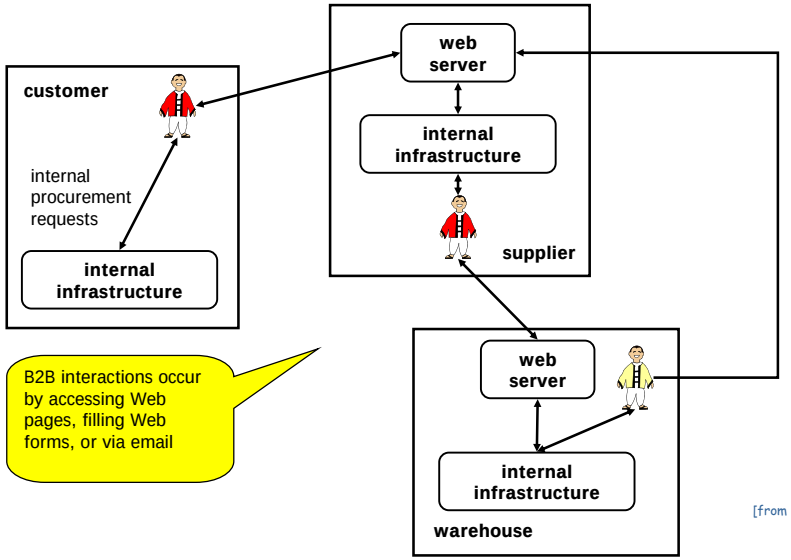
Seminari di Ingegneria del Software
A.A. 2005/2006

A Brief Introduction to Web Services and Related Technologies

Massimo Mecella

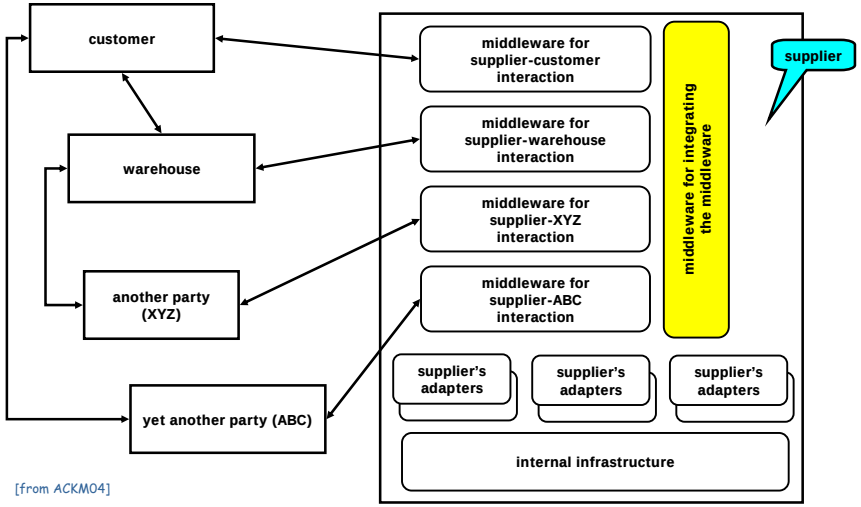
Materiale didattico per questa parte:
- Slide
- Alonso, Casati et al.: Web Services. Springer Verlag, 2004. Cap. 5 – 8
- Berardi et al. @ IJCIS 2005
- Baina et al. @ CAiSE 2004
- Berardi et al. @ AISC 2006

(naive) Business-to-Business Integration



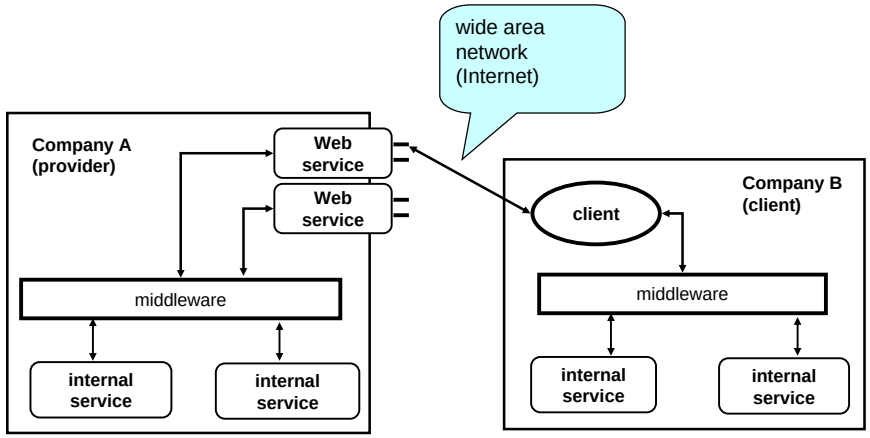
[from ACKM04]

