Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

Concrete Services

The Hands-on of the Service Researchers

Massimo Mecella

Dipartimento di Informatica e Sistemistica ANTONIO RUBERTI
SAPIENZA Università di Roma
mecella@dis.uniroma1.it

based on a talk given at INFINT 2009

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

Outline

- Basic Concepts
- Relevant Technologies and Abstractions
- Examples in Specific Application Domains

BASIC CONCEPTS

e-Services, Web Services, Services … (1) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- An e-Service is often defined as an application accessible via the Web, that provides a set of functionalities to businesses or individuals. What makes the e-Service vision attractive is the ability to automatically discover the e-Services that fulfill the users' needs, negotiate service contracts, and have the services delivered where and when users needs them

  Guest editorial. In [VLDBJ01]

- e-Service: an application component provided by an organization in order to be assembled and reused in a distributed, Internet-based environment; an application component is considered as an e-Service if it is: (i) open, that is independent, as much as possible, of specific platforms and computing paradigms; (ii) developed mainly for inter-organizations applications, not only for intra-organization applications; (iii) easily composable; its assembling and integration in an inter-organizations application does not require the development of complex adapters. e-Application: a distributed application which integrates in a cooperative way the e-Services offered by different organizations

  M. Mecella, B. Pernici: Designing Wrapper Components for e-Services in Integrating Heterogeneous Systems. In [VLDBJ01]

e-Services, Web Services, Services … (2) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols

Web Services Architecture Requirements, W3C Working Group Note, 11 Feb. 2004, *http://www.w3.org/TR/wsa-reqs/*

e-Services, Web Services, Services … (3) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Services are self-describing, open components that support rapid, low-cost composition of distributed applications. Services are offered by service providers — organizations that procure the service implementations, supply their service descriptions, and provide related technical and business support.

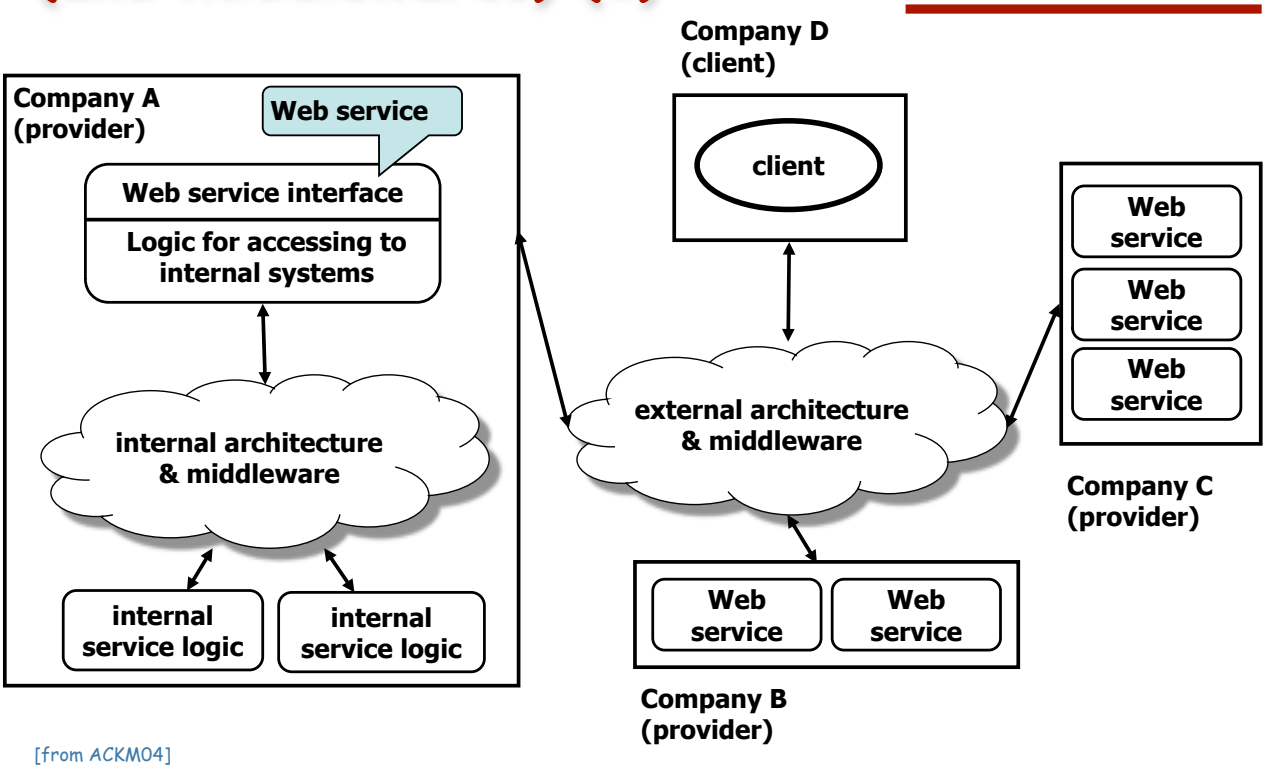
  Since services may be offered by different enterprises and communicate over the Internet, they provide a distributed computing infrastructure for both intra and cross-enterprise application integration and collaboration.

  Service descriptions are used to advertise the service capabilities, interface, behavior, and quality. Publication of such information about available services provides the necessary means for discovery, selection, binding, and composition of services. In particular, the service capability description states the conceptual purpose and expected results of the service (by using terms or concepts defined in an application-specific taxonomy). The service interface description publishes the service signature (its input/output/error parameters and message types). The (expected) behavior of a service during its execution is described by its service behavior description. Finally, the Quality of Service (QoS) description publishes important functional and nonfunctional service quality attributes […]. Service clients (end-user organizations that use some service) and service aggregators (organizations that consolidate multiple services into a new, single service offering) utilize service descriptions to achieve their objectives.
- The application on the Web (including several aspects of the SOA) is manifested by Web services

And Today ?

- e-Service
  - e-Service is the provision of a service via the Internet (the prefix "e" standing for "electronic")
  - True Web jargon, meaning just about anything done online
  - Basically whichever Web application usable by a human, through a user interface
- Web service
  - software component available on the Web, to be invoked by some other client application/component
  - A way of building Web-scale component-based distributed systems
- For building an e-Service, a designer may need to use/

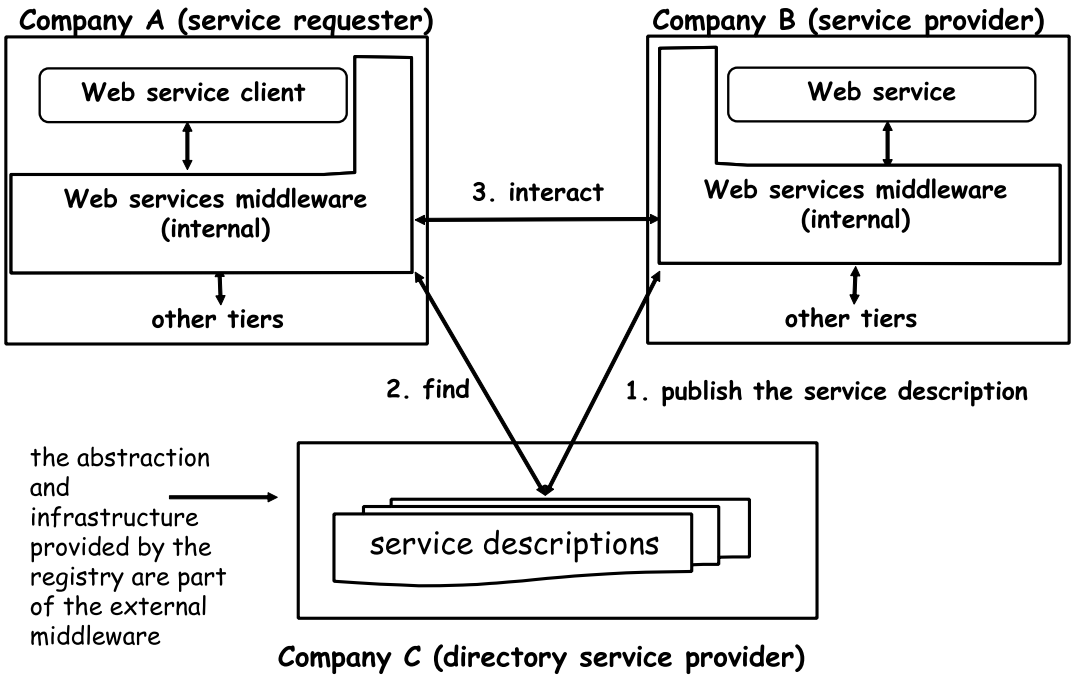
Two Architectures (and Middlewares) (1)



[from ACKM04]

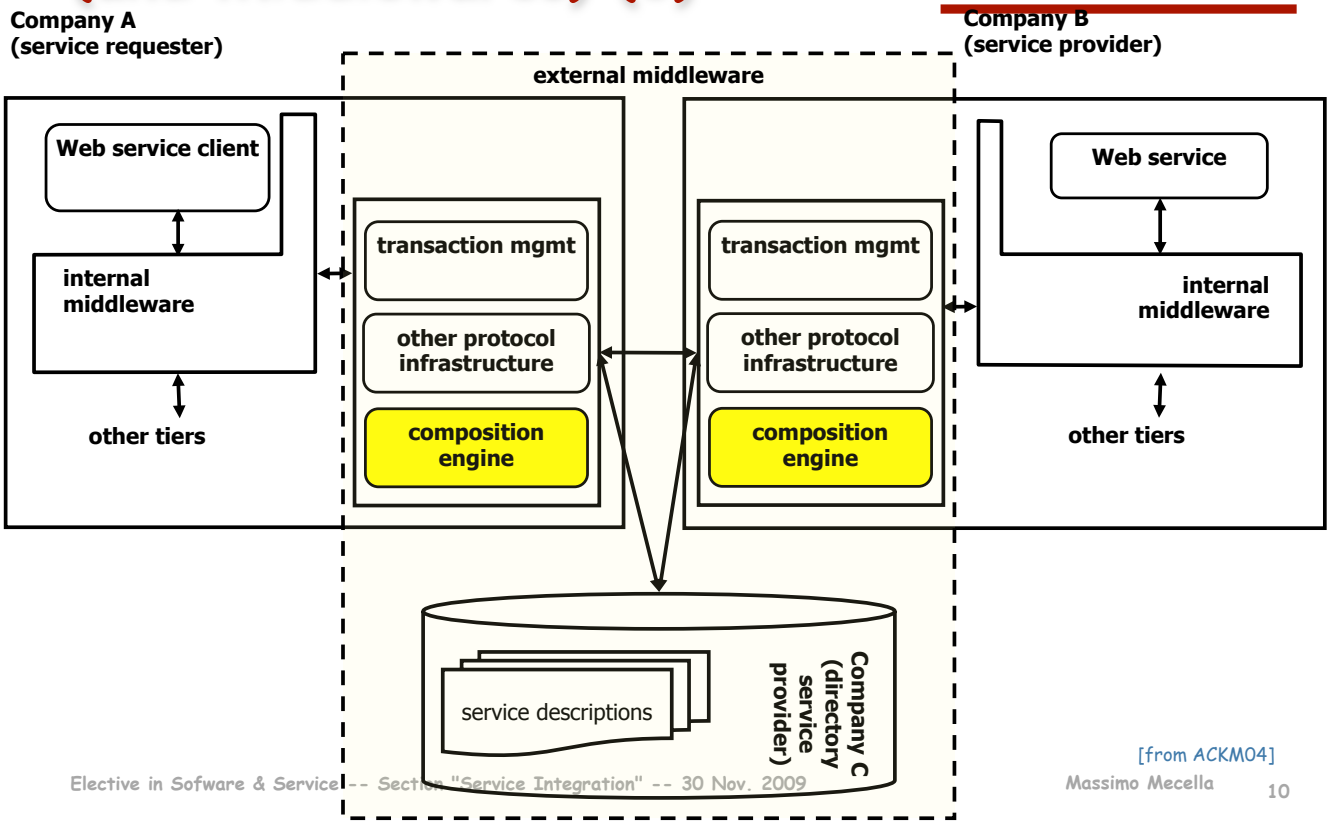
Two Architectures (and Middlewares) (2)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



Two Architectures (and Middlewares) (3)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

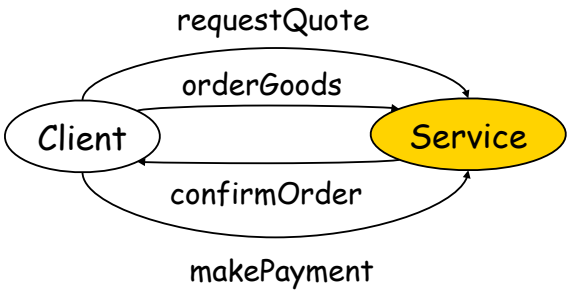


[from ACKM04]

Services

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

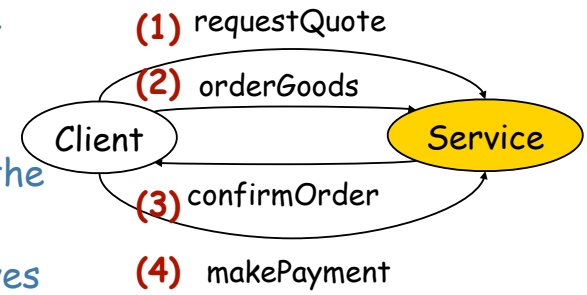
- A service is characterized by the set of (atomic) operations that it exports …

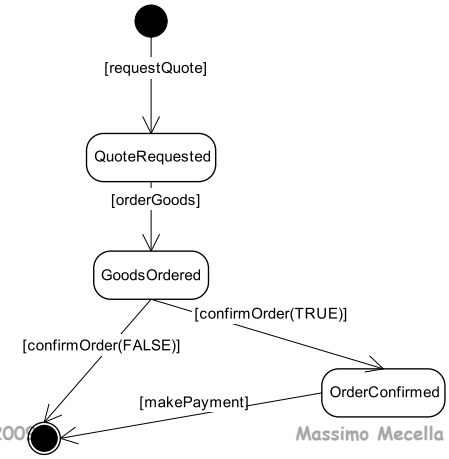


Services

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- A service is characterized by the set of (atomic) operations that it exports …
- … and possibly by constraints on the possible conversations
  - Using a service typically involves performing sequences of operations in a particular order (conversations)
  - During a conversation, the client typically chooses the next operation to invoke (on the basis of previous results, etc.) among the ones that the service allows at that point

