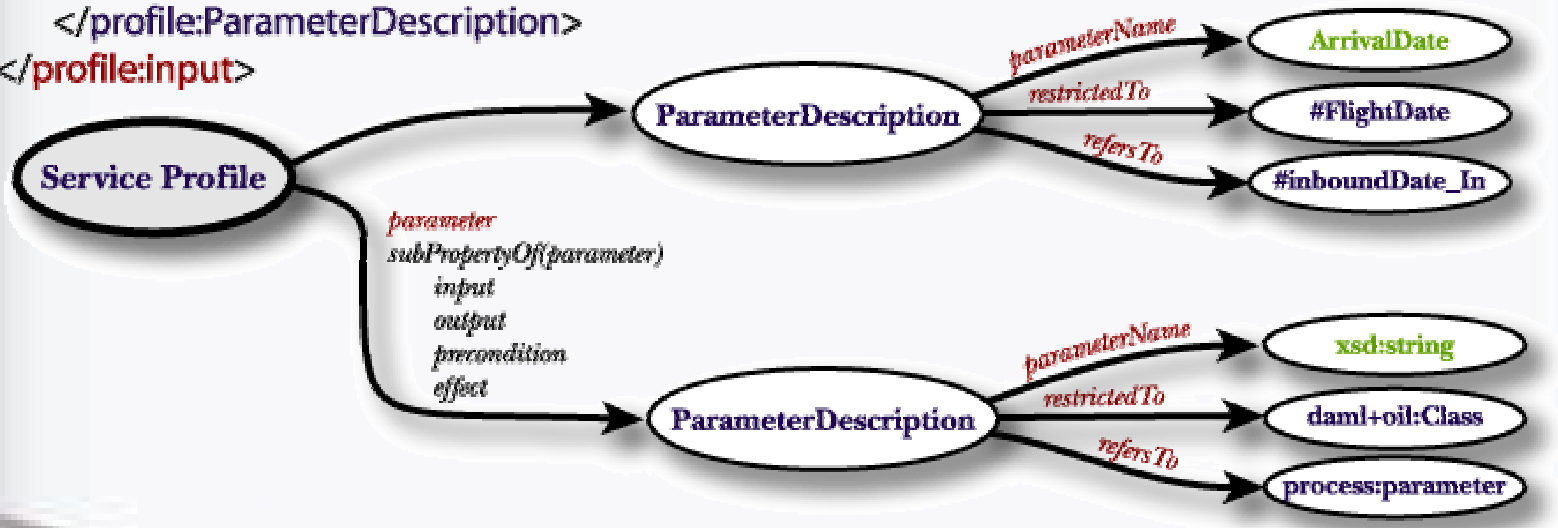
- ***Preconditions***
  - Set of conditions that should hold prior to the service being invoked
- ***Inputs***
  - Set of necessary inputs that the requester should provide to invoke the service
- ***Outputs***
  - Results that the requester should expect after interaction with the service provider is completed
- ***Effects***
  - Set of statements that should hold true if the service is invoked successfully.
  - Often refer to real-world effects
    - Package being delivered
    - Credit card being debited

DAML-S Service Profile
Functionality Description

- An *Input/Output/Precondition/Effect* parameter has three properties:
  - ***parameterName***: the name of the parameter
  - ***restrictedTo*** : a resource corresponding to some RDF/DAML property type within some ontology (i.e. the range of a parameter instance)
  - ***refersTo*** : the corresponding parameter defined within the process model

DAML-S Service Profile
Functionality Description

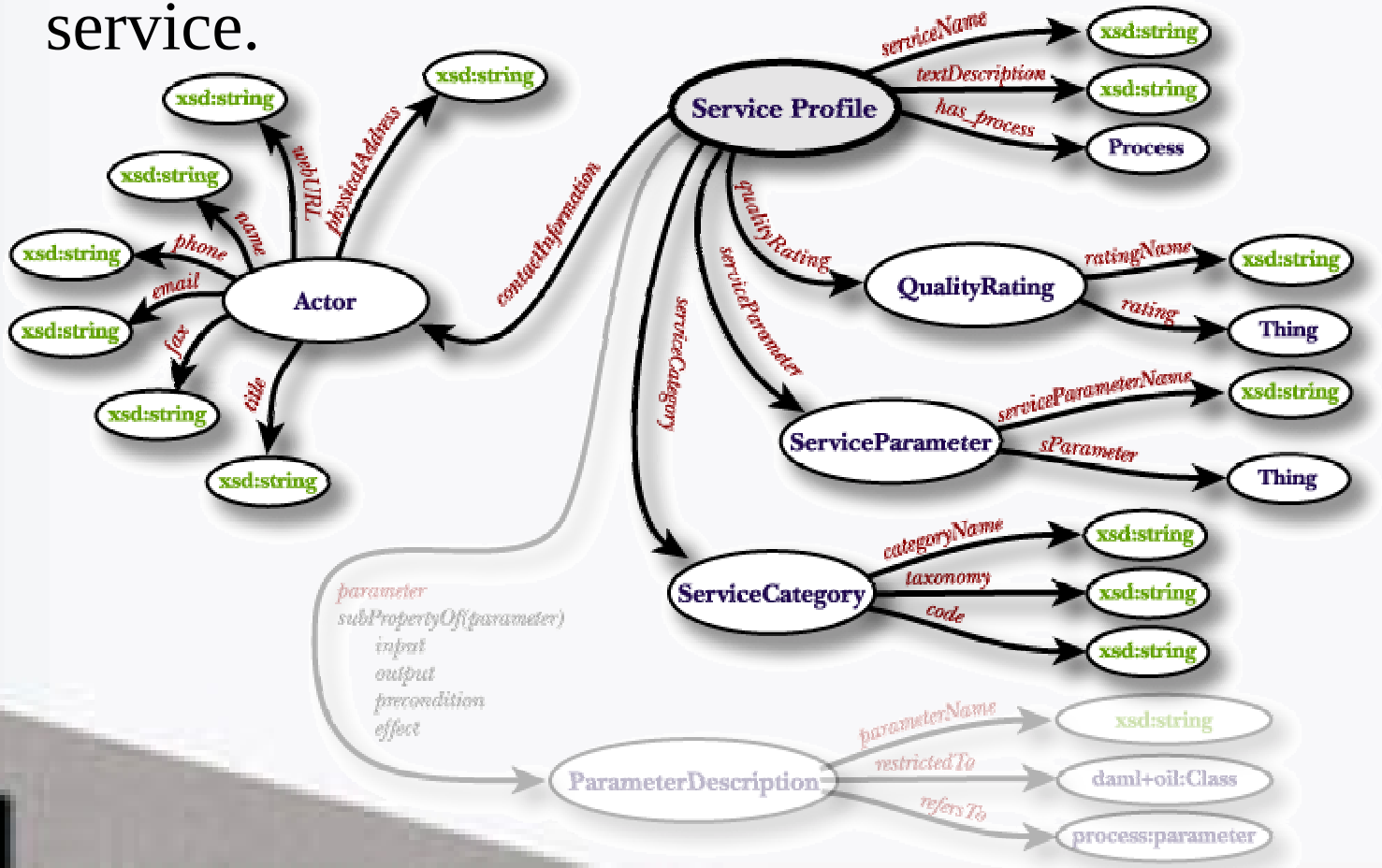
```
<profile:input>
    <profile:ParameterDescription rdf:ID="ArrivalDate">
        <profile:parameterName>ArrivalDate</profile:parameterName>
        <profile:restrictedTo rdf:resource="http://www..../concepts.daml#FlightDate" />
        <profile:refersTo rdf:resource="http://www..../BravoAirProcess.daml#inboundDate_In" />
    </profile:ParameterDescription>
</profile:input>
```



DAML-S Service Profile

Non Functional Properties

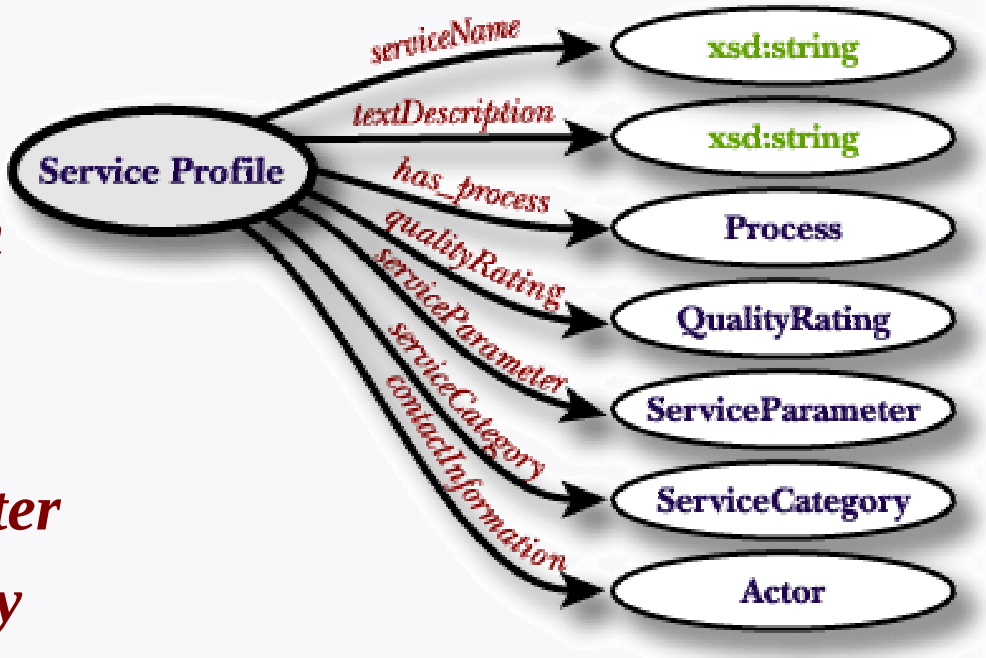
- Provide supporting information about the service.