WSs: the Evolution of Middleware and EAI Technologies (1)



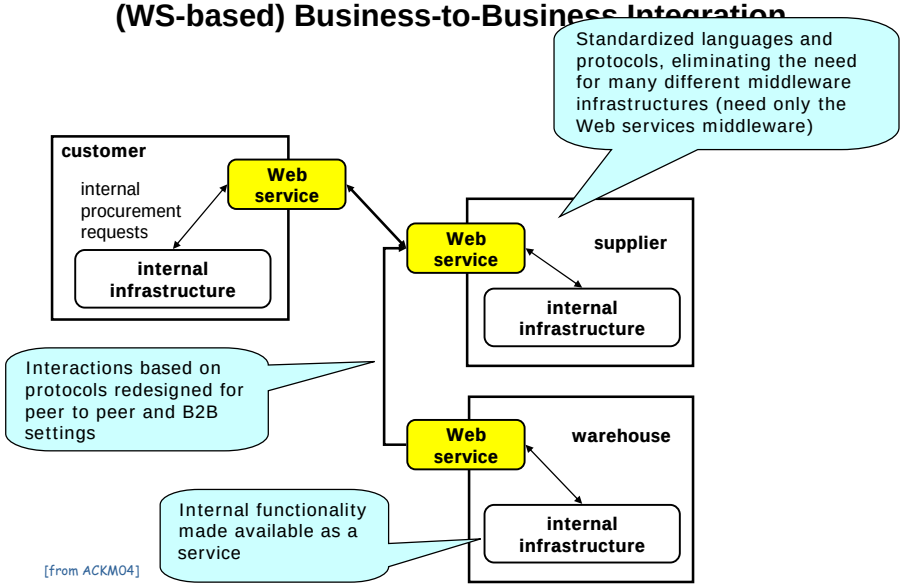
[from ACKM04]

WSs: the Evolution of Middleware and EAI Technologies (2)



[from ACKM04]

(WS-based) Business-to-Business Integration



[from ACKM04]

When Web Services Should Be Applied ?

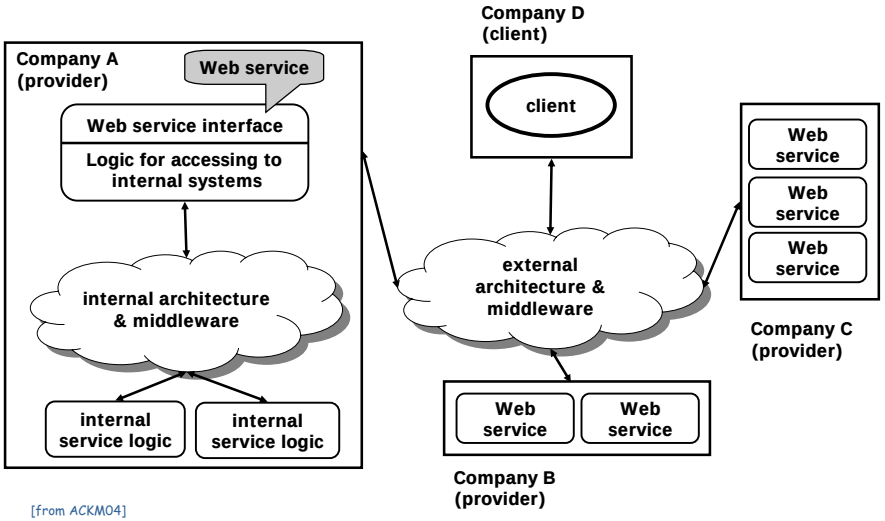
When it is no possible to easily manage deployment so that all requesters and providers are upgraded at once

When components of the distributed system run on different platforms and vendor products

When an existing application needs to be exposed over a network for use by unknown requesters