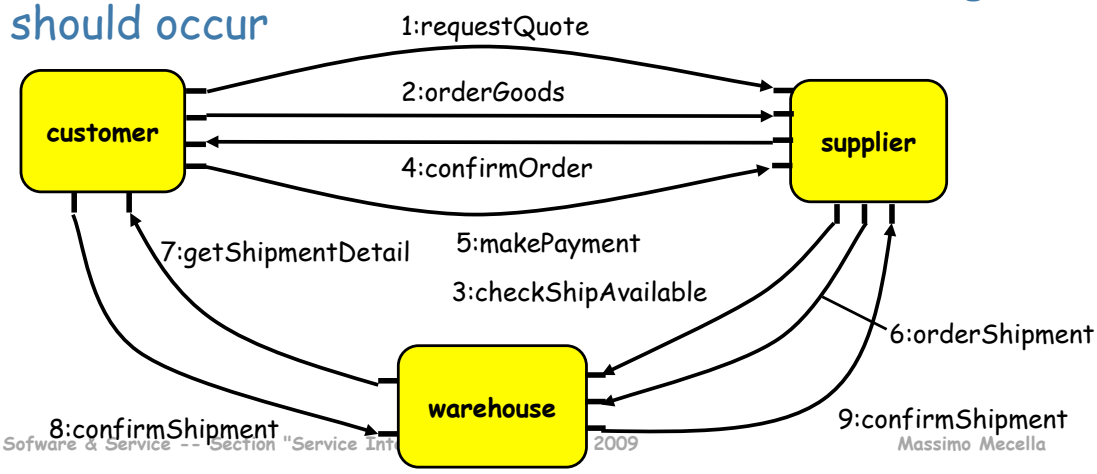




Choreography: Coordination of Conversations of N Services

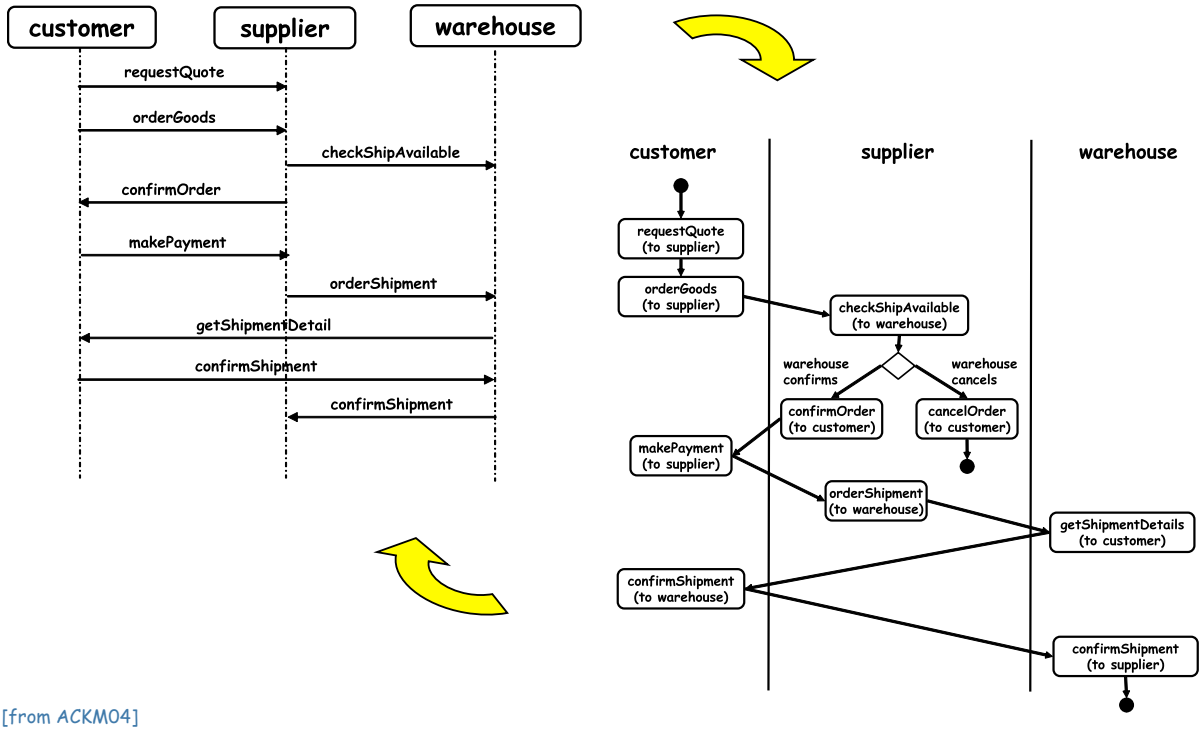
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Global specification of the conversations of N peer services (i.e., multi-party conversations)
  - Roles
  - Message exchanges
  - Constraints on the order in which such exchanges should occur



Choreography: Coordination of Conversations of N Services

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

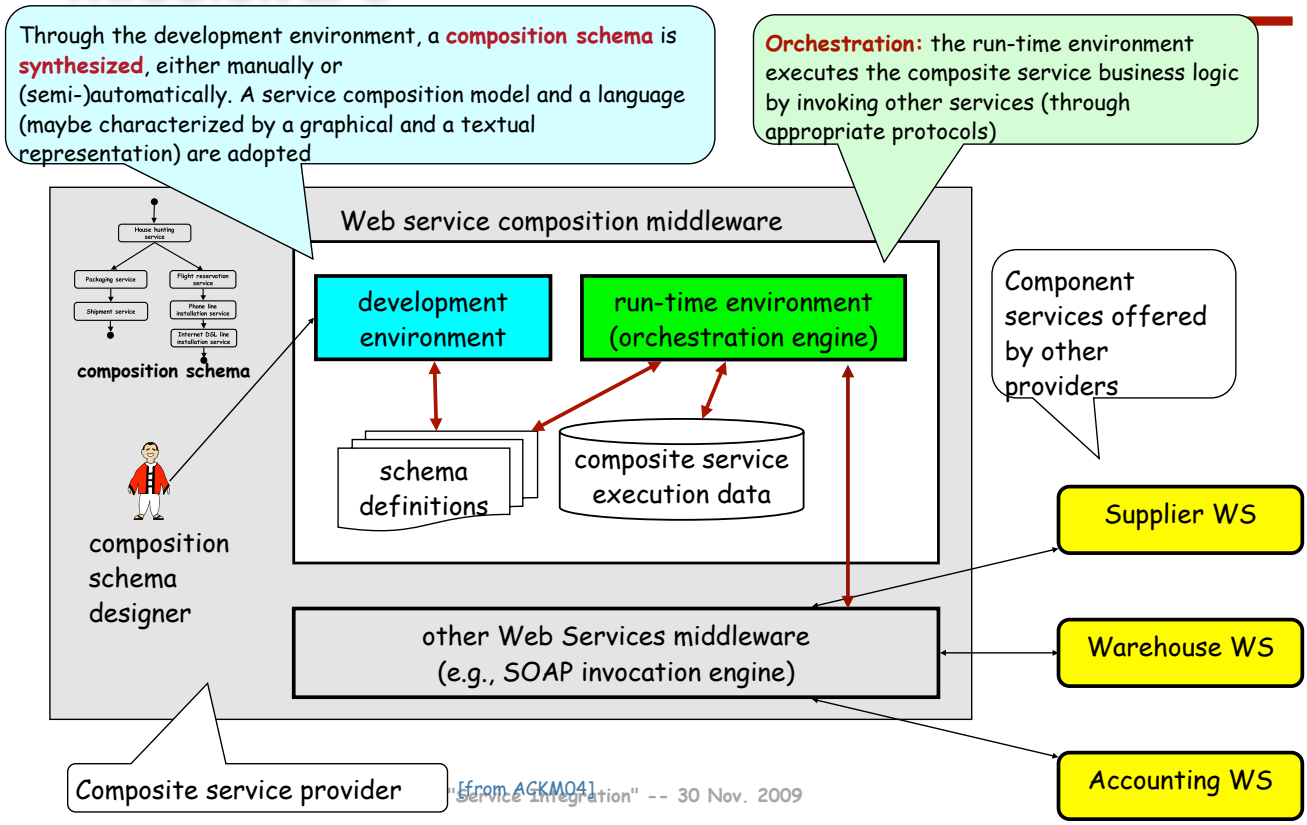


[from ACKM04]

Composition

- Deals with the implementation of an application (in turn offered as a service) whose application logic involves the invocation of operations offered by other services
  - The new service is the composite service
  - The invoked services are the component services

The Composition Engine/ Middleware



[from AGKM04]

Synthesis and Orchestration

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- (Composition) Synthesis: building the specification of the composite service (i.e., the composition schema)
  - Manual
  - Automatic
- Orchestration: the run-time management of the composite service (invoking other services, scheduling the different steps, etc.)
  - Composition schema is the "program" to be executed
  - Similarities with WfMSs (Workflow Management

Composition Schema

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- A composition schema specifies the "process" of the composite service
  - The "workflow" of the service
- Different clients, by interacting with the composite service, satisfy their specific needs (reach their goals)
  - A specific execution of the composition schema for a given client is an orchestration instance

Choreography (Coordination) vs. Composition (Orchestration)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Composition is about implementing new services
  - From the point of view of the client, a composite service and a basic (i.e., implemented in a traditional programming language) one are indistinguishable
- Choreography is about global modeling of N peers, for proving correctness, design-time discovery of possible partners and run-time bindings

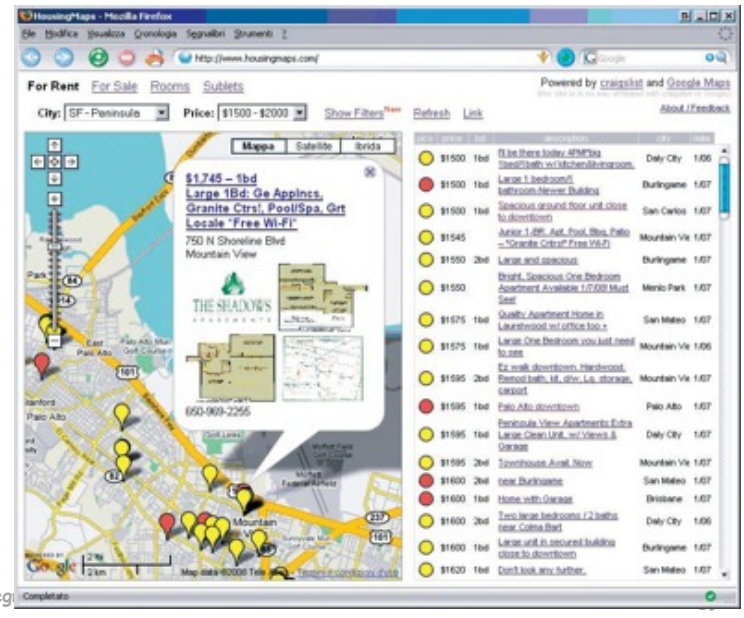
- N.B.: There is a strong relationship between a service internal composition and the external choreographies it can participate in
  - if A is a composite service that invokes B, the A's composition schema must reflect the coordination protocol governing A – B interactions
  - in turn, the composition schema of A determines the coordination protocols that A is able to support (i.e., the

Services Mash-up (1)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

Web application that combines data from one or more sources into a single integrated tool

- easy, fast integration, frequently done by access to open APIs and data sources to produce results that were not the original reason for producing the raw source data.
- E.g., cartographic data from Google Maps to add location information to real estate data, thereby creating a new and distinct e-Service that was not originally provided by either source
- Bottom-up, developers-driven approach



D. Benslimane, S. Dustdar, A. Sheth (eds.). Services Mashups – Special Issue. IEEE Internet Computing, vol. 12, no. 5, 2008.

Services Mash-up (2)

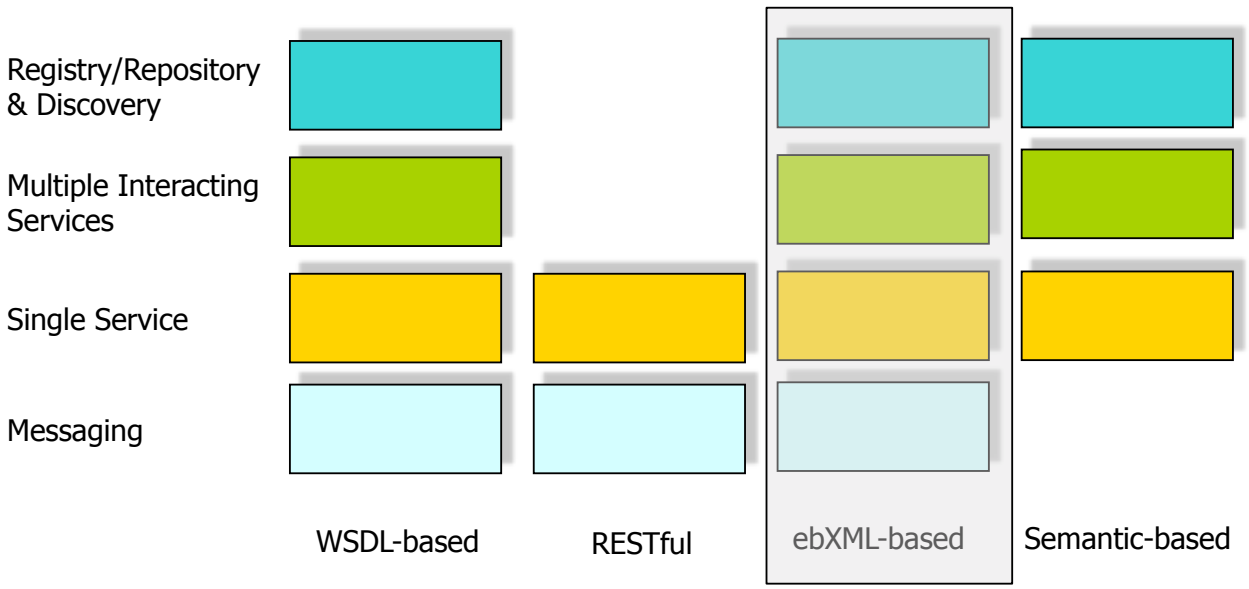
- Based on various technologies
  - Web services
    - SOAP
    - RESTful
    - Atom/RSS
- Basically a lightweight form of composition

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

RELEVANT TECHNOLOGIES AND ABSTRACTIONS

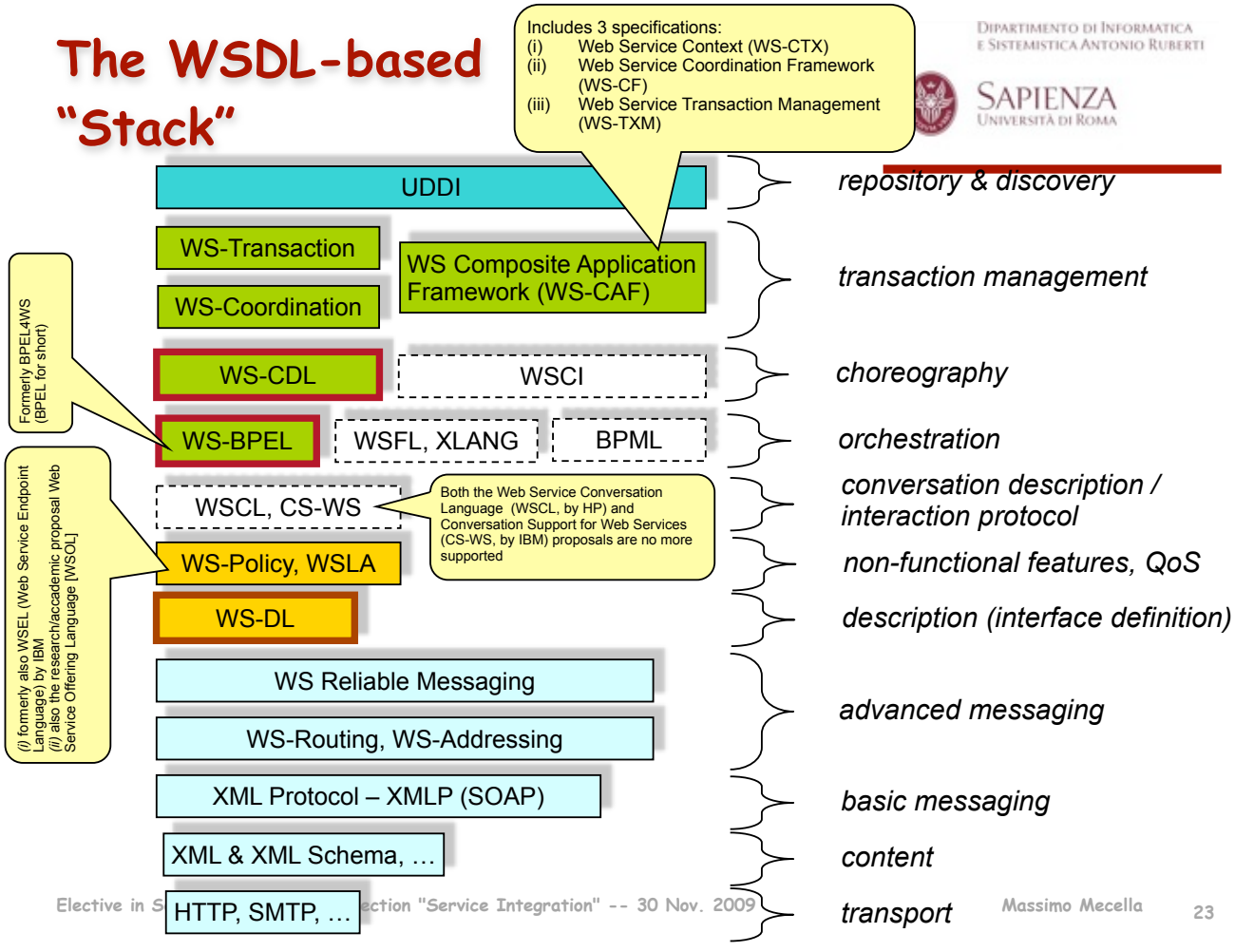
The "Stacks" of Service Technologies

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



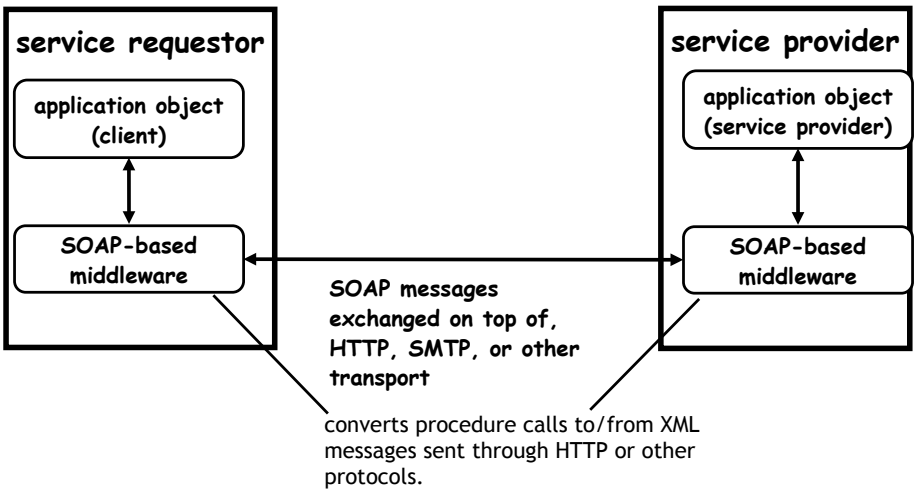
The WSDL-based "Stack"