DAML-S Service Profile

Non Functional Properties

- These include
  - ***serviceName***
  - ***textDescription***
  - ***has_process***
  - ***qualityRating***
  - ***serviceParameter***
  - ***serviceCategory***
  - ***contactInformation***



DAML-S Service Profile

Non Functional Properties - Actor

```
<profile:contactInformation>
    <profile:Actor rdf:ID="BravoAir-reservation">
        <profile:name>BravoAir Reservation department</profile:name>
        <profile:title>Reservation Representative</profile:title>
        <profile:phone>412 268 8780</profile:phone>
        <profile:fax>412 268 5569</profile:fax>
        <profile:email>Bravo@Bravoair.com</profile:email>
        <profile:physicalAddress>
            Airstrip 2, Teetering Cliff Hights, Florida 12321, USA
        </profile:physicalAddress>
        <profile:webURL>http://www. .../BravoAir.html</profile:webURL>
    </profile:Actor>
</profile:contactInformation>
```



Semantic Web Services

DAML-S Service Profile

Non Functional Properties - QualityRating

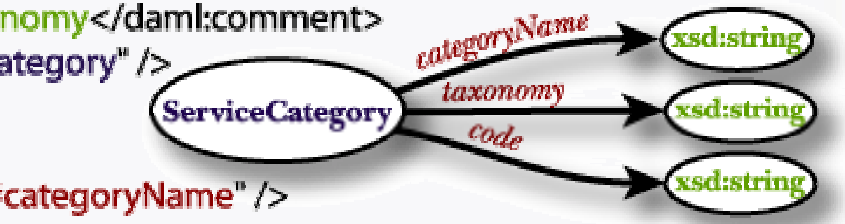


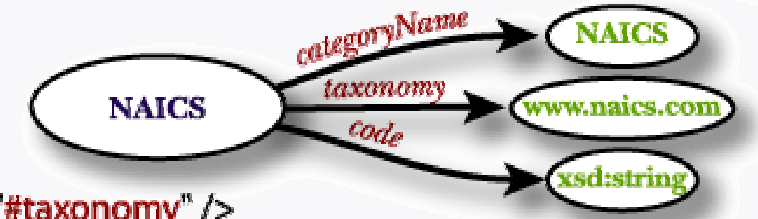

```
<daml:Class rdf:ID="DAndBRating">
    <rdfs:subClassOf rdf:resource="#QualityRating" />
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#ratingName" />
            <daml:toValue rdf:resource="Dun and Bradstreet Rating" />
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```

DAML-S Service Profile

Non Functional Properties – ServiceCategory

```
<daml:Class rdf:ID="NAICS">
    <daml:comment>Hook to the NAICS taxonomy</daml:comment>
    <rdfs:subClassOf rdf:resource="#ServiceCategory" />
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#categoryName" />
            <daml:hasValue>NAICS</daml:hasValue>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#taxonomy" />
            <daml:hasValue>www.naics.com</daml:hasValue>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```





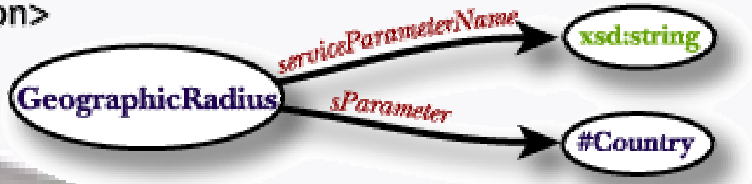
DAML-S Service Profile

Non Functional Properties – ServiceParameter

```
<daml:Class rdf:ID="ResponseTime">
    <rdfs:subClassOf rdf:resource="#ServiceParameter" />
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#sParameter" />
            <daml:toClass rdf:resource="#Duration" />
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```




```
<daml:Class rdf:ID="GeographicRadius">
    <rdfs:subClassOf rdf:resource="#ServiceParameter" />
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#sParameter" />
            <daml:toClass rdf:resource="http://www..../Country.daml#Country" />
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```



Semantic Web Services

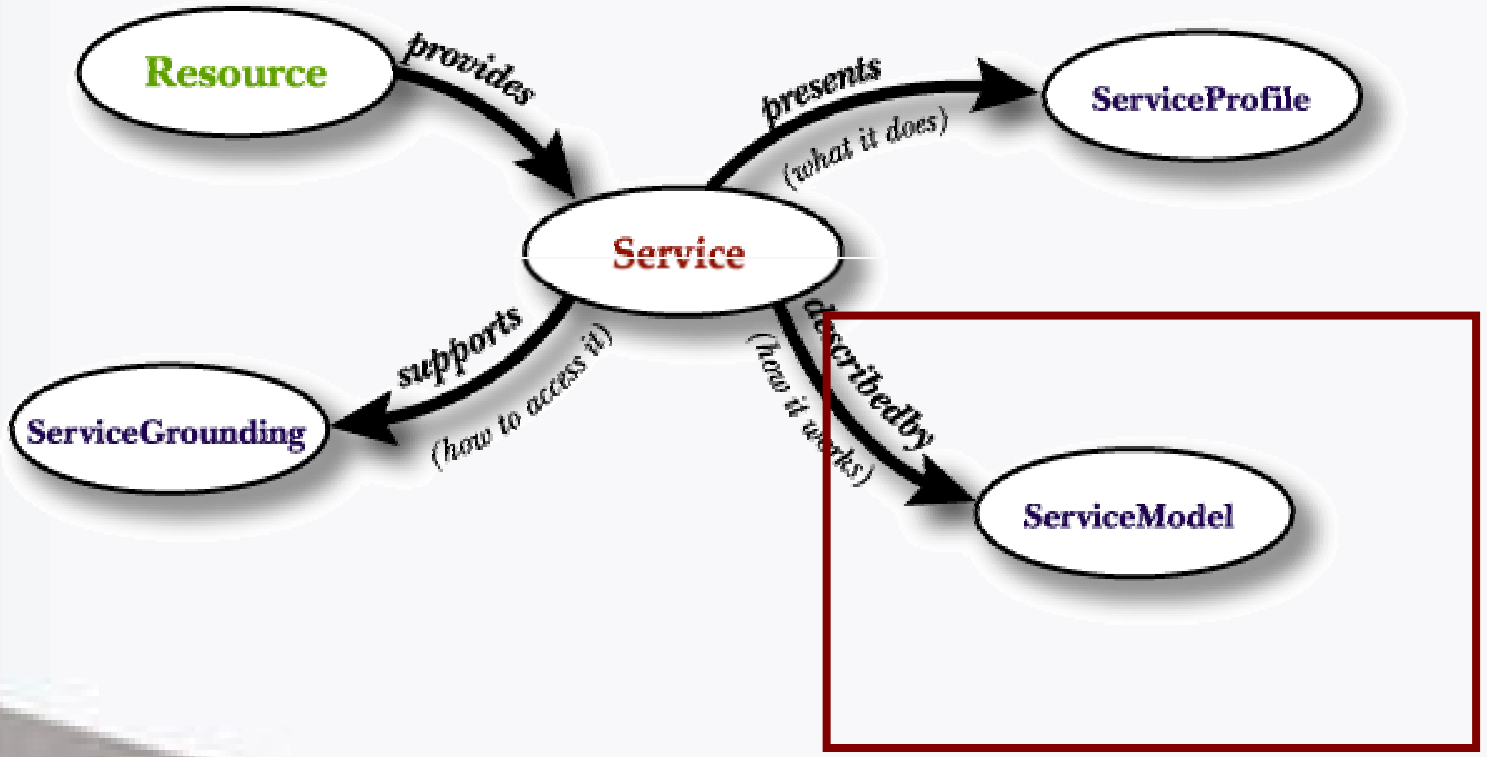
DAML-S Service Ontologies

- Ontologies of service types
  - For example:
    - Reporting Services (e.g. weather information)
    - Directory Services (e.g. yellow pages)
    - Manufacturing or Retail Services, etc.
- Classification Schemes for Products
  - E.g. NAICS (North American Industry Classification System)