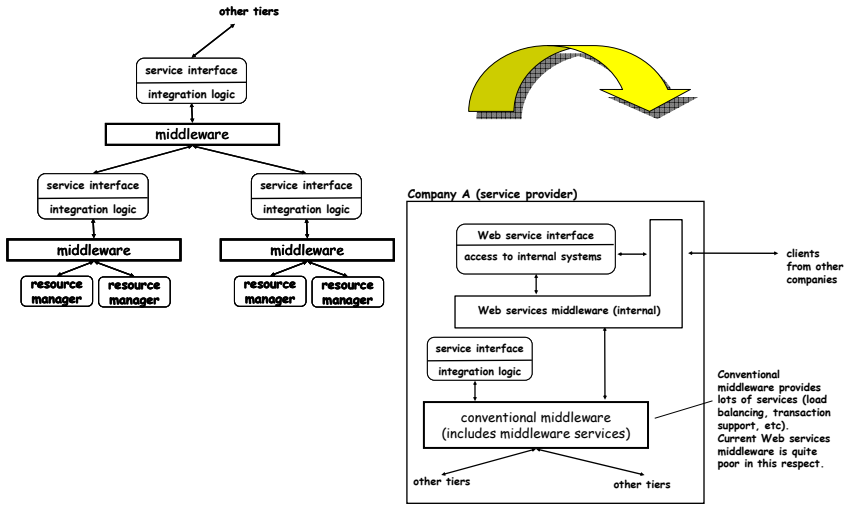
Web Services Architecture,
W3C Working Group Note, 11 Feb. 2004, `http://www.w3.org/TR/ws-arch/`

Two Architectures (and Middlewares) (1)

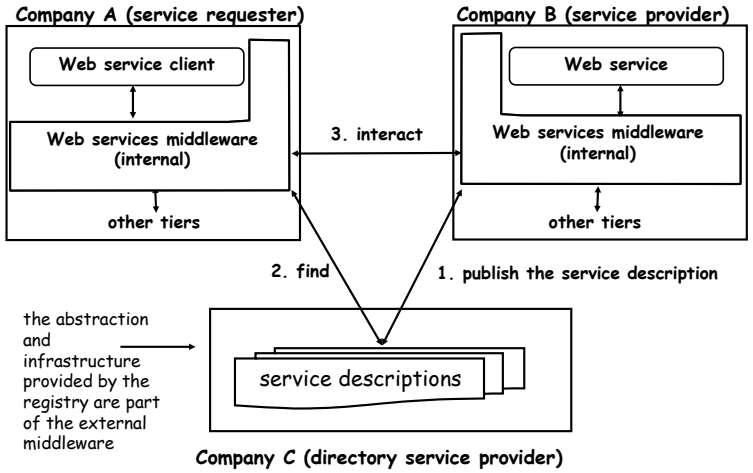


[from ACKM04]