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



A Minimalist Infrastructure for Web Service

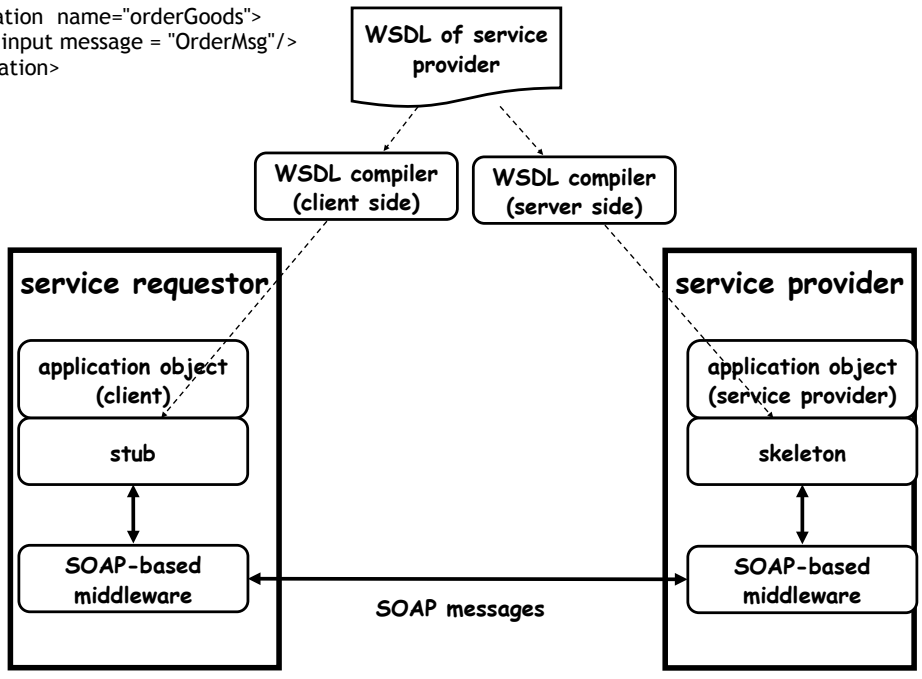
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



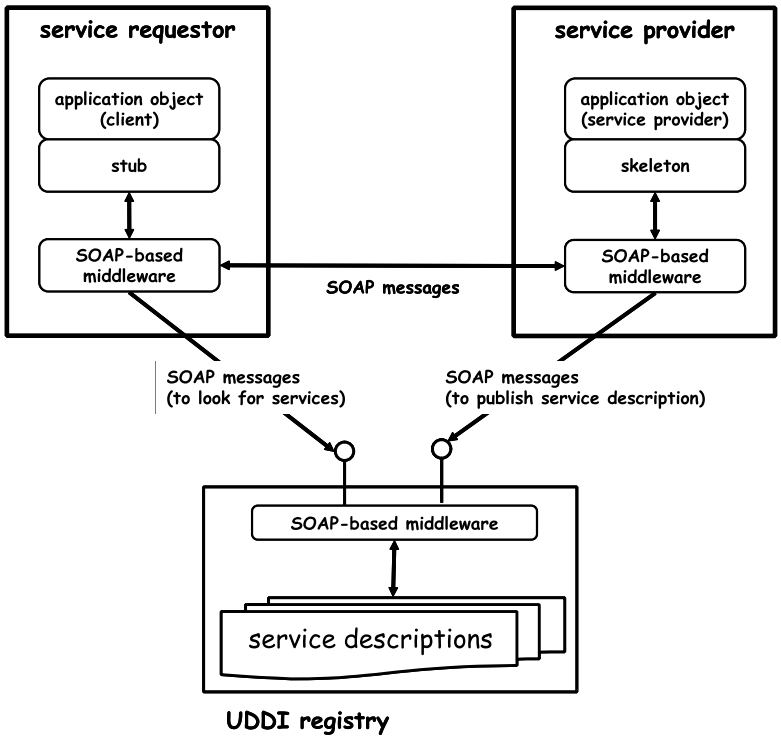
From Interfaces to Stub/ Skeleton

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

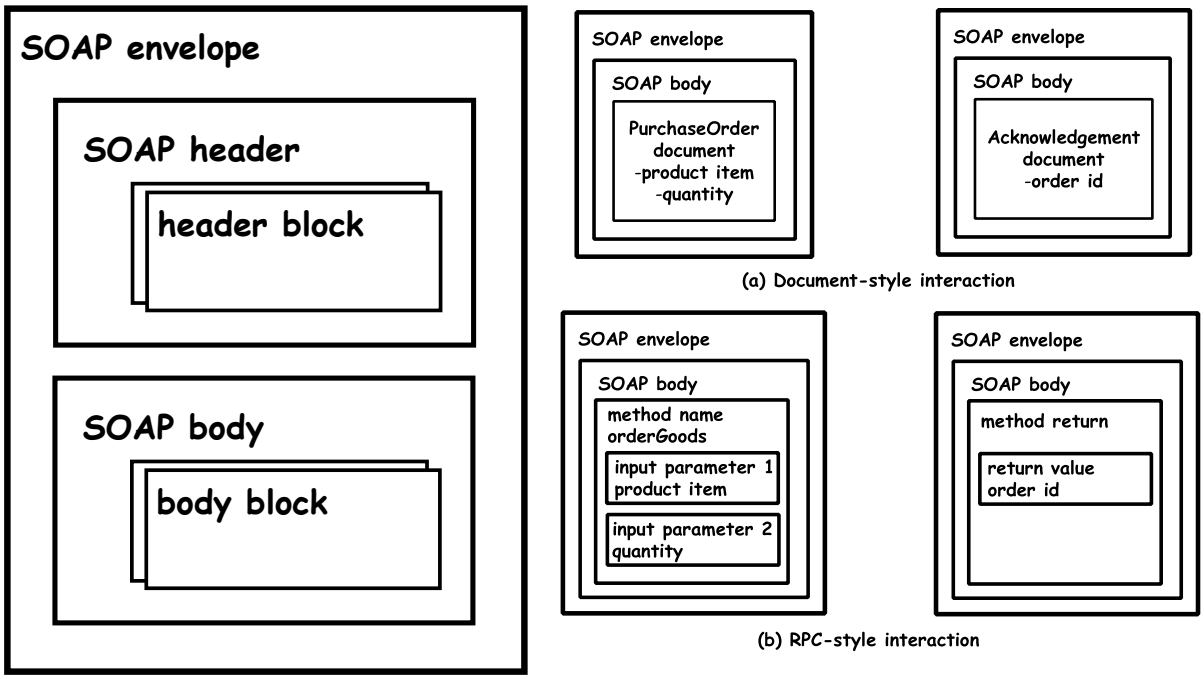
```
<operation  name="orderGoods">
      <input message = "OrderMsg"/>
</operation>
```



Registry



SOAP (1)



SOAP (2)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

```
<ProductItem>
  <name>…</name>
  <type>…</type>
  <make>…</make>
</ProductItem>
```

```
<ProductItem
  name="…"
  type="…"
  make="…"
/>
```

```
<ProductItem name="…"
  <type>…</type>
  <make>…</make>
</ProductItem>
```

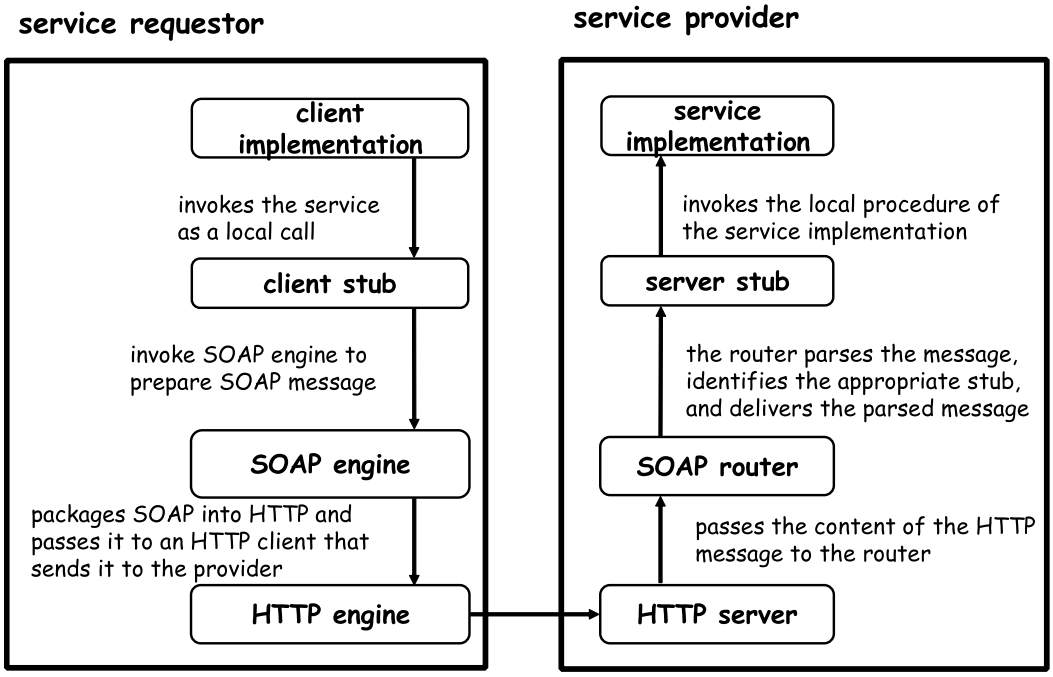
Different encoding styles

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
 <env:Header>
  <t:transactionID
          xmlns:t="http://intermediary.example.com/procurement"
          env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
          env:mustUnderstand="true" >
          57539
  </t:transactionID>
 </env:Header>

 <env:Body>
  <m:orderGoods
     env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
     xmlns:m="http://example.com/procurement">
   <m:productItem>
         <name>ACME Softener</name>
   </m:productItem>
   <m:quantity>
     35
   </m:quantity>
  </m:orderGoods>
 </env:Body>

</env:Envelope>
```

envelope
header
blocks
body

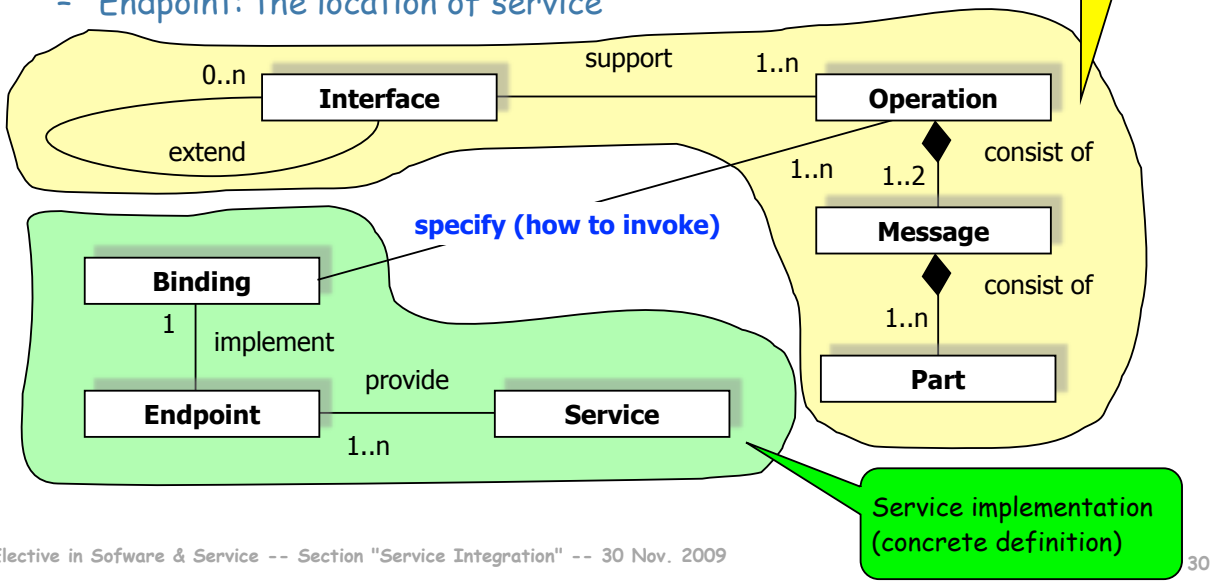
The Simplest SOAP Middleware

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

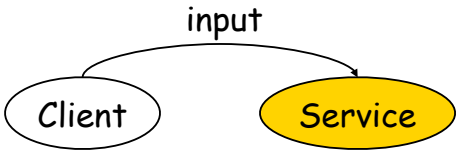


Web Service Definition Language (WS-DL)

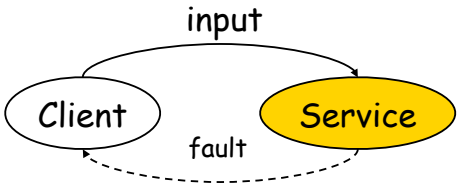
- WS-DL (v2.0) provides a framework for defining
  - Interface: operations and input/output formal parameters
  - Access specification: protocol bindings (e.g., SOAP)
  - Endpoint: the location of service



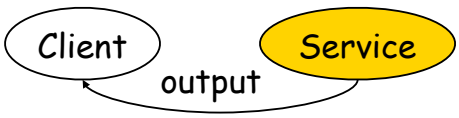
Message Exchange Patterns (1)



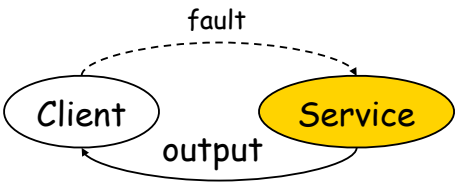
in-only (no faults)



robust in-only (message triggers fault)

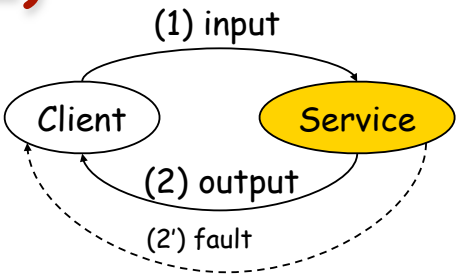


out-only (no faults)

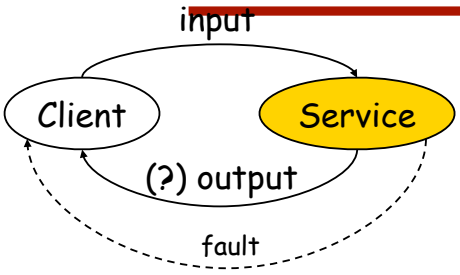


robust out-only (message triggers fault)

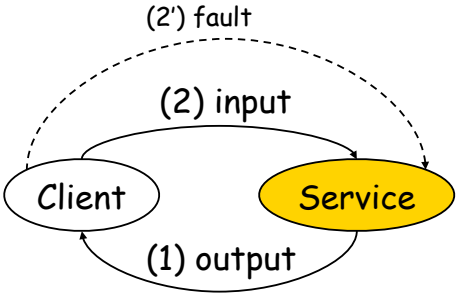
Message Exchange Patterns (2)



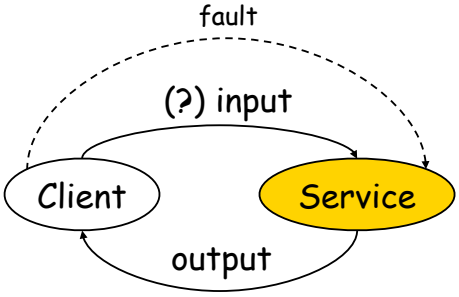
in-out (fault replaces message)



in-optional-out (message triggers fault)



out-in (fault replaces message)



out-optional-in (message triggers fault)

An Example (1)

Definition of a message and its formal parameters

```
<definitions … >
   <types>
        <element name="ListOfSong_Type">
               <complexType><sequence>
                      <element minOccurs="0" maxOccu   "unbound"
                             name="SongTitle" type="   :string"/>
               </sequence></complexType>
        </element>
        <element name="SearchByTitleRequest">
               <complexType><all>
                      <element name="containedInTitle"
                             type="xs:string"/>
               </all></complexType>
        </element>
        <element name="SearchByTitleResponse">
               <complexType><all>
                      <element name="matchingSongs"
                             xsi:type="ListOfSong_Type"/>
               </all></complexType>
         </element>
```

An Example (2)