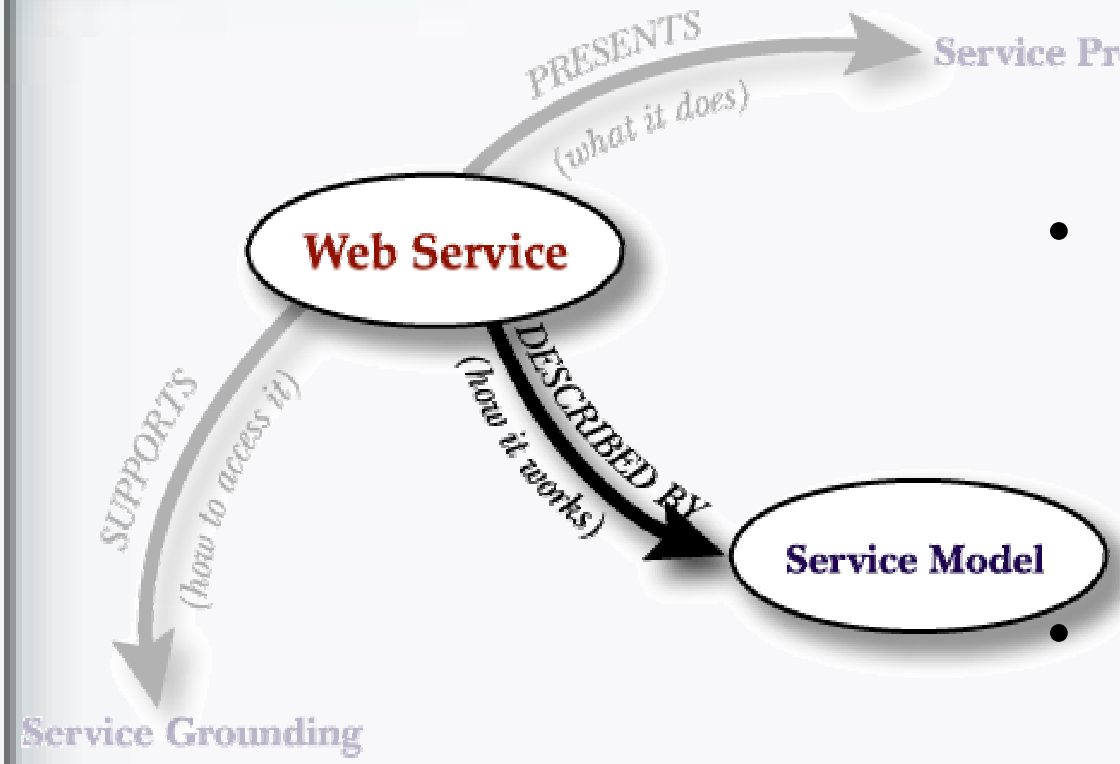
DAML-S Service Ontologies

- Ontologies used to augment service requests:
  - Industry ratings to denote quality guarantees etc.
  - Descriptions such as *cheapest*, *best*, *fastest*, etc.
  - Security parameters to describe types of security and verifications used
  - Ontologies for transaction contracts, such as cost per invocation, guaranteed response by, etc.

DAML-S Service Models



Describing Service Models

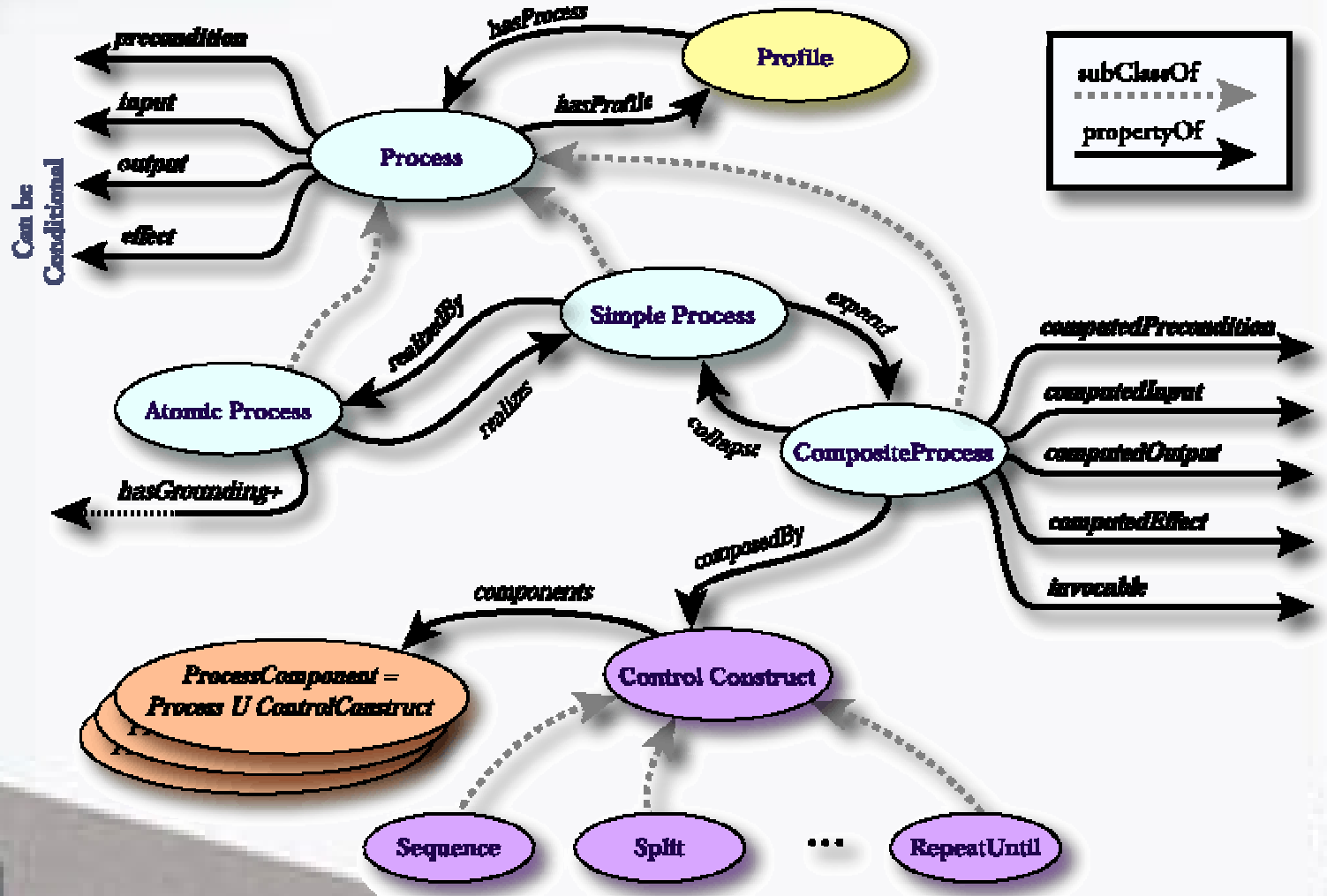


- Service Process
  - Describes how a service works.
- Facilitates
  - (automated) Web service invocation
  - composition
  - interoperation

DAML-S Service Model (Overview)

- Service Model allows a service requester
  - to determines whether the service meets its needs;
  - to compose service descriptions from multiple services to perform a specific task;
  - during the course of the service invocation, to coordinate the activities of the different participants;
  - to form expectations as to the execution of the service.