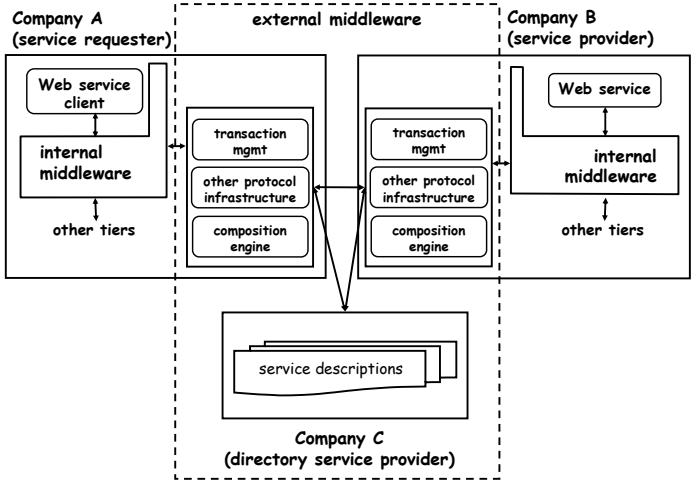
The Internal Architecture



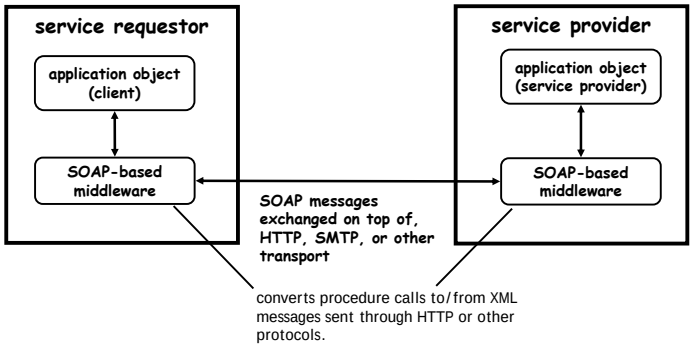
The External Architecture



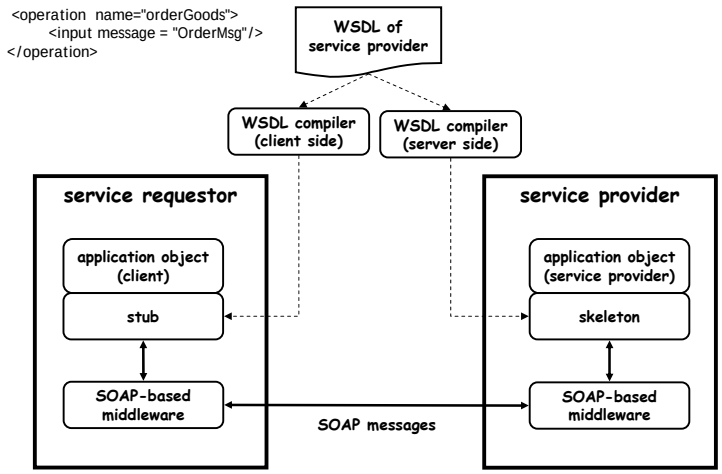
The External Middleware



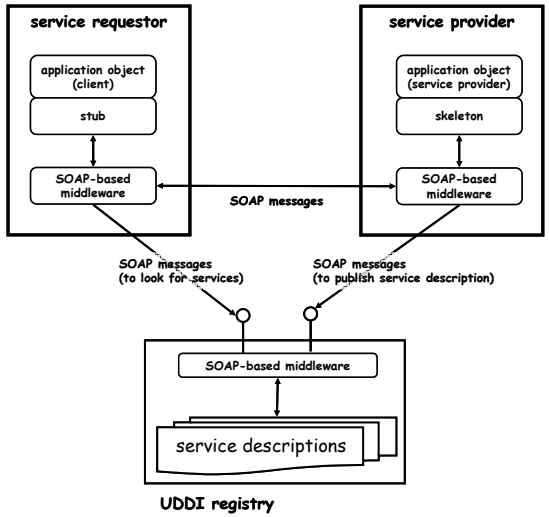
A Minimalist Infrastructure for Web Service



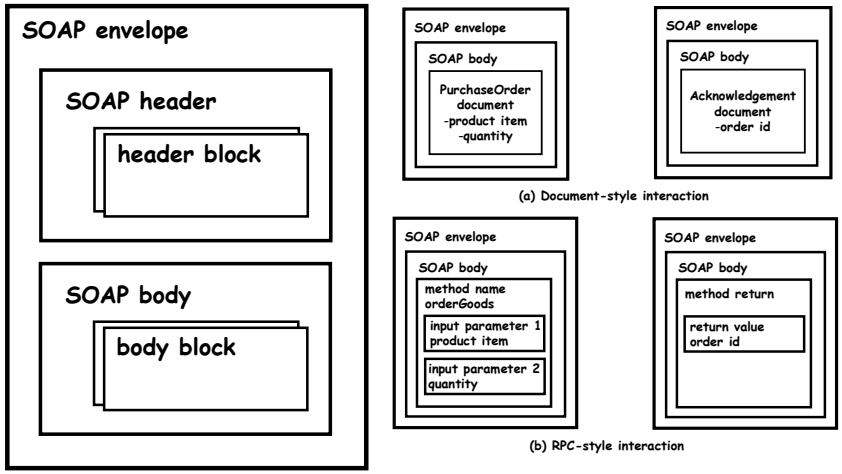
From Interfaces to Stub/Skeleton



Registry



SOAP (1)



SOAP (2)