```
<element name="SearchByAuthorRequest">
        <complexType><all>
                <element name="authorName"
                        type="xs:string"/>
        </all></complexType>
</element>
<element name="SearchByAuthorResponse">
        <complexType><all>
                <element name="matchingSongs"
                        xsi:type="ListOfSong_Type"/>
        </all></complexType>
</element>
<element name="ListenRequest">
        <complexType><all>
                <element name="selectedSong"
                        type="xs:string"/>
        </all></complexType>
</element>
```


An Example (3)


```
        <element name="ListenResponse">
                <complexType><all>
                        <element name="MP3fileURL" type="xs:string"/>
                </all></complexType>
        </element>
        <element name="ErrorMessage">
                <complexType><all>
                        <element name="cause" type="xs:string"/>
                </all></complexType>
        </element>
</types>
```

An Example (4)

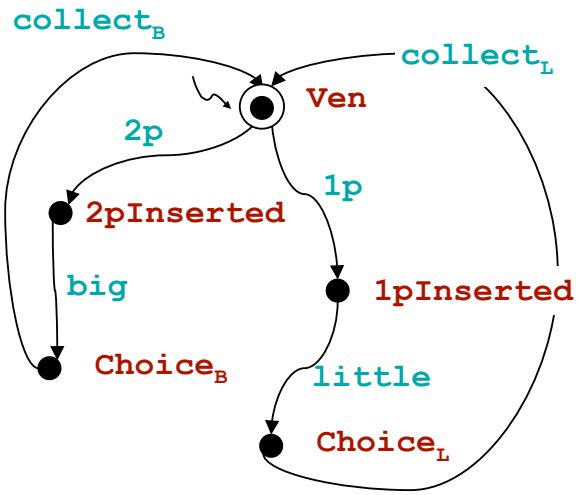
Definition of a service interface

```
<interface name="MP3ServiceType">
     <operation name="search_by_title" pattern="in-out">
            <input message="SearchByTitleRequest"/>
            <output message="SearchByTitleResponse"/>
            <outfault message="ErrorMessage"/>
     </operation>
     <operation name="search_by_author" pattern="in-out">
            <input message="SearchByAuthorRequest"/>
            <output message="SearchByAuthorResponse"/>
            <outfault message="ErrorMessage"/>
     </operation>
     <operation name="listen" pattern="in-out">
            <input message="ListenRequest"/>
            <output message="ListenResponse"/>
            <outfault message="ErrorMessage"/>
     </operation>
</interface>
</definitions>
```

Definition of an operation and its message exchange pattern

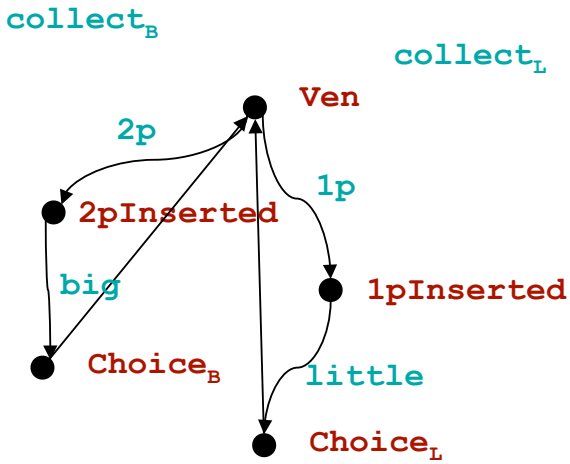
Transition Systems

- A transition system (TS) is a tuple $T = \langle A, S, S^0, \delta, F \rangle$ where:
  - A is the set of actions
  - S is the set of states
  - $S^0 \in S$ is the set of initial states
  - $\delta \subseteq S \S A \S S$ is the transition relation
  - $F \subseteq S$ is the set of final states



Process Algebras and TSs

- Process theory:
  - a process is a term of an algebraic language
  - a transition $E \rightarrow_a F$ means that process E may become F by performing (participating in, or accepting) action a
  - structured rules guide the derivation
- A graph:
  - nodes are process terms
  - labelled directed arcs between nodes



Ven = 2p.2pInserted + 1p.1pInserted

2pInserted = big.Choice$_B$

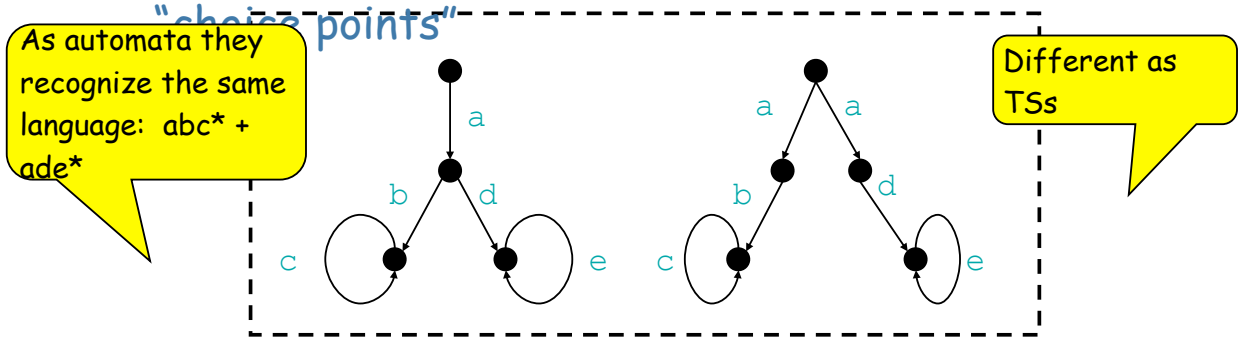
1pInserted = little.Choice$_L$

Choice$_B$ = collect$_B$.Ven

Choice$_L$ = collect$_L$.Ven

Automata vs. Transition Systems

- Automata
  - define sets of runs (or traces or strings): (finite) length sequences of actions
- TSs
  - … but I can be interested also in the alternatives "encountered" during runs, as they represent client's "choice points"

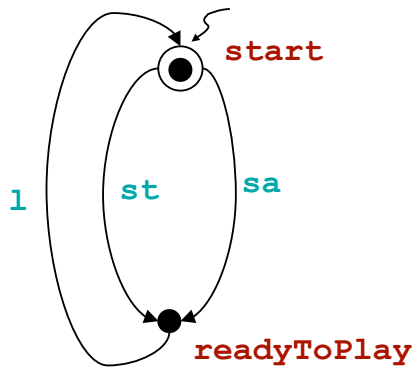


WS-DL is the Set of Actions

- A message exchange pattern (and the related operation) represents an interaction with the service client
  - an action that the service can perform by interacting with its client
- Abstracting from formal parameters, we can associate a different symbol to each operation …
- … thus obtaining the alphabet of actions

An Example

- The MP3ServiceInterface defines 3 actions:
  - search_by_title / st
  - search_by_author / sa
  - listen / l

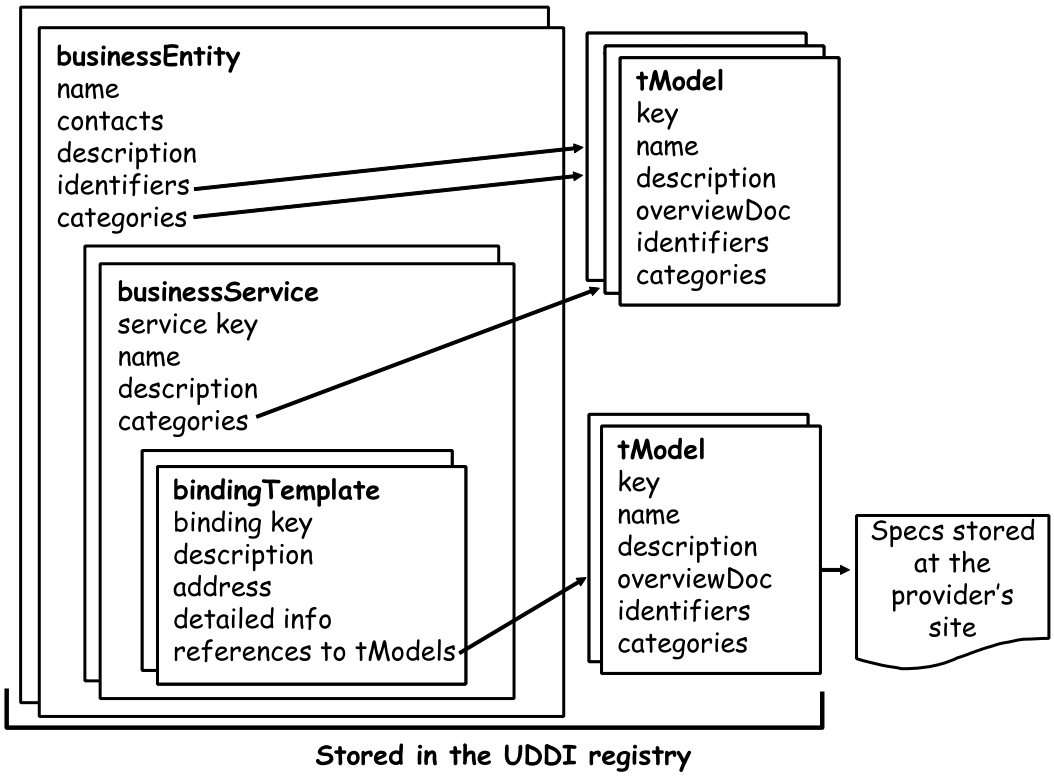


- Formally A = {st,sa,l}

Representing Service Behaviors in XML

- Different approaches for representing TSs
  - Web Service Transition (System) Language (WSTL/ WS-TSL)
    - The "Roman" proposals
  - Web Service Choreography Description Language (WS-CDL)
    - Standard
    - Not really designed for this
  - Web Service Business Process Execution Language (WS-BPEL) abstract
  - OWL-S
    - see, e.g., [Pistore&Traverso ISWC04]
  - WSMO

UDDI Data Structures



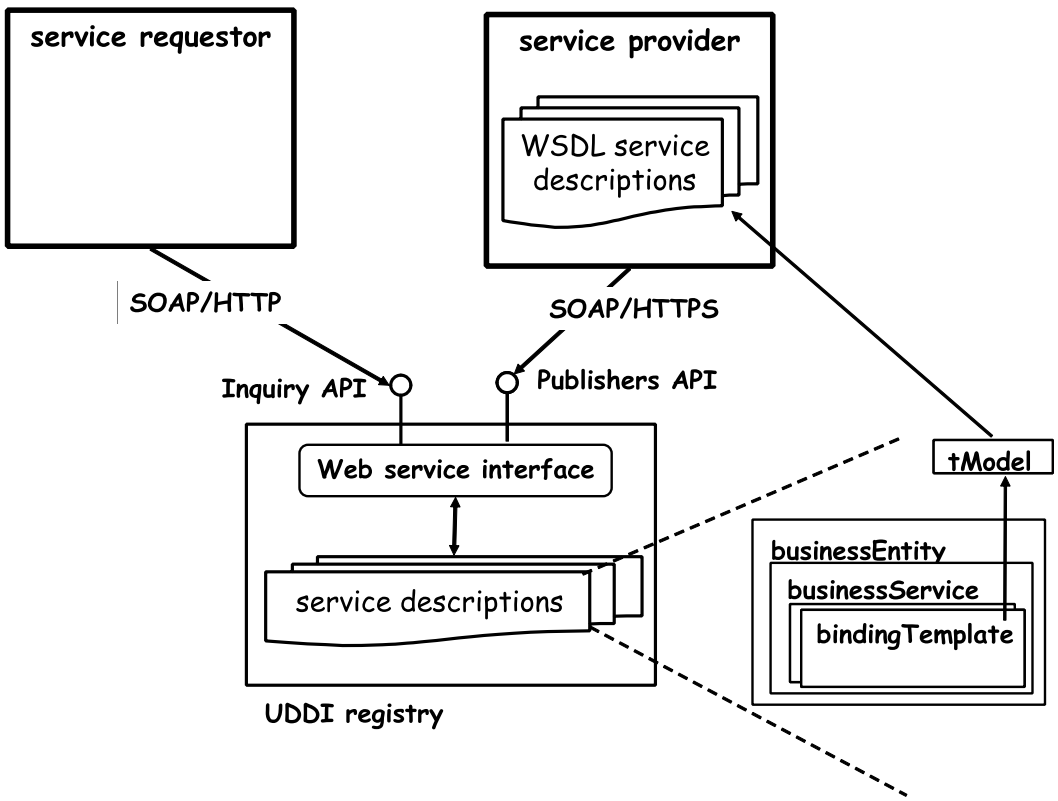
A Registry Not a Repository

overviewDoc (refer to WSDL specs and to API specs)

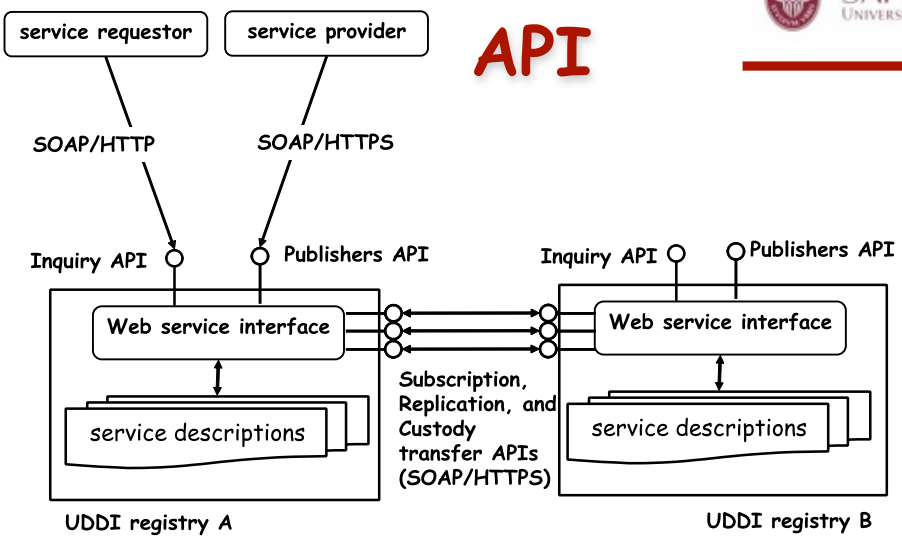
classification information (specifies that this tModel is about XML, WSDL, and SOAP specs)

```
<tModel tModelKey="uddi:uddi.org:v3_publication">
  <name>uddi-org:publication_v3</name>
  <description>UDDI Publication API V3.0</description>
  <overviewDoc>
     <overviewURL useType="wsdlInterface">
http://uddi.org/wsdl/uddi_api_v3_binding.wsdl#UDDI_Publication_SoapBinding
     </overviewURL>
  </overviewDoc>
  <overviewDoc>
     <overviewURL useType="text">
        http://uddi.org/pubs/uddi_v3.htm#PubV3
     </overviewURL>
  </overviewDoc>

  <categoryBag>
     <keyedReference keyName="uddi-org:types:wsdl"
        keyValue="wsdlSpec"
        tModelKey="uddi:uddi.org:categorization:types"/>
     <keyedReference keyName="uddi-org:types:soap"
        keyValue="soapSpec"
        tModelKey="uddi:uddi.org:categorization:types"/>
     <keyedReference keyName="uddi-org:types:xml"
        keyValue="xmlSpec"
        tModelKey="uddi:uddi.org:categorization:types"/>
     <keyedReference keyName="uddi-org:types:specification"
        keyValue="specification"
        tModelKey="uddi:uddi.org:categorization:types"/>
  </categoryBag>