DAML-S Service Model (Overview)



Anatomy of a DAML-S Service Model

- Processes are conceived as:
  - Atomic
  - Simple
  - Composite
- Associated with each service is a set of:
  - Inputs
  - Outputs
  - Preconditions
  - Effects
- Invocable processes have an associated grounding:
  - Includes WSDL description to model:
    - Operation
    - Message formats
    - Ports & Bindings

Anatomy of a DAML-S Service Model

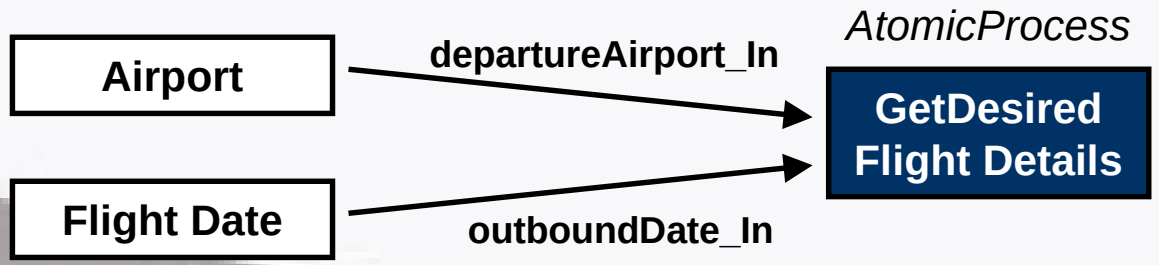
- Composite processes are compositions of simple or other composite processes in terms of constructs:
  - Sequence
  - if-then-else
  - Fork
  - Etc.
- Data flow and Control flow should be described for each composite service
- A black box and glass box view may be given of each composite service

Atomic Process Example

```
<!– Atomic Pss Definition - GetDesiredFlightDetails  -->
  <rdfs:Class rdf:IDroce="GetDesiredFlightDetails">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/Process#AtomicProcess" />
  </rdfs:Class>

<!– (sample) Inputs used by atomic process GetDesiredFlightDetails -->
  <rdf:Property rdf:ID="departureAirport_In">
    <rdfs:subPropertyOf rdf:resource="http://www.daml.org/Process#input" />
    <rdfs:domain rdf:resource="#GetDesiredFlightDetails" />
    <rdfs:range rdf:resource="http://www.daml.ri.cmu.edu/ont/
            DAML-S/concepts.daml#Airport" />
  </rdf:Property>

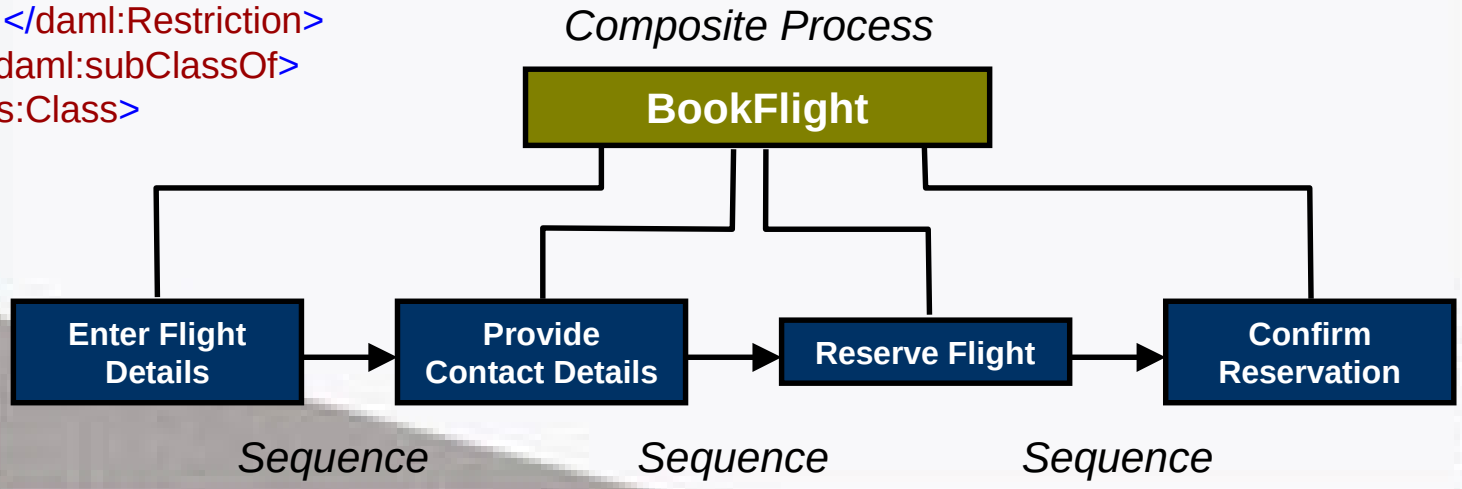
  <rdf:Property rdf:ID="outbounDate_In">
  <rdfs:subPropertyOf rdf:resource="http://www.daml.org/Process#input" />
    <rdfs:domain rdf:resource="#GetDesiredFlightDetails" />
    <rdfs:range rdf:resource="http://www.daml.ri.cmu.edu/ont/
          DAML-S/concepts.daml#FlightDate" />
  </rdf:Property>
```