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

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

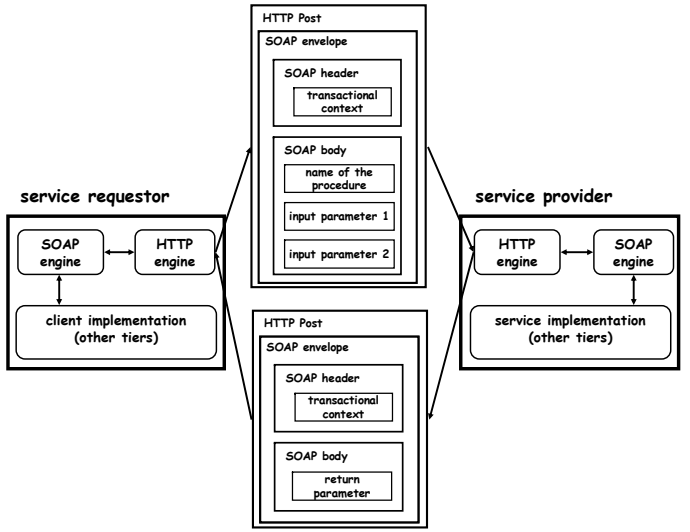
```
<ProductItem name="…"
  <type>…</type>
  <make>…</make>
</ProductItem>
```

Different encoding styles

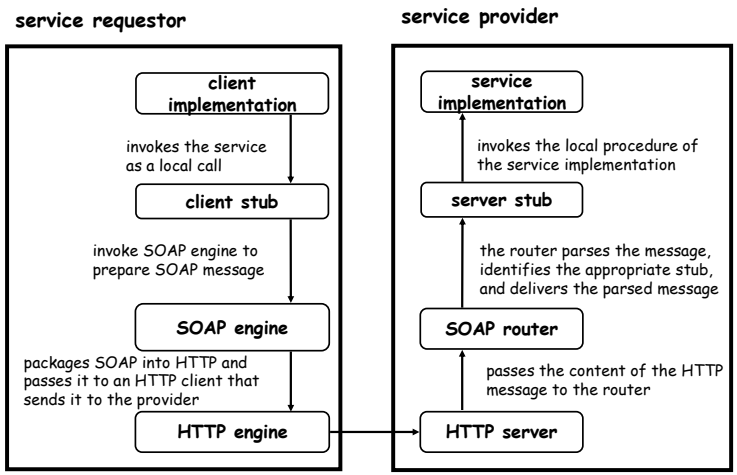
```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
 <env:Header>
  <t:transactionID
          xmlns:t="http://intermediary.example.com/procurement"
          env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
          env:mustUnderstand="true" >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

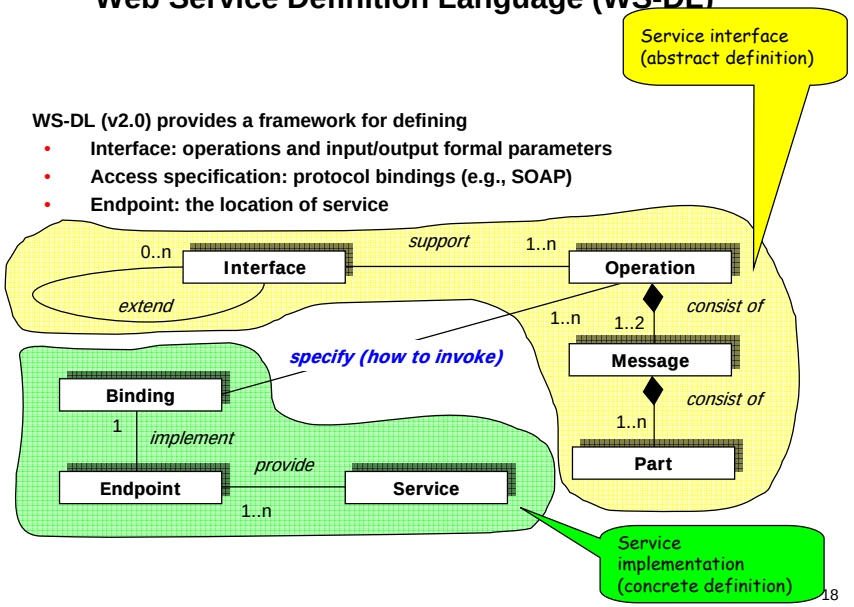
RPC with SOAP



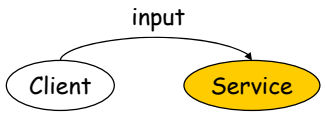
The Simplest SOAP Middleware



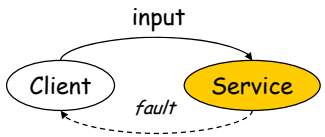
Web Service Definition Language (WS-DL)



Message Exchange Patterns (1)

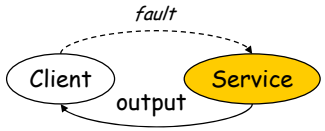


in-only (no faults)



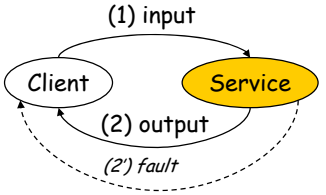
robust in-only (message triggers fault)

out-only (no faults)

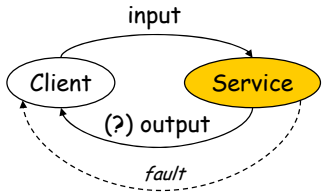


robust out-only (message triggers fault)

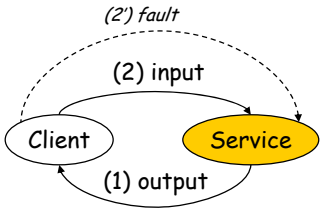
Message Exchange Patterns (2)



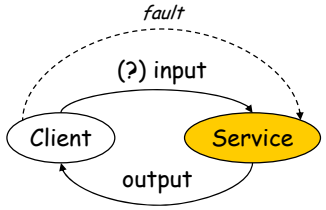
in-out (fault replaces message)



in-optional-out (message triggers fault)



out-in (fault replaces message)



out-optional-in (message triggers fault)

An Example (1)