</tModel>
```

UDDI and WSDL

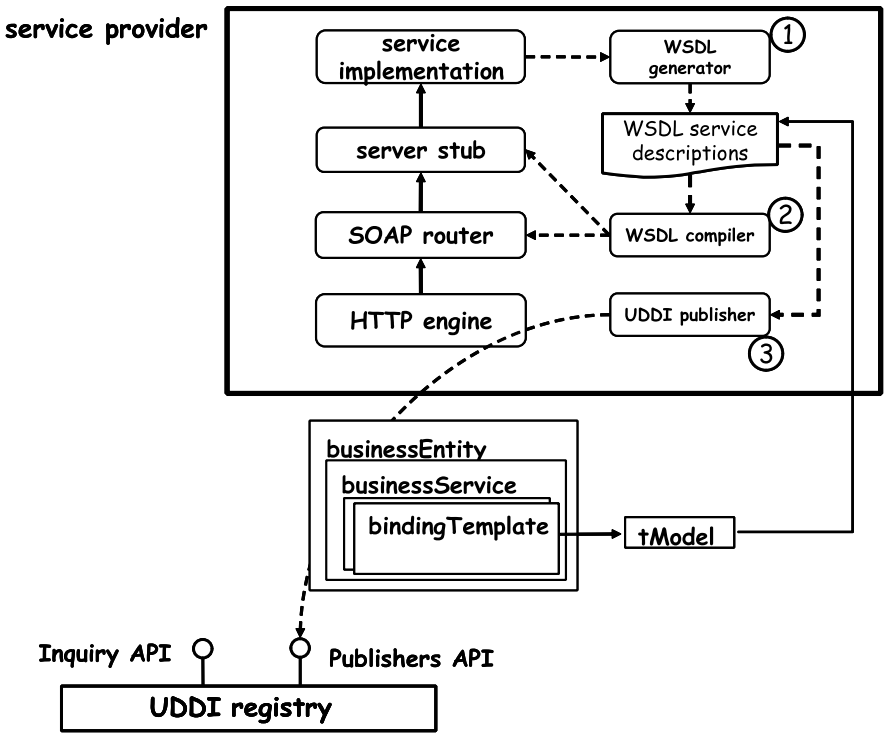


UDDI API



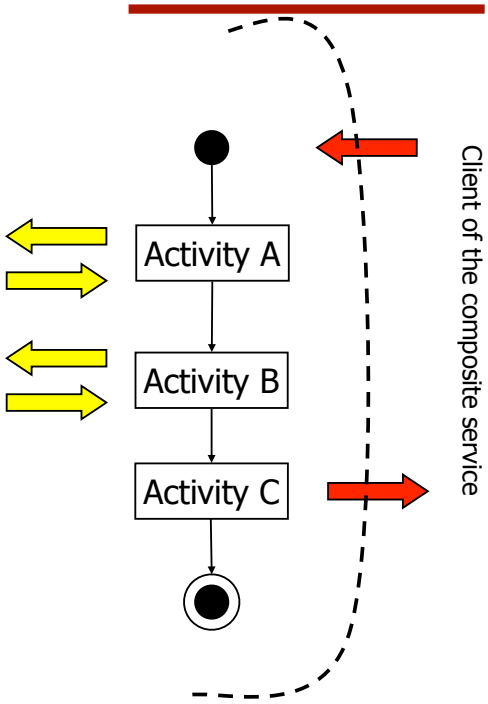

```
<?xml version="1.0"?>
<find_tModel generic="1.0" xmlns="urn:uddi-org:api">
  <categoryBag>
    <keyedReference tModelKey="UUID:C25893AF-1977-3528-36B5-4192C2AB9E2C"
                    keyName="uddi-org:types" keyValue="wsdlSpec"/>
    <keyedReference tModelKey="UUID:A15019C5-AE14-236C-331C-650857AE0221"
                    keyName="book pricing"
                    keyValue="36611349"/>
  </categoryBag>
```

Putting All Together



Business Process Execution Language for Web Services (WS-BPEL)

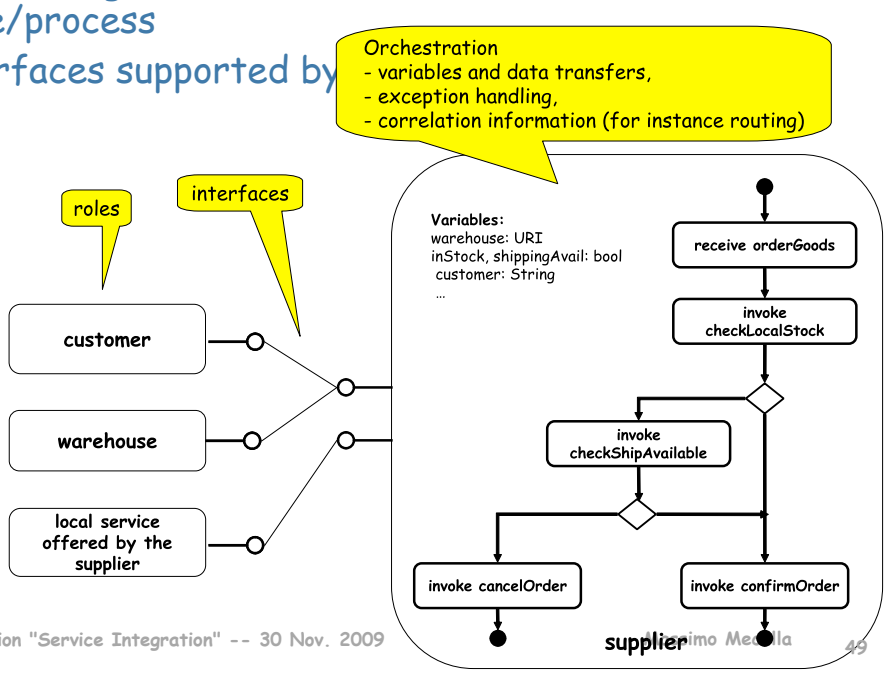
- Allows specification of composition schemas of Web Services
  - Business processes as coordinated interactions of Web Services
  - Business processes as Web Services
- Allows abstract and executable processes
- Influenced from
  - Traditional flow models
  - Structured programming
  - Successor of WSFL and XLANG
- Component Web Services described in WS-DL (v1.1)



WS-BPEL Specification

An XML document specifying

- Roles exchanging messages with the composite service/process
- The (WSDL) interfaces supported by such roles
- The orchestration of the process
  - Variables and data transfer
  - Exception handling
  - Correlation information



Process Model
(Activities)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Primitive
  - **invoke**: to invoke a Web Service (in-out) operation
  - **receive**: to wait for a message from an external source
  - **reply**: to reply to an external source message
  - **wait**: to remain idle for a given time period
  - **assign**: to copy data from one variable to another
  - **throw**: to raise exception errors
  - **empty**: to do nothing
- Structured
  - **sequence**: sequential order
  - **switch**: conditional routing
  - **while**: loop iteration
  - **pick**: choices based on events
  - **flow**: concurrent execution (synchronized by **links**)
  - **scope**: to group activities to be treated "transactionally" (managed by the same fault handler, within the

A link connects exactly one source activity S to exactly one target activity T; T starts only after S ends. An activity can have multiple incoming (possibly with join conditions) and outgoing links. Links can be guarded

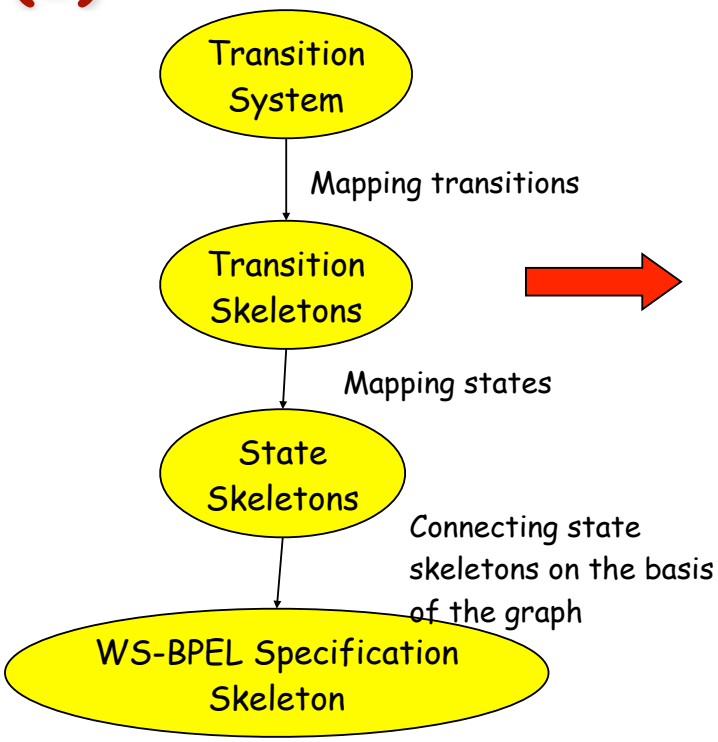
Process Model
(Data Manipulation and Exception Handling)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Blackboard approach
  - a blackboard of variables is associated to each orchestration instance (i.e., a shared memory within an orchestration instance)
  - variables are not initialized at the beginning; they are modified (read/write) by assignments and messages
  - manipulation through XPath
- Try-catch-throw approach
  - definition of fault handlers
  - … but also event handlers and compensation handlers (for managing transactionality as in the SAGA model)

From a TS to WS-BPEL (1)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma




```
<process name = "…">
    <partnerLinks>
     …
    </partnerLinks>

    <variables>
     …
    </variables>

    <flow>
       <links>
          …
       </links>
       <!-- state skel. -->
       …
       <!-- state skel. -->
    </flow>

</process>
```


From a TS to WS-BPEL

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

Intuition [Baina etal CAISE04, Berardi etal VLDB-TES04]

1. Each transition corresponds to a WS-BPEL pattern consisting of (i) an `<onMessage>` operation (in order to wait for the input from the client of the composite service), (ii) followed by the effective logic of the transition, and then (iii) a final operation for returning the result to the client. Of course both before the effective logic and before returning the result, messages should be copied forth and back in appropriate variables
2. All the transitions originating from the same state are collected in a `<pick>` operation, having as many `<onMessage>` clauses as transitions originating from the state
3. The WS-BPEL file is built visiting all the nodes of the graph, starting from the initial state and applying the previous rules.

N.B.: (1) and (2) works for in-out interactions (the ones shown in the following). Simple modifications are needed for in-only, robust-in-only and in-optional-out. The other kinds of interactions implies a proactive behaviour of the composite service, possibly guarded by `<onAlarm>` blocks.

Transition Skeletons


```
<onMessage … >
      <sequence>
            <assign>
               <copy>
                  <from variable="input" … />
                  <to variable="transitionData" … />
               </copy>
            </assign>
            < !-- logic of the transition -->
            <assign>
               <copy>
                  <from variable="transitionData" … />
                  <to variable="output" … />
               </copy>
            </assign>
            <reply ... />
      </sequence>
</onMessage>
```


State Skeletons

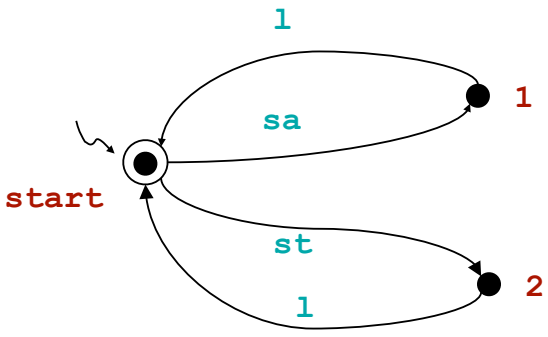
- N transitions from state S_i are mapped onto:

```
<pick name = "Si">
      <!-- transition #1 -->
      <onMessage … >
            <!-- transition skeleton -->
      </onMessage>
      … … …
      <!-- transition #N -->
      <onMessage … >
            <!-- transition skeleton -->
      </onMessage>
```

Mapping the TS

- All the <pick> blocks are enclosed in a surrounding <flow>; the dependencies are modeled as <link>s
  - <link>s are controlled by specific variables s_i-to-s_j that are set to TRUE iff the transition $S_i \rightarrow S_j$ is executed
  - Each state skeleton has many outgoing <link>s as states connected in output, each going to the appropriate <pick> block
  - Transitions going back into the initial state should not be considered, as they can be represented as the

An Example (1)




```
<partnerLinks>
  <!-- The "client" role represents the requester of this composite service -->
      <partnerLink name="client"
                   partnerLinkType="tns:Transition"
                   myRole="MP3ServiceTypeProvider"
                   partnerRole="MP3ServiceTypeRequester"/>
      <partnerLink name="service"
                   partnerLinkType="nws:MP3CompositeService"
                   myRole="MP3ServiceTypeRequester"
                   partnerRole="MP3ServiceTypeProvider"/>
</partnerLinks>
```

An Example (2)

```
<variables>
     <variable name="input" messageType="tns:listen_request"/>
     <variable name="output" messageType="tns:listen_response"/>
     <variable name="dataIn" messageType="nws:listen_request"/>
     <variable name="dataOut" messageType="nws:listen_response"/>
</variables>

      <pick>
         <onMessage partnerLink="client"
                  portType="tns:MP3ServiceType"
                  operation="listen"
                  variable="input">
                  <sequence>
                       <assign>
                            <copy>
                                 <from variable="input" part="selectedSong"/>
                                 <to variable="dataIn" part="selectedSong"/>
                            </copy>
                       </assign>
                       … …
                       <assign>
                            <copy>
                                 <from variable="dataOut" part="MP3FileURL"/>
                                 <to variable="output" part="MP3FileURL"/>
                            </copy>
                       </assign>
                       <reply name="replyOutput"
                             partnerLink="client"
                             portType="tns:MP3ServiceType"
                             operation="listen"
```


An Example (3)

A new instance is created in the initial state. This resolve also the presence of the cycles without the need of enclosing `<while>`

The <sa> transition skeleton should set variables:
`start-to-1 = TRUE`
`start-to-2 = FALSE`

The <st> transition skeleton should set variables:
`start-to-1 = FALSE`
`start-to-2 = TRUE`

```
<process suppressJoinFailure = "no">
   <flow>
   <links>
      <link name="start-to-1"/>
      <link name="start-to-2"/>
    </links>

   <pick createInstance = "yes">
      <onMessage="sa">
         <sequence>
            <copy>...</copy>
            … …
            <copy>...</copy>
            <reply ... />

         </sequence>
      </onMessage>
      <onMessage="st">
         <sequence>
            <copy>...</copy>
            … …
            <copy>...</copy>
            <reply ... />
         </sequence>
      </onMessage>
      <source linkName="start-to-1" transitionCondition = "bpws:getVariableData('start-to-1') = 'TRUE' " />
      <source linkName="start-to-2" transitionCondition = "bpws:getVariableData('start-to-2') = 'TRUE' " />
   </pick>
```

An Example (4)