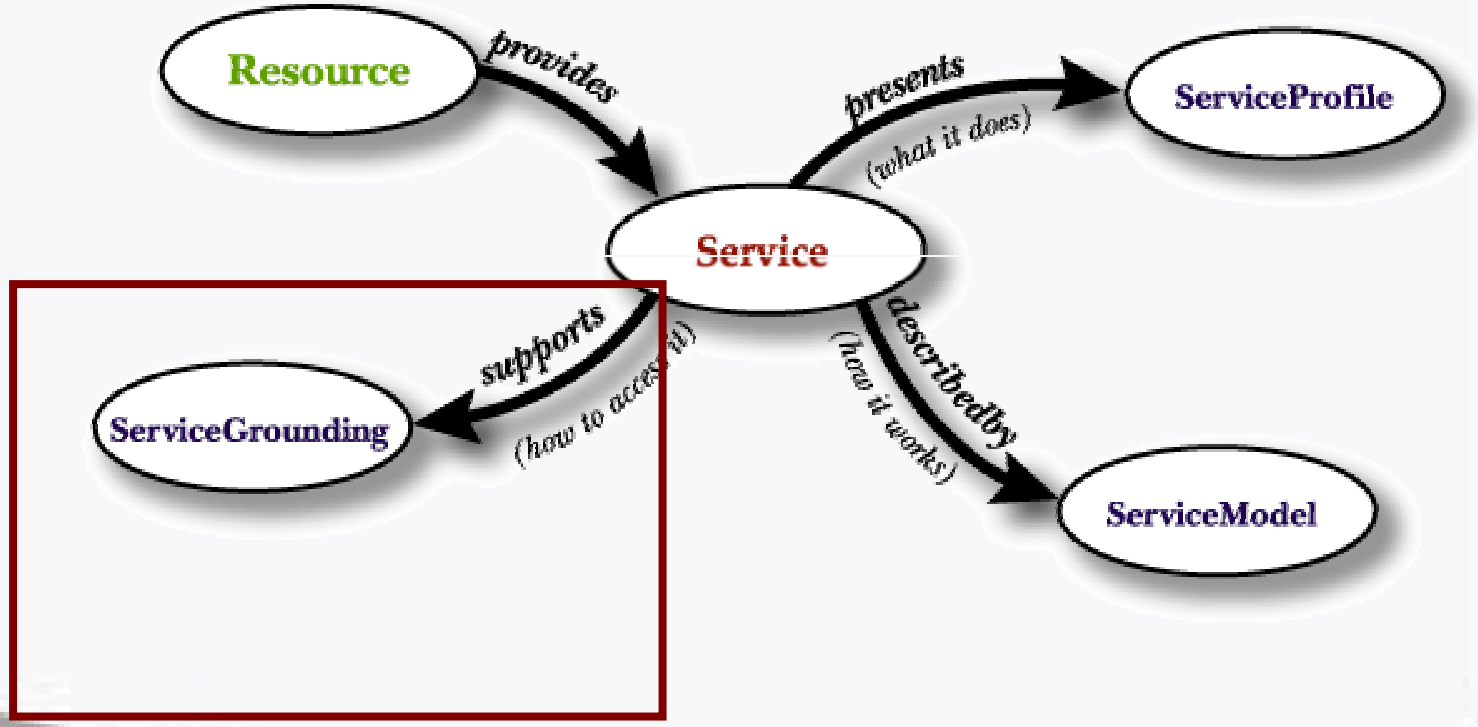
Semantic Web Services

Composite Process Example

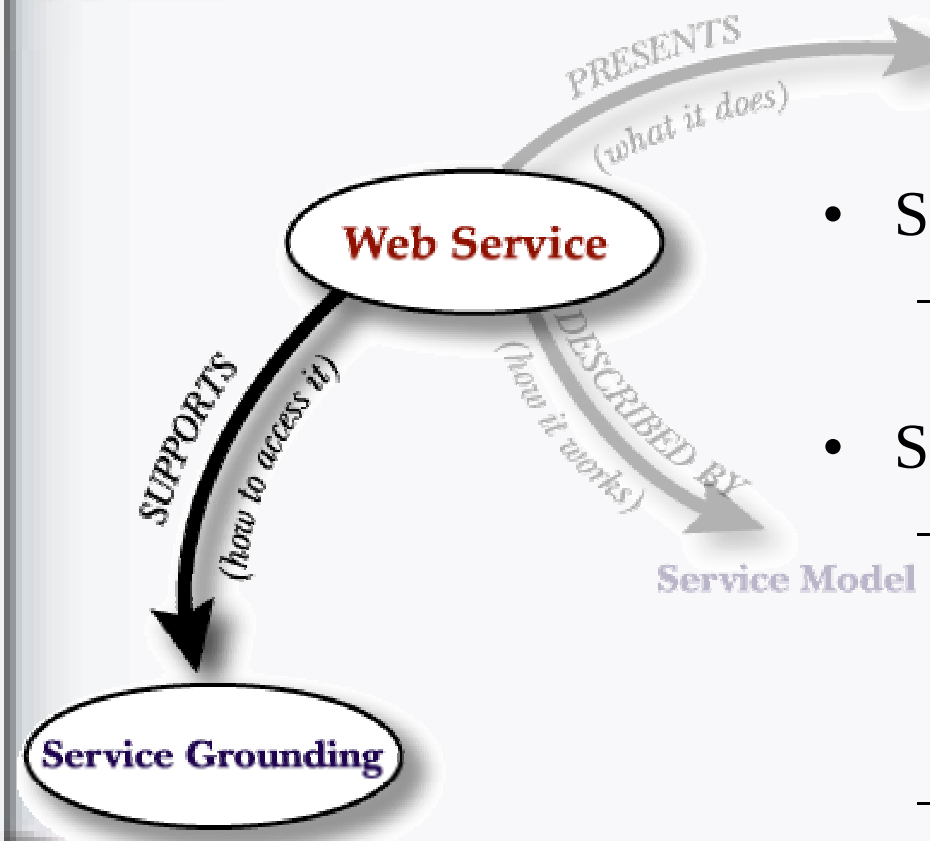
```
<rdfs:Class rdf:ID="BookFlight">
   <rdfs:subClassOf rdf:resource="#CompositeProcess" />
   <rdfs:subClassOf rdf:resource="http://www.daml.org/Process#Sequence" />
   <daml:subClassOf>
      <daml:Restriction>
         <daml:onProperty
            rdf:resource="http://www.daml.org/Process#components" />
         <daml:toClass>
            <daml:subClassOf>
               <daml:unionOf rdf:parseType="daml:collection">
                  <rdfs:Class rdfs:about="#ReserveFlight" />
                  <rdfs:Class rdfs:about="#ConfirmReservation" />
               </daml:unionOf>
            </daml:subClassOf>
         </daml:toClass>
      </daml:Restriction>
   </daml:subClassOf>
</rdfs:Class>
```



DAML-S Service Models



Supporting a Service Grounding

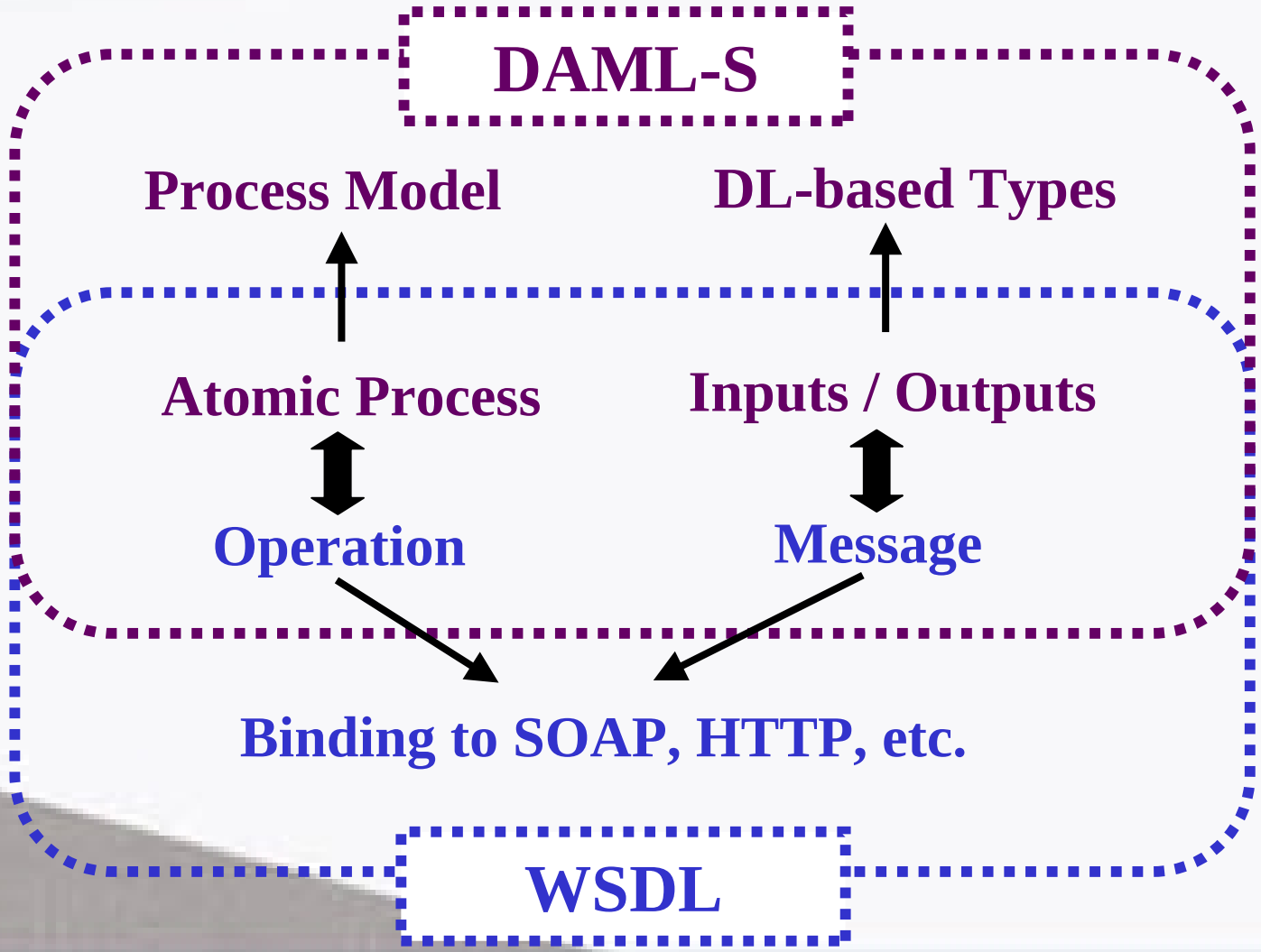


- Service Process
  - Provides a specification of service access information.
- Specifies:
  - communication protocols, transport mechanisms, agent communication languages, etc.
  - E.g., SOAP, HTTP forms, KQML, OAA ACL, Java RMI, RPC, etc.

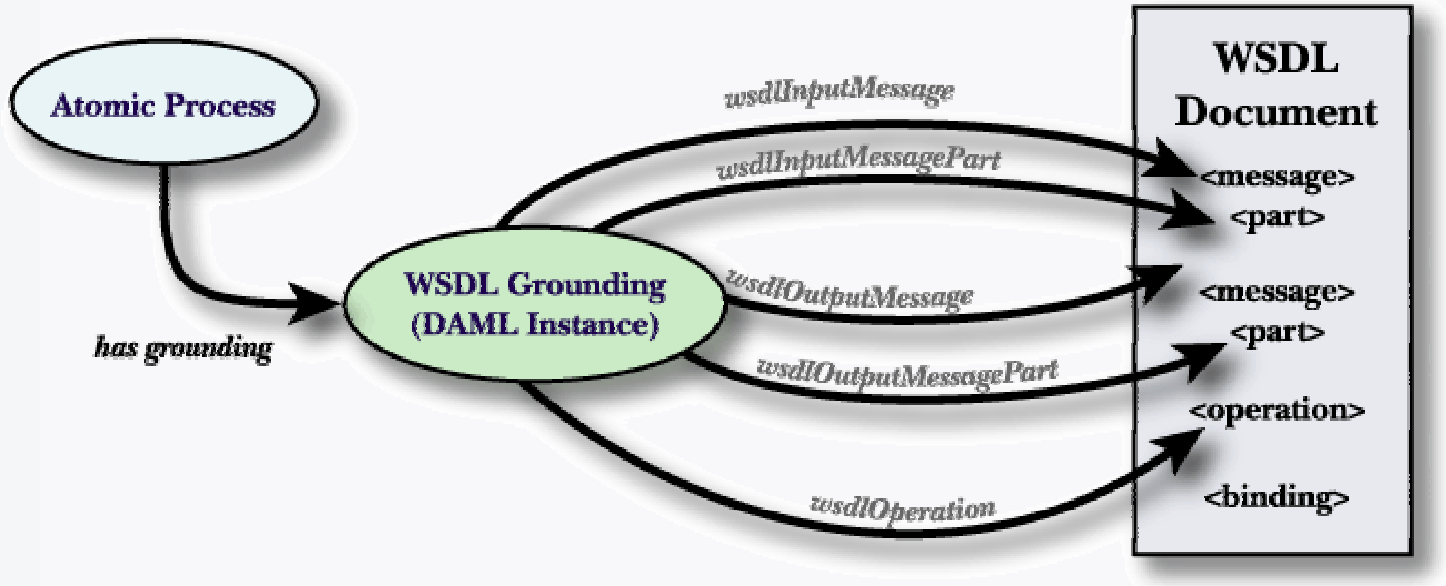
Service Grounding: "How to access it"