```
<definitions … >
   <types>
      <element name="ListOfSong_Type">
         <complexType><sequence>
            <element minOccurs="0" maxOccurs="unbound“
               name="SongTitle" type="xs: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>
```

Definition of a message and its formal parameter

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>
```

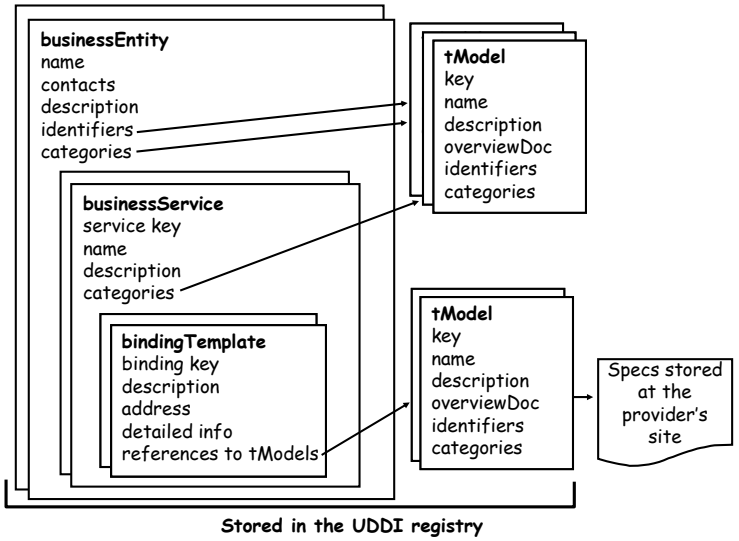

An Exa

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

UDDI Data Structures



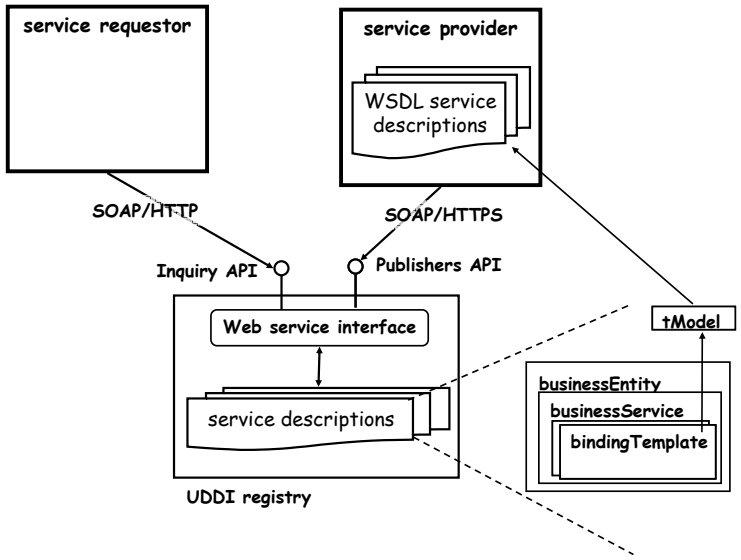
A Registry Not a Repository

overviewDoc (refer to WSDL specs and to API specs)