```
<pick>
  <onMessage="I">
        <sequence>
           <copy>...</copy>
           … …
          <copy>...</copy>
          <reply ... />
        </sequence>
  </onMessage>
  <target linkName="start-to-1" />
</pick>
<pick>
  <onMessage="I">
        <sequence>
           <copy>...</copy>
           … …
          <copy>...</copy>
          <reply ... />
        </sequence>
  </onMessage>
  <target linkName="start-to-2" />
</pick>
</process>
```


Choreography

(As Reported in Literature: Classical Ballet

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Consider a dance with more than one dancer
  - Each dancer has a set of steps that they will perform. They orchestrate their own steps because they are in complete control of their domain (their body)
  - A choreographer ensures that the steps all of the dancers make is according to some overall, pre-defined scheme. This is a choreography
  - The dancers have no control over the steps they make: their steps must conform to the choreography
  - The dancers have a single view-point of the dance
  - The choreographer has a multi-party or global view-point of the dance

Choreography

(A Possible Evolution: Jam Session Style)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Consider a jazz band with many players
  - There is a rhythm and a main theme. This is the choreography
  - Each player executes his piece by improvising variations over the main theme and following the given rhythm
  - The players still have a single view-point of the music; in addition they have full control over the music they play
  - There is a multi-party or global view-point of the music, but this is only a set of "sketchy" guidelines

WS-BPEL vs. WS-CDL

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Orchestration/WS-BPEL is about describing and executing a single peer
- Choreography/WS-CDL is about describing and guiding a global model (N peers)
- You should derive the single peer from the global model by projecting based on participant

WS-CDL Basics (1)

- Participants & Roles
  - Role type
    - Enumerate the observable behavior that a collaborating participant exhibits
    - Behavior type specifies the operations supported
      - Optional WSDL interface type
  - Relationship type
    - Specify the mutual commitments, in terms of the Roles/ Behavior types, **two** collaborating participants are required to provide
    - Note: all multi-party relationships are transformed into binary ones
  - Participant type
    - Enumerate a set of one or more Roles that a collaborating participant plays

WS-CDL Basics (2)

- Channels
  - A channel realizes a dynamic point of collaboration, through which collaborating participants interact
    - Where & how to communicate a message
      - Specify the Role/Behavior and the Reference of a collaborating participant
      - Identify an Instance of a Role
    - Identify an instance of a conversation between two or more collaborating participants
      - A conversation groups a set of related message exchanges
- One or more channel(s) MAY be passed around from a Role to one or more other Role(s), possibly in a daisy fashion through one or more intermediate Role(s), creating new points of collaboration dynamically
  - A Channel type MAY restrict the types of Channel(s) allowed to be exchanged between the Web Services participants, through this Channel

WS-CDL Basics (3)

- Activities are the building blocks of a choreography
  - Basic Activity
    - **Interaction**: message exchange between participants
      - Only in-out and in-only
    - **Assign**: within one role, assign the value of a variable to another one
      - Variables can be about information (exchanged documents), states and channels
    - **No action**: do null
  - Ordering structure
    - **Sequence** (P.Q)
    - **Parallel** (P | Q)
    - **Choice** (P + Q)
  - **Perform**: a complete, separately defined choreography is performed
    - Basis for scalable modeling

Attention: a choreography performing another one is referred to as "choreography composition" in the standard

WS-CDL Basics (4)

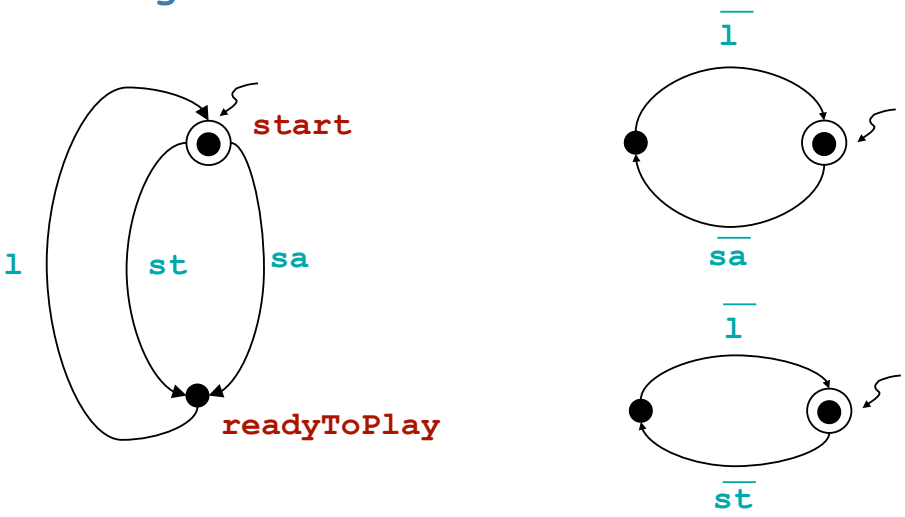
- A Choreography combines all previous elements, forming a collaboration unit of work
  - Enumerate all the binary relationships interactions act in
  - Localize the visibility of variables
    - Using variable definitions
  - Prescribe alternative patterns of behavior
    - Using work/units and reactions
  - Enable Recovery
    - Using work/units and reactions
    - Backward: handle exceptional conditions
    - Forward: finalize already completed activities

TSs and Choreography

(only an intuition :-))

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- A Choreography can be seen as the specification of a set of concurrent peers, each one exposing a TS, that fulfills the global model

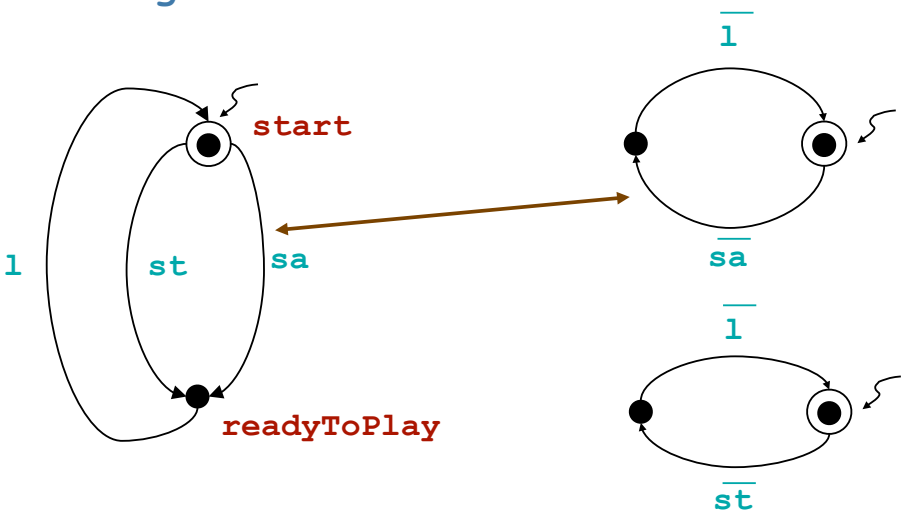


TSs and Choreography

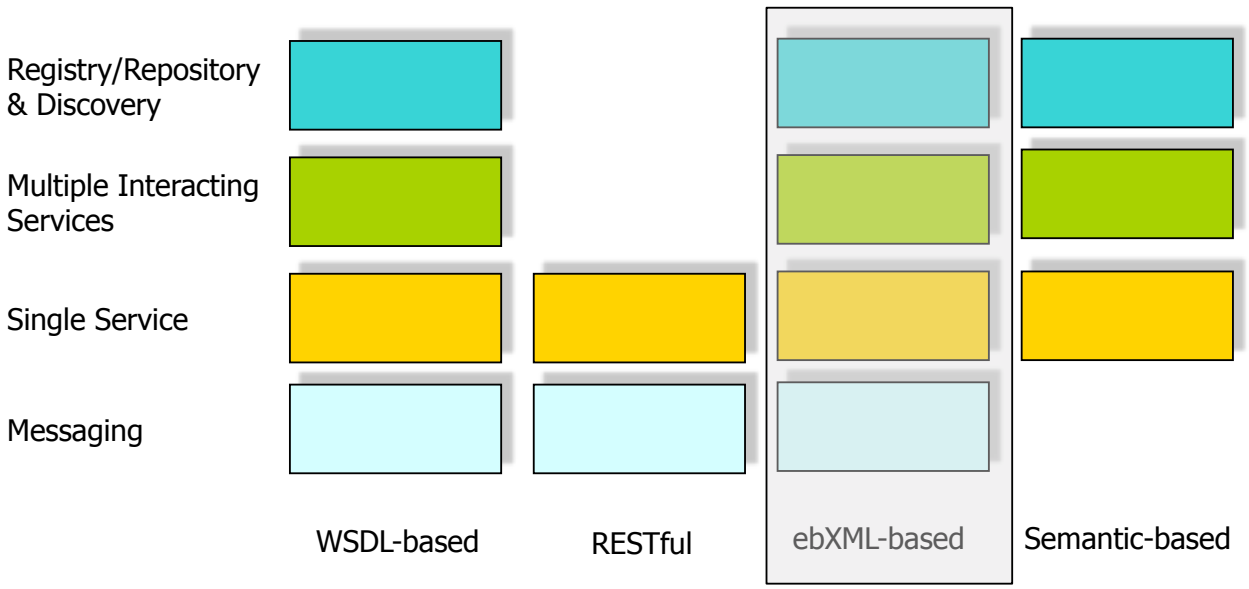
(only an intuition :-))

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- A Choreography can be seen as the specification of a set of concurrent peers, each one exposing a TS, that fulfills the global model



The "Stacks" of Service Technologies



RESTful Services (1)

- REST refers to simple application interfaces transmitting data over HTTP without additional layers as SOAP
  - Web page meant to be consumed by program as opposed to a Web browser or similar UI tool
  - require an architectural style to make sense of them (the REST one), because there's no smart human being on the client end to keep track

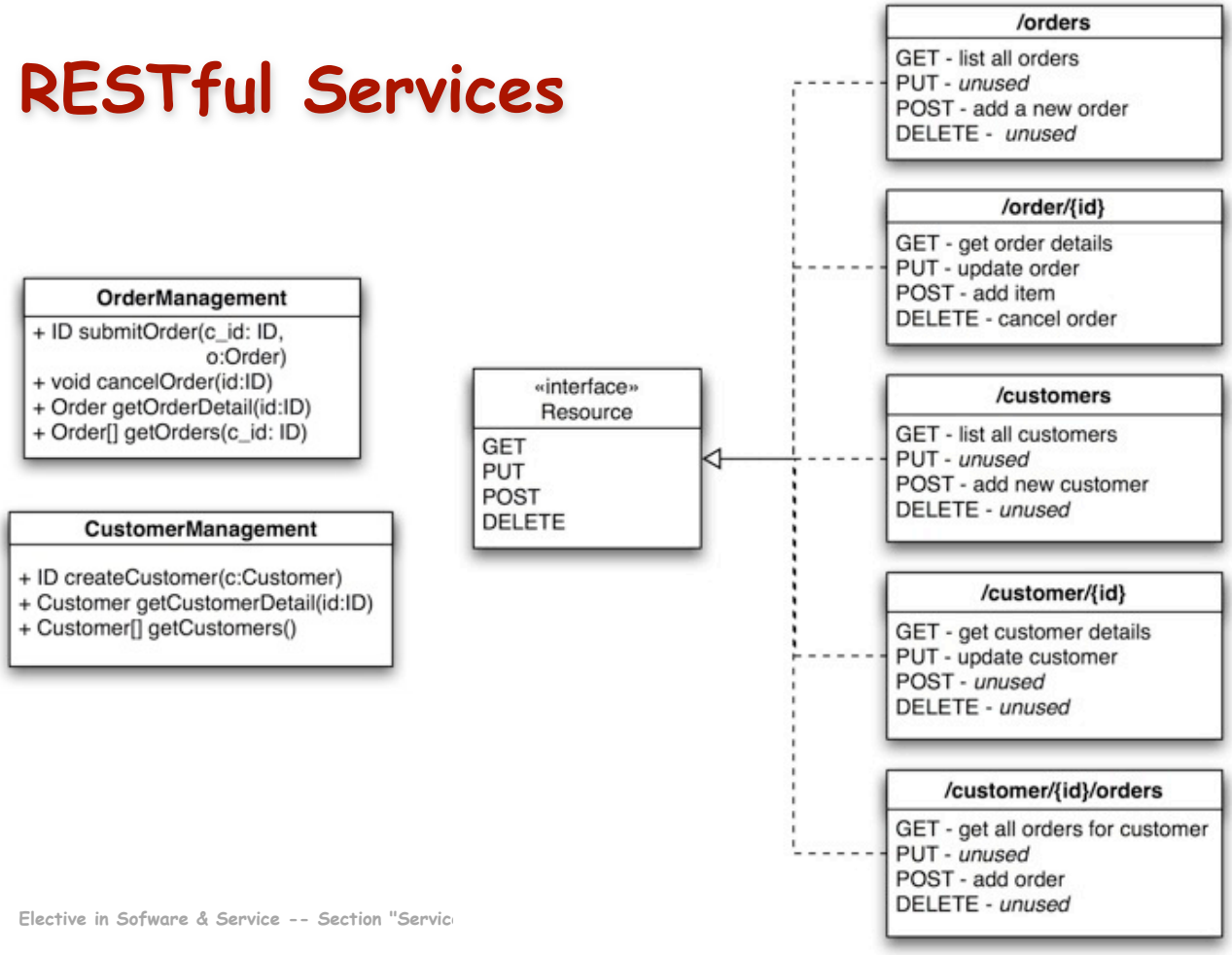
RESTful Services (2)

- Metaphor based on nouns and verbs
  - URIs ~ nouns
  - Verbs describe actions that are applicable to nouns
    - GET -- retrieve information / READ, SELECT
    - POST (PUT) – add/update new information / CREATE, INSERT, UPDATE
    - DELETE -- discard information / DELETE
- State means the application/session state, maintained as part of the content transferred (in XML) from client to server back to client

RESTful Services (3)

- REST is, in a sense, a kind of RPC, except the methods have been defined in advance
  - Consider the stock example of a remote procedure called "getStockPrice"
  - It's not clear what what it means to GET, PUT, and POST to something called "getStockPrice"
  - But if we change the name from "getStockPrice" to "CurrentStockPrice" all is well !!

RESTful Services



RESTful Services (4)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- REST is incompatible with "end-point" RPC -- Either you address data objects or you address "software components"
  - REST does the former
  - End-point RPC does the latter

```
POST /purchase_orders HTTP/1.1
Host: accounting.mycompany.com
content-type:
application/purchase-order+xml
....
<po>...</po>
```