- *How to access* the service
- Implementation-specific
- Message formatting, transport mechanisms, protocols, serializations of types
- Service Model + Grounding give everything needed for using the service

DAML-S / WSDL Binding



DAML-S / WSDL Mapping



Main Elements of DAML-S

- Utilizing Semantic Web Standards
- Building upon and Augmenting Existing Web Services Standards
- Ontologies
- **Supporting Infrastructure**
  - Semantic Service Registries
    - DAML-S Matchmaker
  - DAML-S API

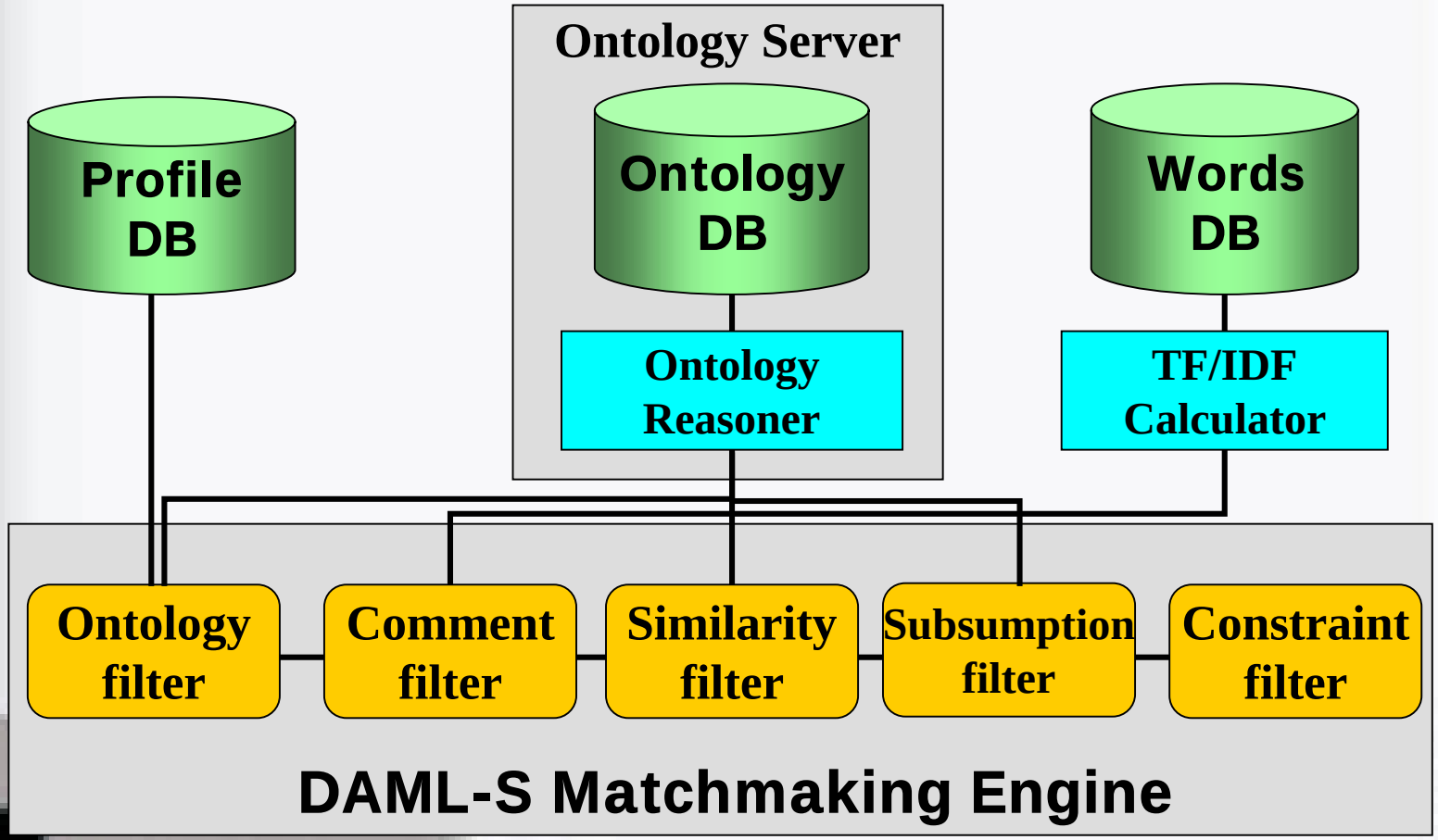
Supporting Infrastructure

- DAML-S Matchmaker
  - Service Registry for DAML-S profiles
  - Subsumption Reasoning Matching Engine
  - Augments a UDDI Server
- DAML-S API
  - APIs are being developed to support generation of requests and reasoning about matching profiles.

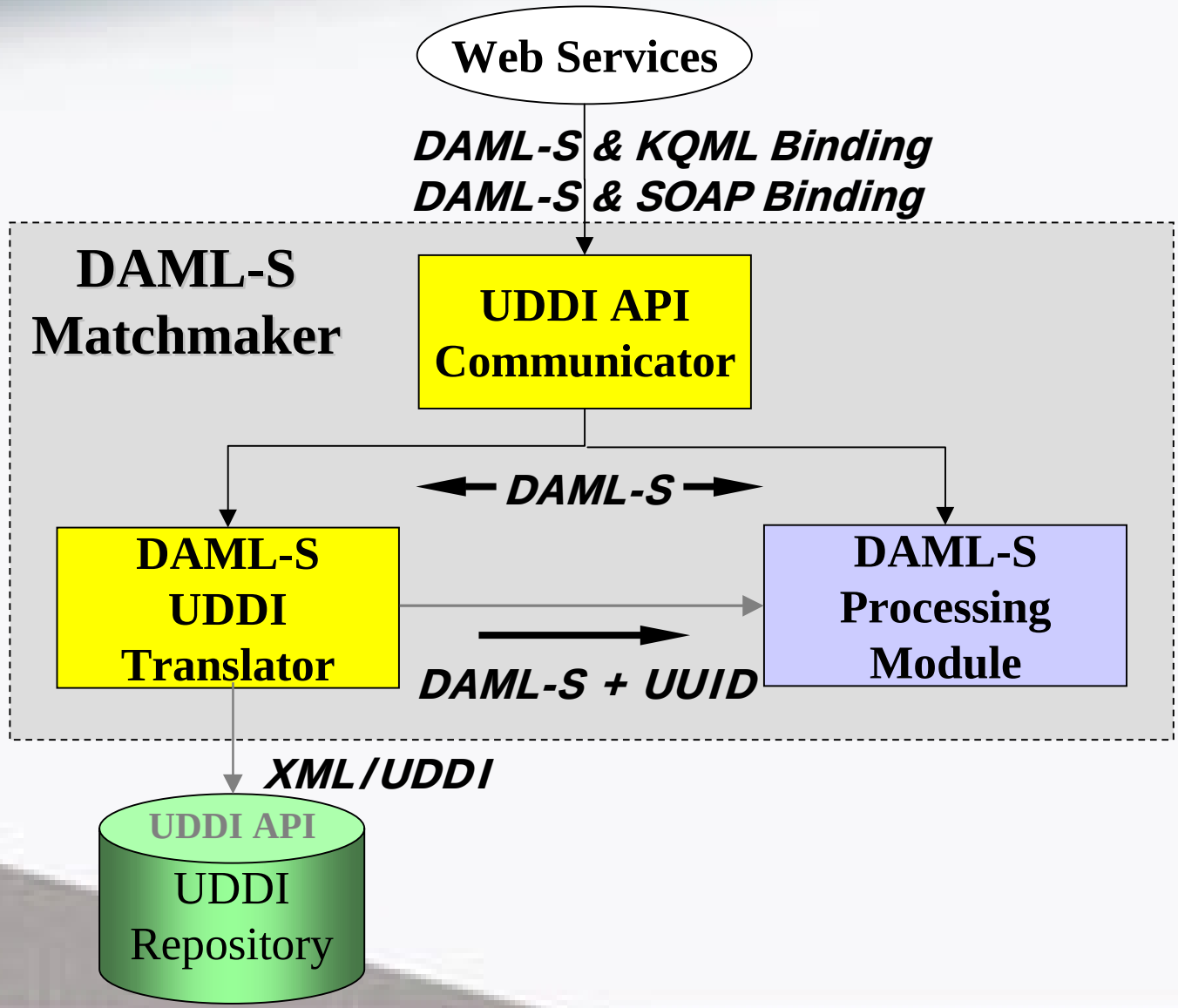
DAML-S Matchmaker

- Filter-Based Semantic Matching Engine for DAML-S profiles based on heuristic filters
  - Logical inference on the DAML Ontologies guarantee correctness of matching
  - Information Retrieval techniques that help to speed up the matching
- Extends an existing UDDI server
  - Enable capability descriptions and matching of DAML enabled services registered with UDDI