```
POST /generic_message_handler
content-type: application/SOAP+XML
<soap:envelope>
   <soap:body>
     <submit-purchase-order>
       <destination>accounting.mycompany.com
       </destination>
       <po>...</po>
     </submit-purchase-order>
   </soap:body>
<soap:envelope>
```

Example (1)

Operation	HTTP Request	HTTP Response	Java Technology Method
Create	POST /restfulwebservice-war/poservice/ HTTP/1.0 Accept: */* Connection: close Content-Type: text/xml Content-Length: 618 Pragma: no-cache <tns:PurchaseOrderDocument xmlns:tns="urn:PurchaseOrderDocument"> <billTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <poID>ABC-CO-19282</poID> <items> <itemname>Copier Paper</itemname> <price>10</price> <quantity>2</quantity> </items> <items> <itemname>Toner</itemname> <price>920</price> <quantity>1</quantity> </items> <shipTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </shipTo> </tns:PurchaseOrderDocument>	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Date: Fri, 21 Jul 2006 17:07:15 GMT Connection: close <?xml version="1.0" encoding="UTF-8"?> <ns2:Status xmlns:ns2="urn:Status" xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns4="urn:POProcessingFault"> <orderid>ABC1153501634787</orderid> <timestamp>Fri Jul 21 13:07:14 EDT 2006</timestamp> </ns2:Status>	public PurchaseOrderStatus acceptPO(PurchaseOrder order)

Example (2)

Read	GET /restfulwebservice-war/poservice/ABC1153501634787 HTTP/1.0 Connection: close Content-Type: text/xml	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Connection: close <?xml version="1.0" encoding="UTF-8"?><ns3:PurchaseOrderDocument xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns2="urn:Status" xmlns:ns4="urn:POProcessingFault"><billTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode></billTo><createDate>2006-07-21T13:08:37.505-04:00</createDate><items><itemname>Copier Paper</itemname><price>10</price><quantity>2</quantity></items><items><itemname>Toner</itemname><price>920</price><quantity>1</quantity></items><poID>ABC1153501634787</poID><shipTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode></shipTo></ns3:PurchaseOrderDocument>	public PurchaseOrder retreivePO (String orderID)
	GET /restfulwebservice-war/poservice/ HTTP/1.1 Connection: keep-alive	HTTP/1.1 400 Bad Request X-Powered-By: Servlet/2.5 Content-Type: text/xml <?xml version="1.0" encoding="UTF-8"?><ns4:POProcessingFault xmlns:ns4="urn:POProcessingFault" xmlns:ns2="urn:Status" xmlns:ns3="urn:PurchaseOrderDocument"><message>Unable to retrieve the order associated with the orderid you specified</message></ns4:POProcessingFault>	Indicates a problem finding the order

Example (3)

Update	PUT /restfulvebservice-var/poservice/ HTTP/1.0 Connection: close Content-Type: text/xml Content-Length: 620 Pragma: no-cache <tns:PurchaseOrderDocument xmlns:tns="urn:PurchaseOrderDocument"> <billTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <poID>ABC-CO-19282</poID> <items> <itemname>Copier Paper</itemname> <price>10</price> <quantity>2</quantity> </items> <items> <itemname>Toner</itemname> <price>920</price> <quantity>1</quantity> </items> <shipTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </shipTo> </tns:PurchaseOrderDocument>	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml <?xml version="1.0" encoding="UTF-8"?><ns3:PurchaseOrderDocument xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns2="urn:Status" xmlns:ns4="urn:POProcessingFault"> <billTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <items><itemname>Copier Paper</itemname> <price>10</price><quantity>2</quantity></items> <items><itemname>Toner</itemname><price>920</price> <quantity>1</quantity></items> <poID>ABC-CO-19282</poID><shipTo><street>1 Main Street</street><city>Beverly Hills</city> <state>CA</state><zipCode>90210</zipCode></shipTo> </ns3:PurchaseOrderDocument>	public PurchaseOrder updatePO(PurchaseOrder order)
Delete	DELETE /restfulvebservice-var/poservice/ABC-CO-19282 HTTP/1.0 Connection: close Content-Type: text/xml Content-Length: 0 Pragma: no-cache	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Date: Fri, 21 Jul 2006 17:10:38 GMT Server: Sun Java System Application Server Platform Edition 9.1 Connection: close <?xml version="1.0" encoding="UTF-8"?>	public void cancelPO(String orderID)

Why so trendy ?

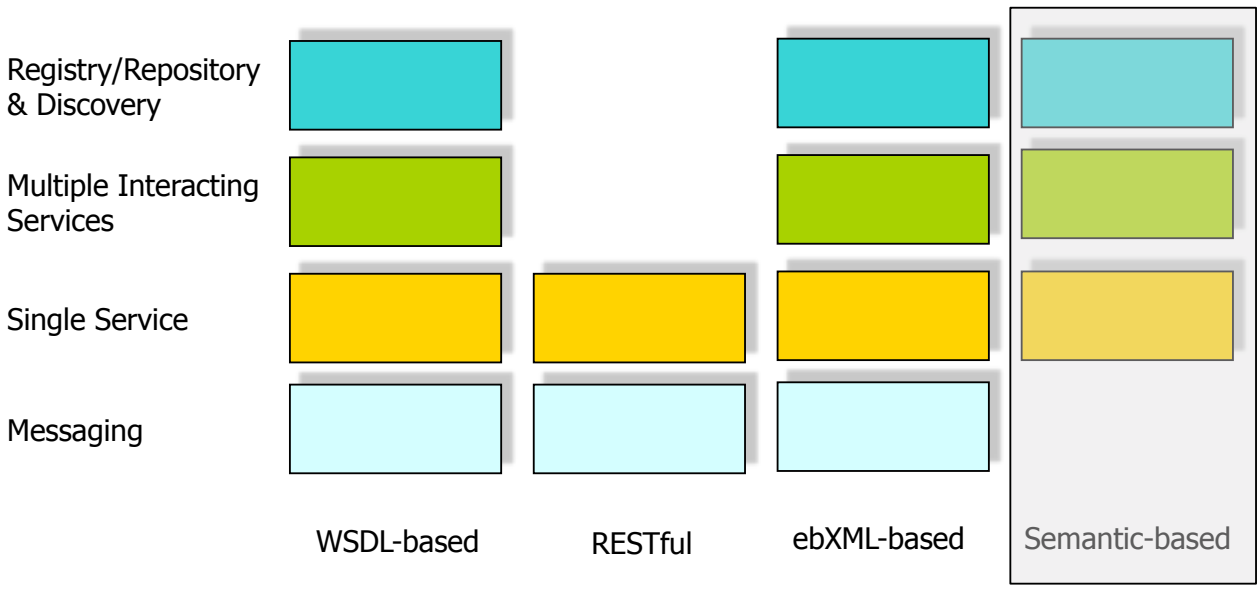
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Easy and lightweight
- Amazon, Yahoo, Google offer their Web services as RESTful

- … but nothing really new for us, basically the same abstractions apply, you can consider the operations as a whole or you can start modeling the data flowing through the service

The "Stacks" of Service Technologies

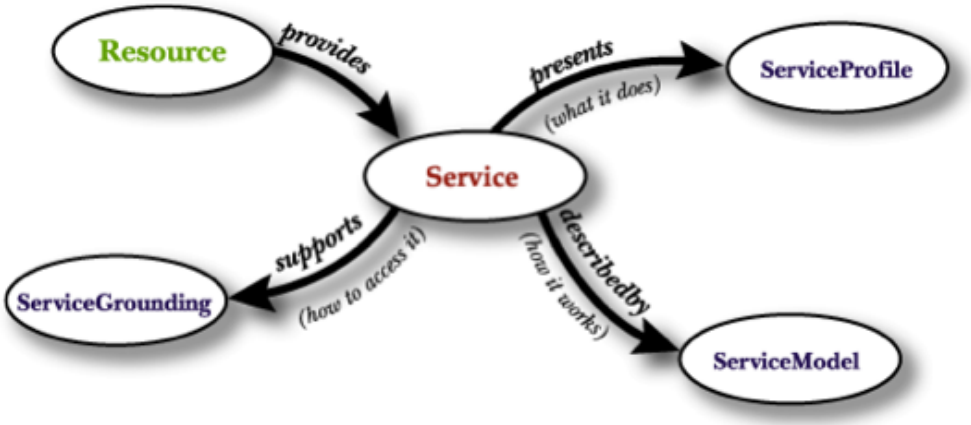
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



OWL-S (formely DAML-S)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Add semantics
  - An upper ontology for describing properties & capabilities of Web Services using OWL
- Enable automation of various activities (e.g., service discovery & selection)



[from DAML-S]

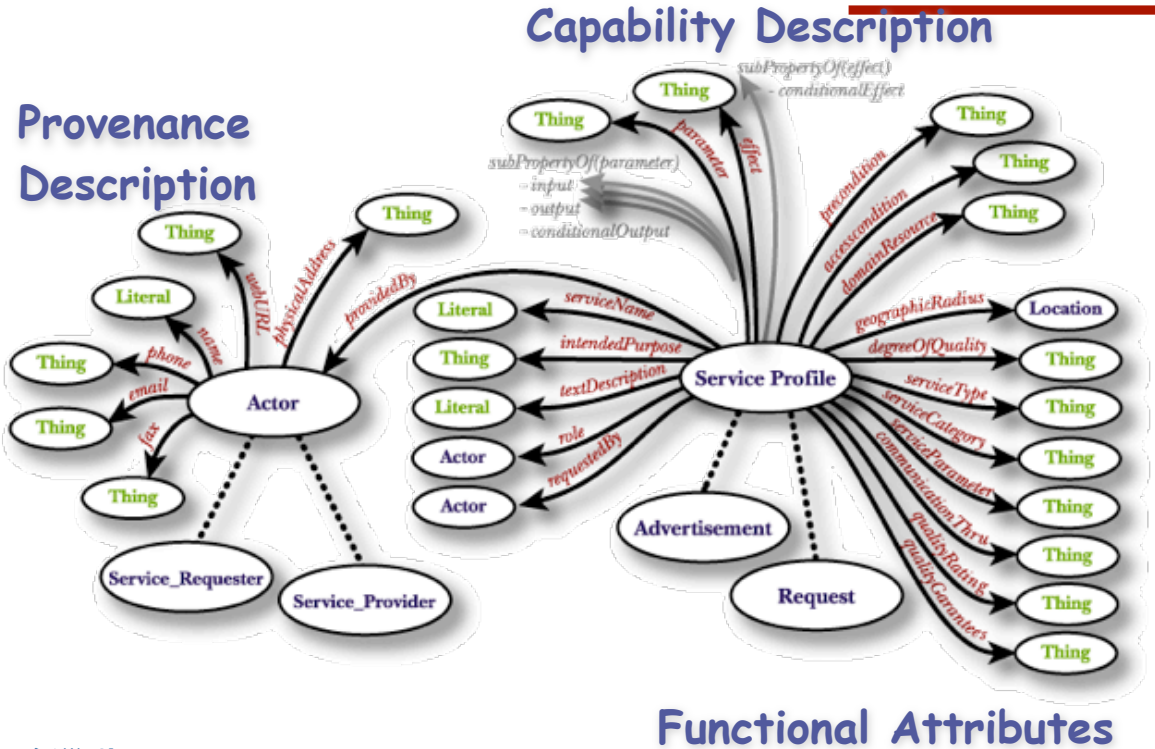
OWL-S Service Profile
(What it does)

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- High-level characterization/summary of a service
  - Provider & participants
  - Capabilities
  - Functional attributes (e.g., QoS, region served)
- Used for
  - Populating service registries
    - A service can have many profiles
  - Automated service discovery
  - Service selection (matchmaking)
- One can derive:
  - Service advertisements
  - Service requests

OWL-S Service Profile

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

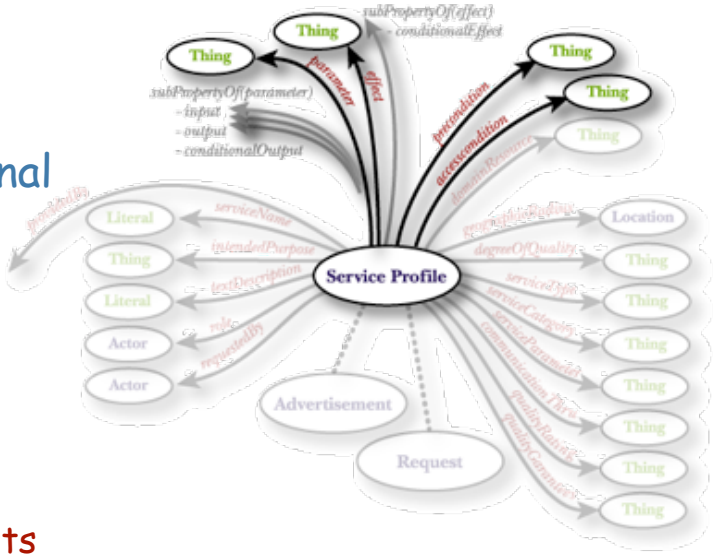


[from DAML-S]

Capability Description

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Specification of what the service provides
  - High-level functional representation in terms of:
    - preconditions
    - inputs
    - (conditional) outputs
    - (conditional) effects



IOPE

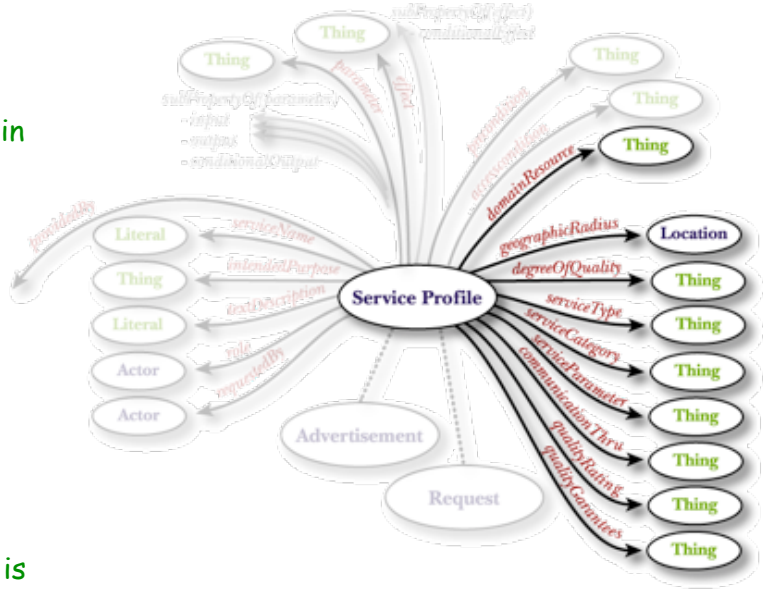
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Inputs
  - Set of necessary inputs that the requester should provide to invoke the service
- (Conditional) Outputs
  - Results that the requester should expect after interaction with the service provider is completed
- Preconditions
  - Set of conditions that should hold prior to service invocation
- (Conditional) Effects
  - Set of statements that should hold true if the service is invoked successfully
  - Often refer to real-world effects, e.g., a package being delivered, or a credit card being debited

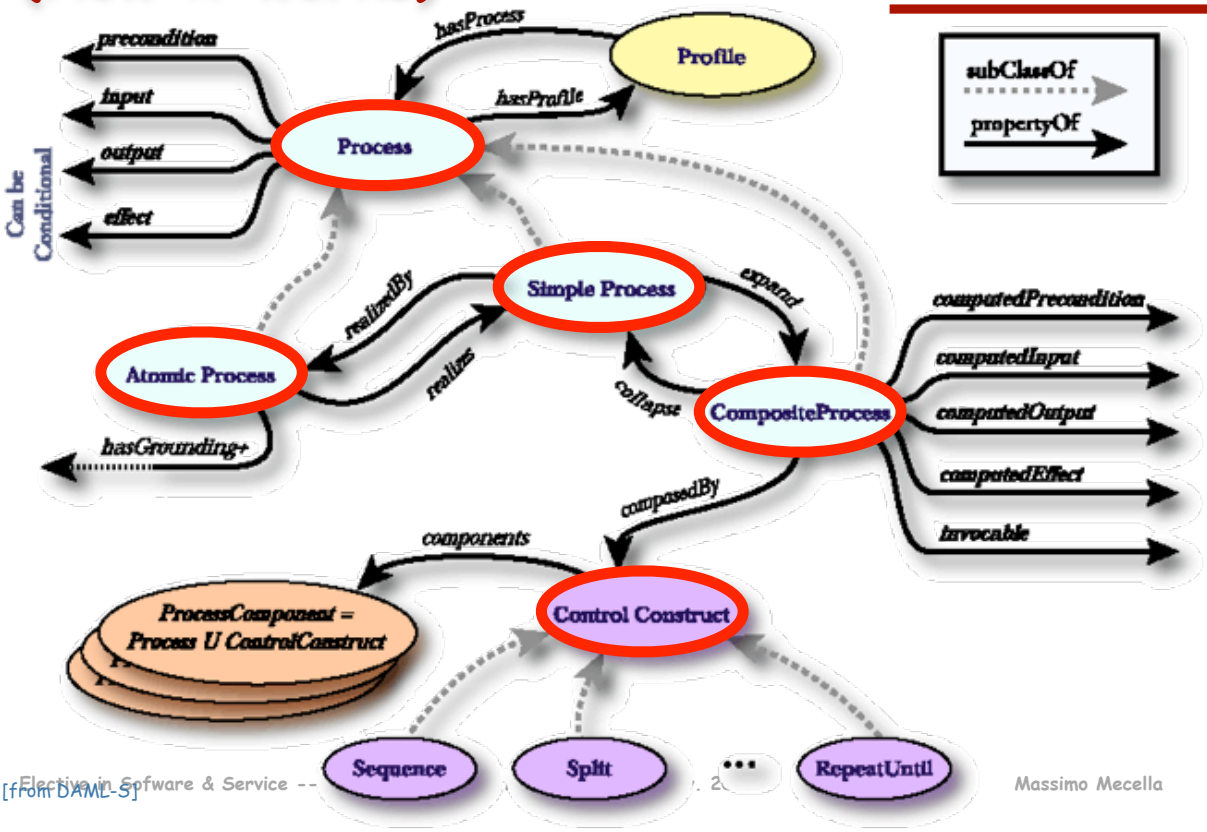
Functional Attributes

Provide supporting information about the service, including:

- geographical scope
  - Pizza Delivery only within the Pittsburgh area
- quality descriptions and guarantees
  - Stock quotes delivered within 10 secs
- service types, service categories
  - Commercial / Problem Solving, etc.
- service parameters
  - Average Response time is currently ...



OWL-S Service Model (How it works)



[from DAML-S]

OWL-S Process Ontology

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Atomic processes: directly invokable, no subprocesses, executed in a single step
- Composite processes: consist of other (non-composite or composite) processes
- Simple processes: a virtual view of atomic process or composite process (as a "black box")

Process Model

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Constructs for complex processes
  - Sequence
  - Concurrency: Split; Split+Join; Unordered
  - Choice
  - If-Then-Else
  - Looping: Repeat-Until; Iterate (non-deterministic)
- Data Flow
  - No explicit variables, no internal data store
  - Predicate "sameValues" to match input of composite service and input of subordinate service
- Less refined than, e.g., WS-BPEL

Enhancements

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Recent proposals aim at improving and detailing process modeling and dynamic semantics
  - SWSF (Semantic Web Service Framework)
    - SWSL - Language
    - SWSO - Ontology
- http://www.w3.org/Submission/SWSF/

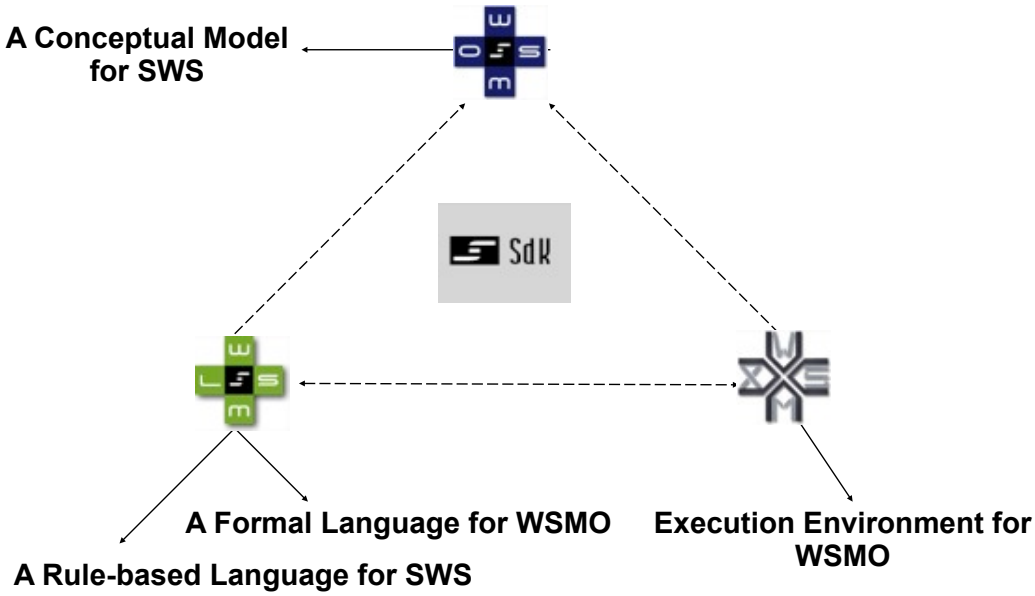
WSMO

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Conceptual model for Semantic Web Services :
  - Ontology of core elements
  - Formal description language (WSML)
  - Execution environment (WSMX)
- … derived from and based on the Web Service Modeling Framework WSMF
- a SDK-Cluster Working Group
  (joint European research and development initiative)

WSMO Working Groups

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

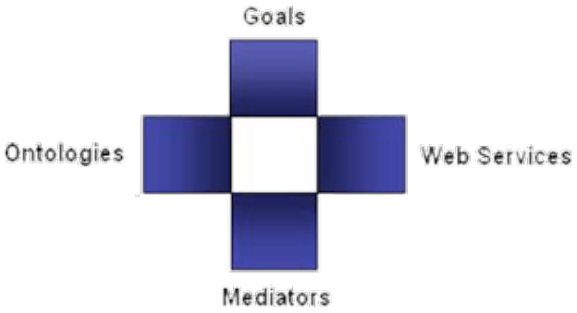


WSMO Top Level Notions

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

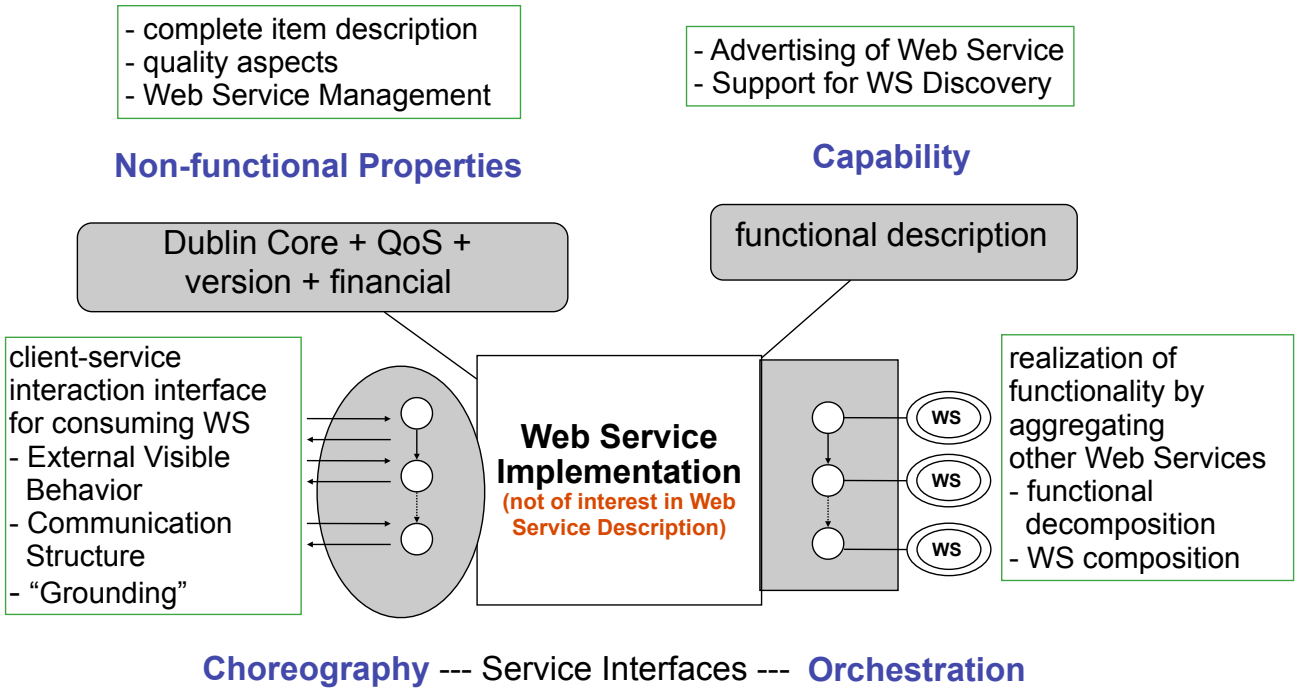


Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)

Connectors between components with mediation facilities for handling heterogeneities

WSMO Web Service Description

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



Capability Specification

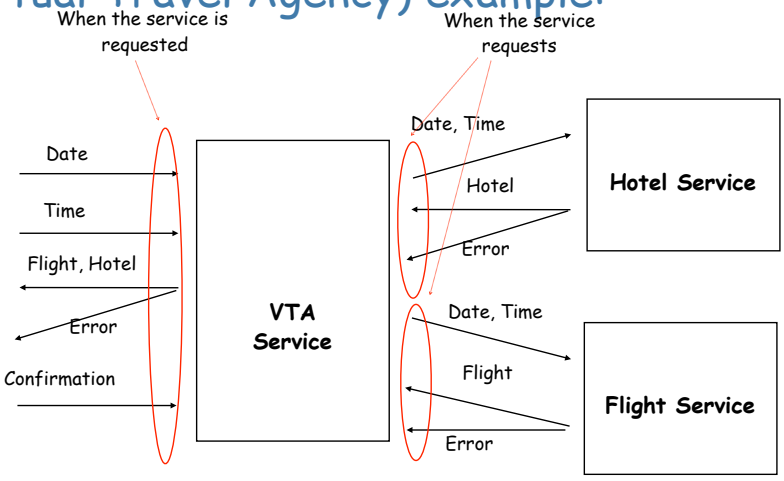
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - OO Mediator: importing ontologies with mismatch resolution
  - WG Mediator: link to a Goal wherefore service is not usable a priori
- **Pre-conditions**
  - What a web service expects in order to be able to provide its service. They define conditions over the input.
- **Assumptions**
  - Conditions on the state of the world that has to hold before the Web Service can be executed
- **Post-conditions**
  - Describes the result of the Web Service in relation to the input, and conditions on it
- **Effects**
  - Conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)

Choreography & Orchestration

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- VTA (Virtual Travel Agency) example:

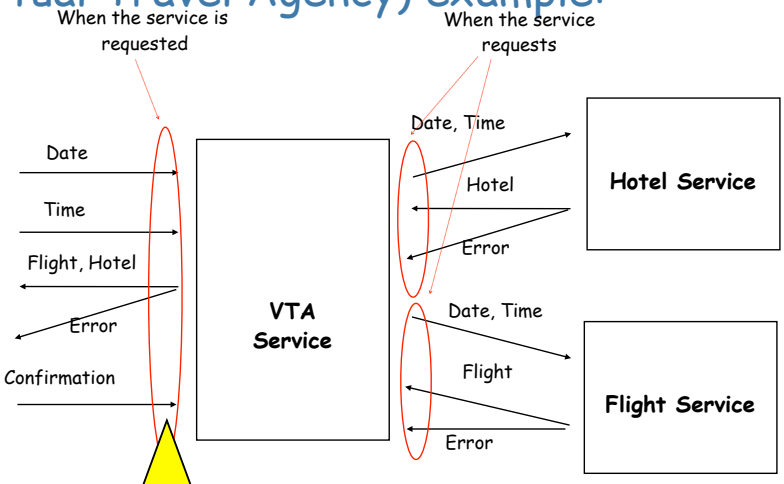


- **Choreography** = how to interact with the service to consume its functionality
- **Orchestration** = how service functionality is achieved by aggregating other Web

Choreography & Orchestration

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- VTA (Virtual Travel Agency) example:



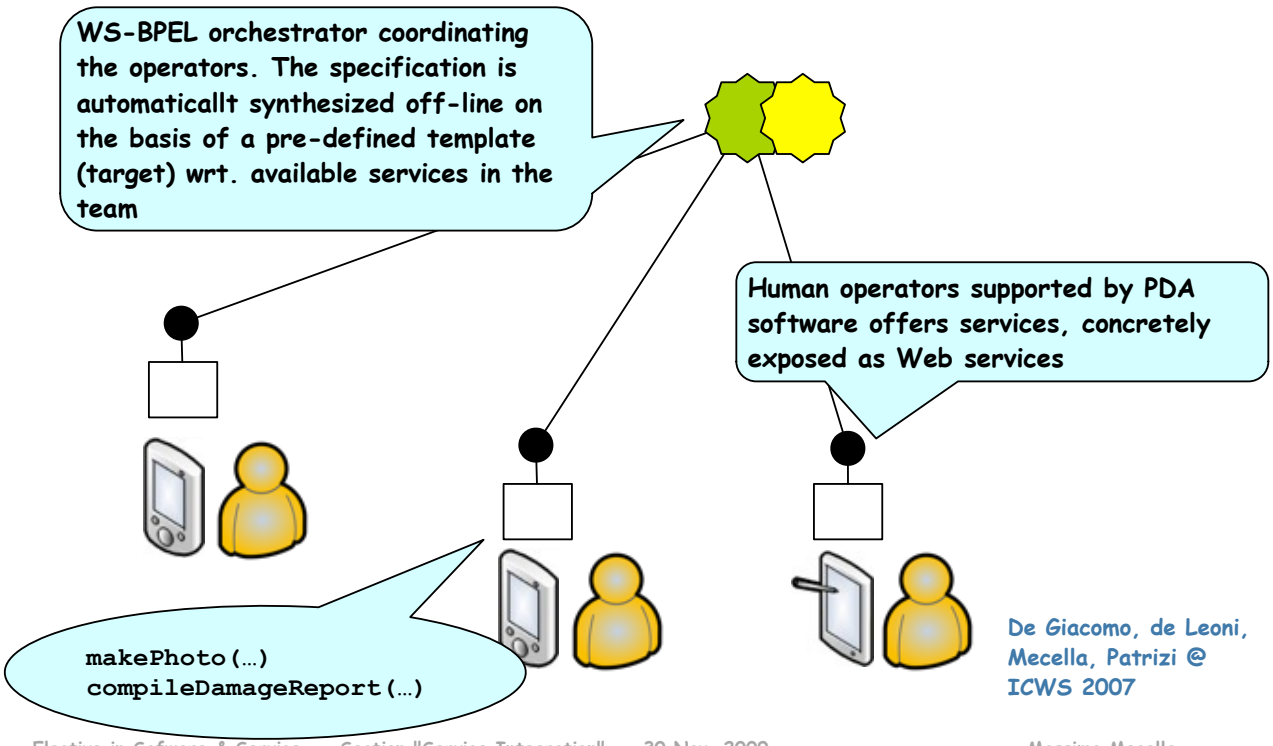
- **Choreography** = w to interact with the service to consume its functionality
- ce functionality is achieved
- y aggregating other Web

What previosly referred as conversation specification

EXAMPLES IN SPECIFIC APPLICATION DOMAINS

Mobile SOA for Emergency Management

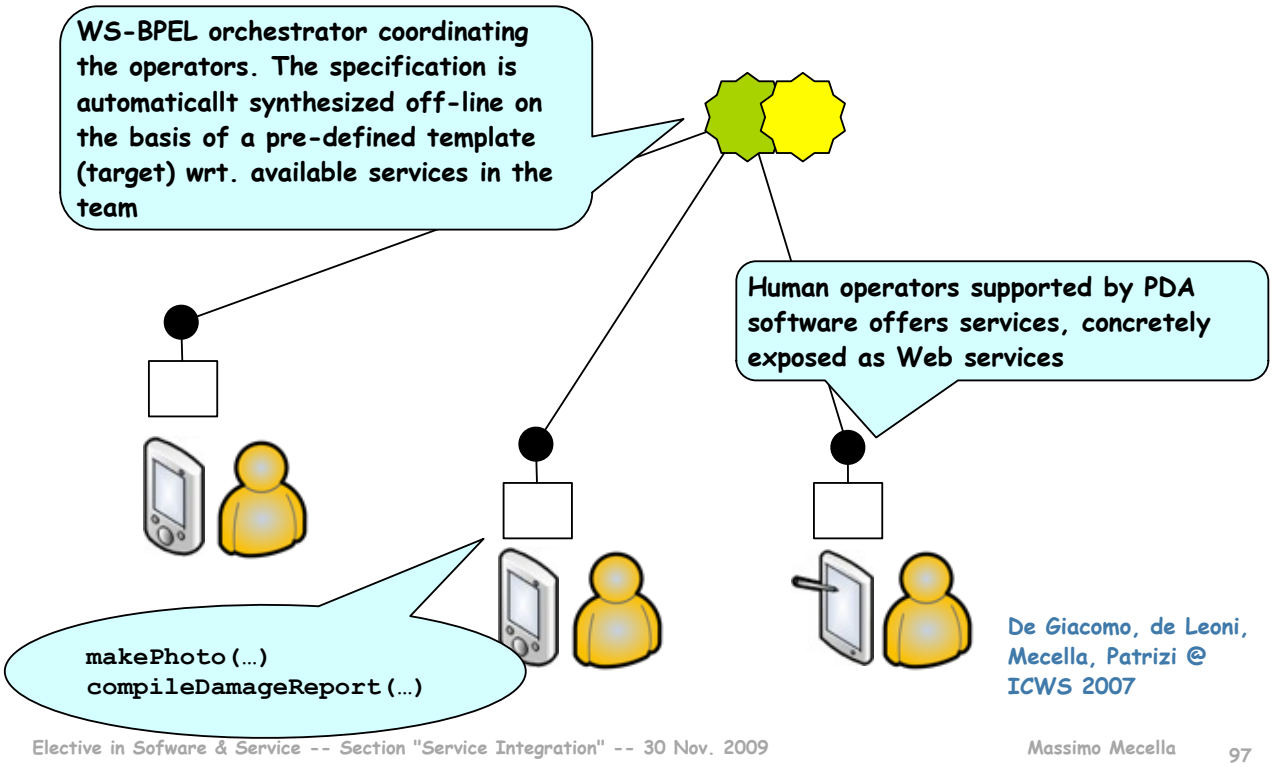
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



De Giacomo, de Leoni, Mecella, Patrizi @ ICWS 2007

Mobile SOA for Emergency Management

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



De Giacomo, de Leoni, Mecella, Patrizi @ ICWS 2007

Embedded SOA for Domotics

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Sensors/devices/actuators/appliances offers services
  - Contextual information is important → need of representing/modeling data
  - The conversations are important
- Clients express goals in terms of "states" of the house they would like to have realized
- The house should synthesize a composition in order to take itself in that state …
- … in a resilient way wrt. failures of the services

References

[ACKM04] - G. Alonso, F. Casati, H. Kuno, V. Machiraju: Web Services. Concepts, Architectures and Applications. Springer-Verlag 2004

[VLDBJ01] - F. Casati, M.C. Shan, D. Georgakopoulos (eds.): Special Issue on e-Services. VLDB Journal, 10(1), 2001
Based on the 1st International Workshop on Technologies for e-Services (VLDB-TES 2001)

[CACM03] – M.P. Papazoglou, D. Georgakopoulos (eds.): Special Issue on Service Oriented Computing. Communications of the ACM 46(10), 2003

[WSOL] - V.Tosic, B. Pagurek, K. Patel, B. Esfandiari, W. Ma: Management Applications of the Web Service Offerings Language (WSOL). To be published in Information Systems, Elsevier, 2004.
An early version of this paper was published in Proc. of CAiSE'03, LNCS 2681, pp. 468-484, 2003

[Benatallah etal IJCIS04] - B. Benatallah, F. Casati, H. Skogsrud, F. Toumani: Abstracting and Enforcing Web Service Protocols, International Journal of Cooperative Information Systems (IJCIS), 13(4), 2004

References

[Baina etal CAISE04] K. Baina, B. Benatallah, F. Casati, F. Toumani: Model-driven Web Service Development, Proc. of CAiSE'04, LNCS 3084, 2004

[Berardi etal ICSOC03] - D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: Proc. of ICSOC'03, LNCS 2910, 2004

[ebpml] - Jean-Jacques Dubray: the ebPML.org Web Site, http://www.ebpml.org/

[DAML-S] – DAML Semantic Web Services, http://www.daml.org/services

References

[WS-Policy] - Web Services Policy Framework (WS-Policy), September 2004, http://www-106.ibm.com/developerworks/library/specification/ws-polfram/

[WSCL] - Web Services Conversation Language (WSCL) 1.0. W3C Note, 14 March 2002, http://www.w3.org/TR/wscl10/

[WSLA] - A. Dan, D. Davis et al: Web Services On Demand: WSLA-driven Automated Management. IBM Systems Journal, 43(1), 2004

[ebXML] - Electronic Business using eXtensible Markup Language, http://www.ebxml.org/

[OASIS] - Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/home/index.php

[WSDL] - R. Chinnici, M. Gudgin, J.J. Moreau, J. Schlimmer, and S. Weerawarana, Web Services Description Language (WSDL) 2.0, Available on line: http://www.w3.org/TR/wsdl20, 2003, W3C Working Draft.

[BPEL4WS] - T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, Business Process Execution Language for Web Services (BPEL4WS) Version 1.1, http://www-106.ibm.com/developerworks/library/

References

[WS-CDL] - N. Kavantzas, D. Burdett, G. Ritzinger, Y. Lafon: Web Services Choreography Description Language (WS-CDL) Version 1.0, Available on line at: http://www.w3.org/TR/ws-cdl-10/, W3C Working Draft.

[UDDI] – Universal Discovery, Description and Integration, http://www.uddi.org/

[WS-C] – Web Services Coordination (WS-C), http://www-106.ibm.com/developerworks/library/ws-coor/

[WS-T] - Web Services Transaction (WS-Transaction), http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/

[WS-CAF] – Web Services Composite Application Framework, http://developers.sun.com/techtopics/webservices/wscaf/