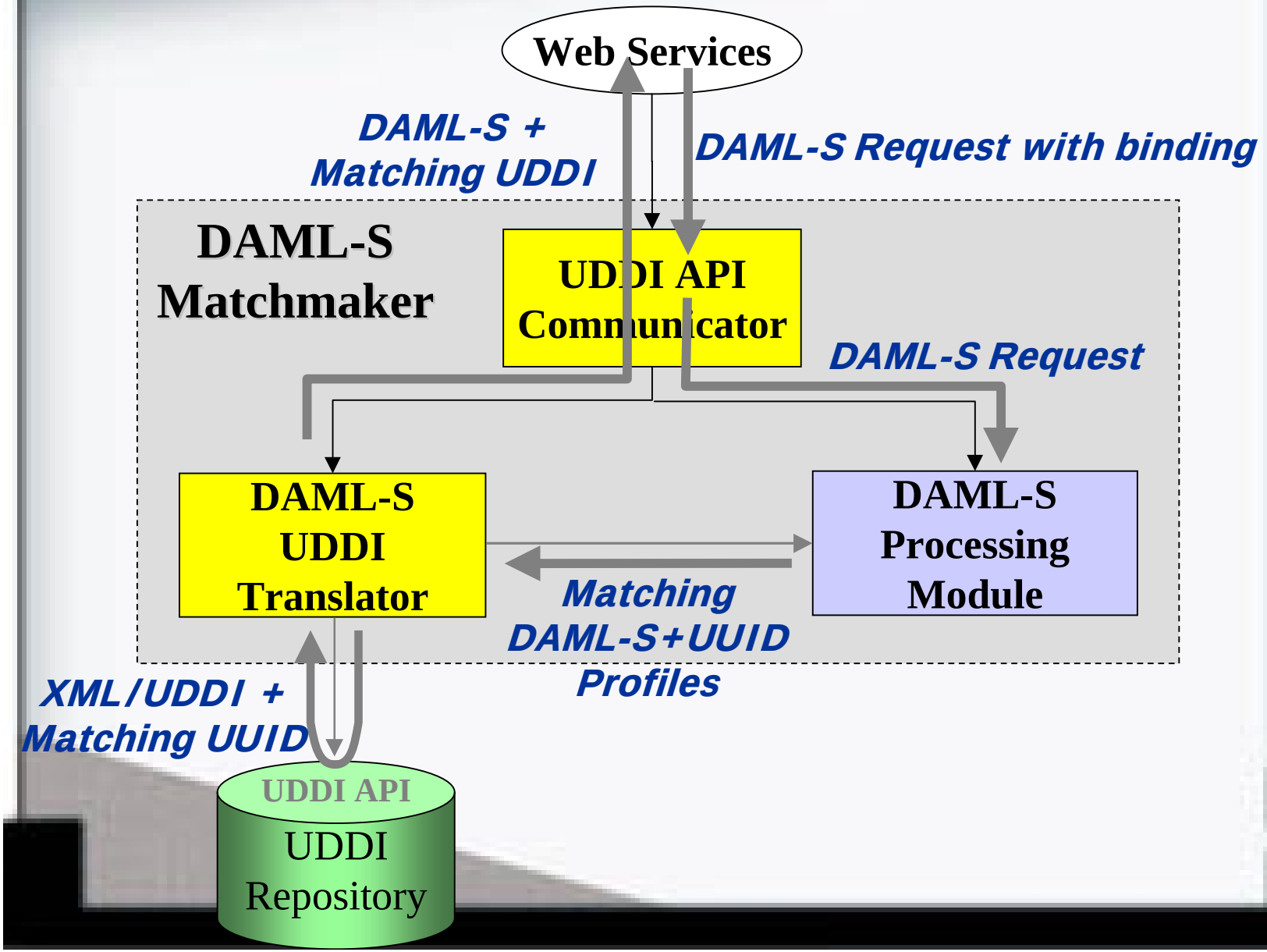
DAML-S Matchmaker Processing Module



Extending UDDI



DAML-S Request Data Flow



Semantic Web Services

Contrasting DAML-S with UDDI

- UDDI
  - Goal: **integration** and **semi-automation** of B2B transactions
  - Provides registry of business and their services in XML
  - Matching of advertisements and requests is based on **keywords**
  - **Interaction** with services is done through creating **customized programs**

Contrasting DAML-S with UDDI (cont)

- Each business description consists of:
  - *businessEntity:* describes businesses by name, key, categorization, services offered and contact information
  - *businessService* describes services by name, key, categorization and multiple bindingTemplate elements
  - *bindingTemplate* describes the service access (phone, FAX, http, ftp etc), key, TModelInstances
  - *TModelInstances* describe the protocols and interchange formats the service comprehends
- All the above elements are described through natural language text, and hence not amenable to machine comprehension.

Contrasting DAML-S with UDDI (cont)

- UDDI does not specify a content language for advertisements, but candidates are WSDL or XML/edi
- DAML-S could be a language candidate to be used on top of UDDI
- Though agents can search UDDI registries, humans must be involved to interpret the descriptions and program the access interface
- UDDI does not support Process Model description, hence no support for automatic composition of services

Contrasting DAML-S with WSDL

- Language to describe interface to business services in UDDI registries
- Services are defined as sets of ports
  - i.e. network addresses with associated protocols and data formats
- Messages and operations are mapped to ports
- A message is defined abstractly as a request, response or a parameter of a request or response
- Service preconditions or effects cannot be expressed

Contrasting DAML-S with E-speak

- E-speak and UDDI have similar goals to facilitate advertisement and discovery of services
  - HP is collaborating with the UDDI consortium
- E-speak describes services as sets of attributes common to a logical group of services
- Matches of lookup requests are done through attribute matching
  - Value types of attributes are String, Int, Boolean, and Double
- Basic vocabulary defining attributes as Name, Type (for string only), Description, Keyword and Version

Contrasting DAML-S with E-speak (cont)

- No semantic meaning attached to any of the attributes
- No (sub)typing has been specified
- Dependencies among attributes and logical constraints are not expressible
- E-speak requires an e-speak engine to be run on all client machines
- E-speak does not support service process model so no execution monitoring is possible

Contrasting DAML-S with ebXML

- Utilizes a workflow approach where information and documents flow between business
- Concept of Collaboration Protocol Profile (CPP) is a specification of the services offered by a ebXML compliant Trading Partner
- A Business process is a set of business document exchanges between the Trading Partners
- CPPs contain industry classification, contact information, supported Business Processes etc.
- CPPs are registered within an ebXML registry
- Trading Partners must agree on how business documents are specified

DAML-S and the Semantic Web

- DAML-S differs from many other Web Service standards:
  - Semantic markup will allow agents to understand content, discover, interoperate, transact and compose services
  - Designed to be fully automated, whereas other schemes involve human-in-the-loop.

DAML-S Resources

- DAML-S Web page - http://www.daml.org/services/
- DAML-S Matchmaker - http://www.damlsmm.ri.cmu.edu/
- DAML Tools - http://www.daml.org/tools/
  - DAML Validator - http://www.daml.org/validator/
  - DAML Editor (Emacs Mode) - http://openmap.bbn.com/~burstein/daml-emacs/

Questions?

Semantic Web Services

Discussion & Wrap Up

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Web Service and the Semantic Web

- Current Status
  - Where are Web Services heading?
- Applications
  - Case studies of emerging Web Services
- Realising Services on the Semantic Web
  - What challenges / issues need to be addressed?

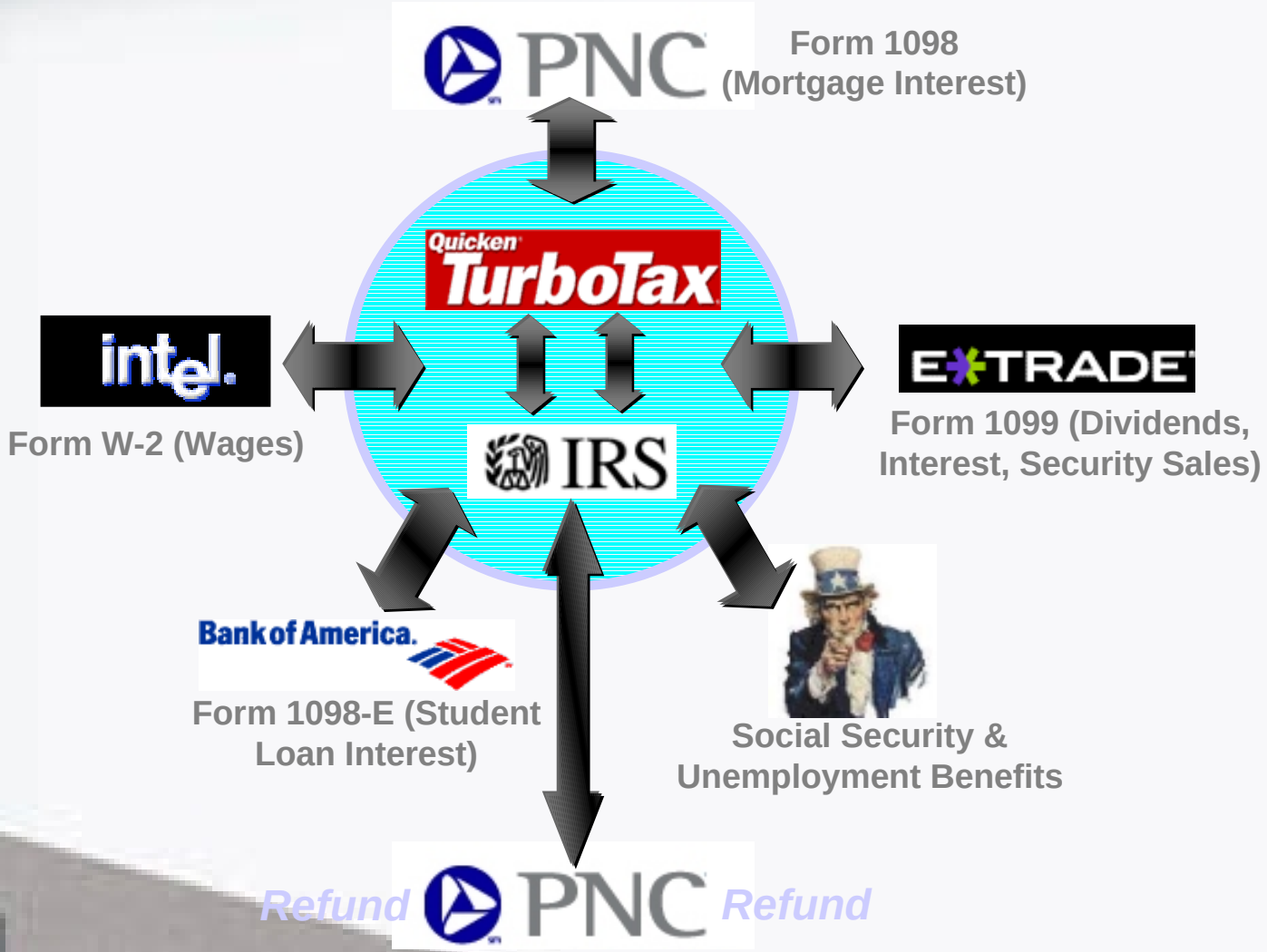
Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Where will Web Services go?

Imagine your income tax refund deposited in your PNC Bank checking account without lifting a finger

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

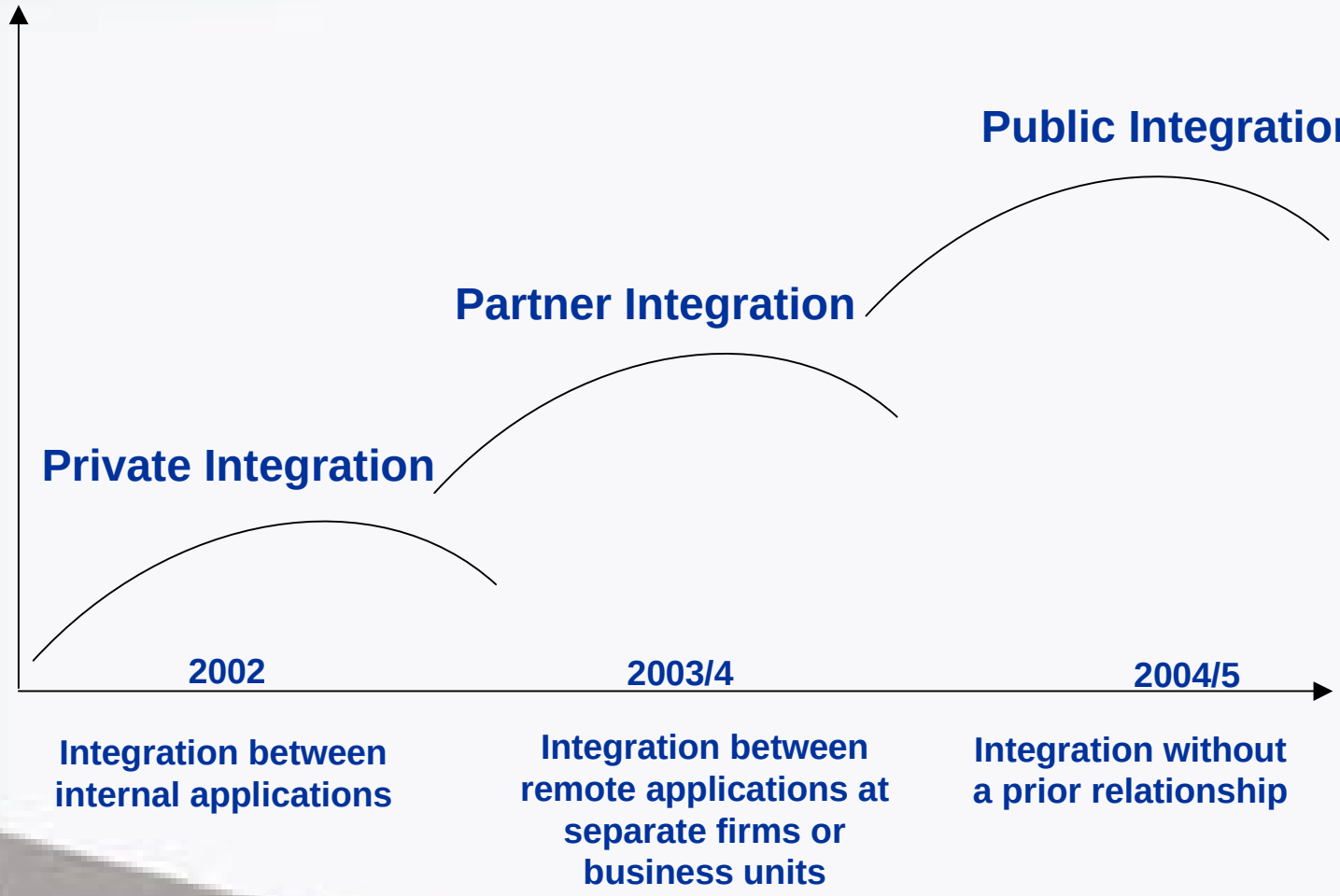
Where will Web Services Go?



Semantic Web Services

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Evolution of Web Services



Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

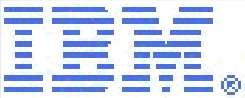
How are Web Services Used?

Current and emerging applications









- Using web services to broker shipping orders to authorized agents or independent third party transportation companies
- Using web services to create a structured data application programming interface that allows third parties to create booking engines and server-based applications
- Using web services to develop an open non-proprietary solution to connect 3rd-party service providers
- Using web services to integrate web-based applications with legacy applications

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

How are Web Services Used?

Current and emerging applications









- Using web services to make its software products more accessible to the end user
- Using web services to connect its reservation system with Southwest Airlines
- Using web services to reduce the costs associated with providing monthly content updates to its Internet partners
- Using web services to form partnerships with e-commerce and online catalog solution providers

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Realizing Web Services on the Semantic Web

- How can a symbiosis of Web Services and the Semantic Web be achieved?
  - Tools for automatic creation of product ontologies, annotations and annunciations
  - Versioning and Managing Changes
  - Managing and Mediating between different versions of Web Service protocols and standards
  - Privacy / Trust / Security
  - Business Models - Thoughts and Solutions

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Realizing Web Services on the Semantic Web (cont)

- Complement and Augment Industrial Efforts
  - e.g. DAML-S Matchmaker
    - Augments UDDI
  - How can adding semantic capability descriptions to existing web service advertisements be automated?
  - Will developers be capable of adequately defining their services at the ‘knowledge level’?
    - What kinds of support will be needed?
- Evaluate utility of adding semantics to Web Services

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Outstanding Technical Issues For DAML-S

- Issues of DAML+OIL expressiveness for describing DAML-S
  - e.g. preconditions and effects, rules (conditionals)
- Addressing Failures
  - reasons for failure
  - default failure behaviours
- Defining implications and rules

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University

Questions?

Copyright 2002 Intelligent Software Agents Laboratory, Carnegie Mellon University