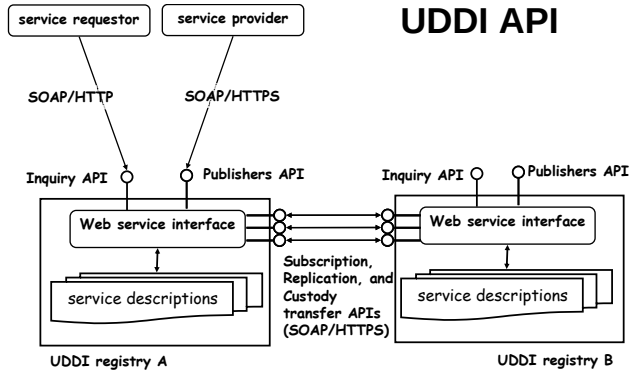
classification information (specifies that this tModel is about XML, WSDL, and SOAP specs)

```
<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

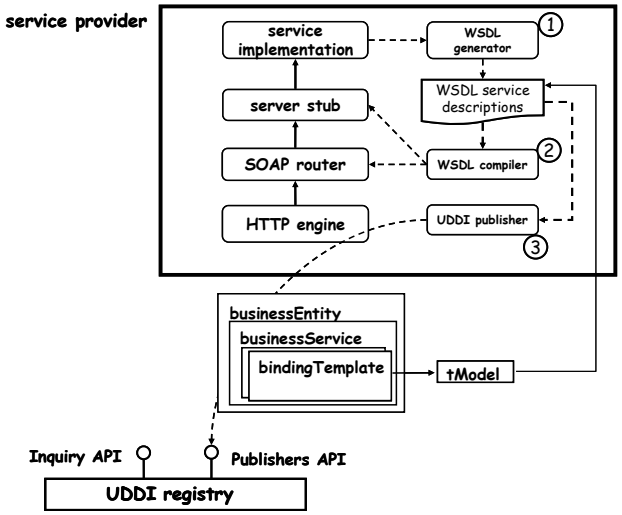


UDDI API

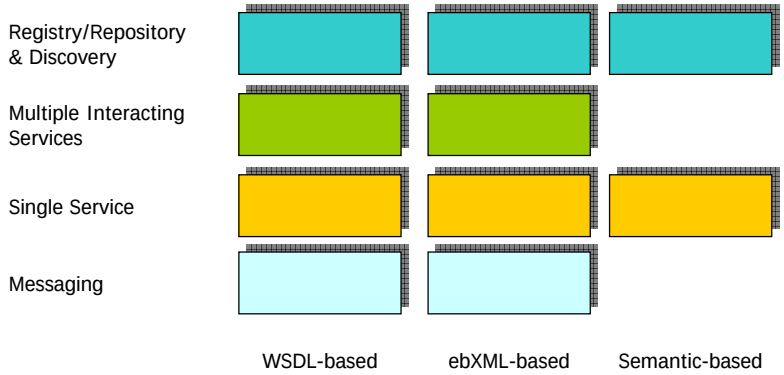



```
<?xml version="1.0"?>
<find_tModel generic="1.0" xmlns="urn:uddi-org:api">
  <categoryBag>
    <keyedReference tModelKey="UUID:C25893AF-1977-3528-36B5-4192C2AB9E2C"
                    keyName="uddi-org:types" keyValue="wsdlSpec"/>
    <keyedReference tModelKey="UUID:A15019C5-AE14-236C-331C-650857AE0221"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

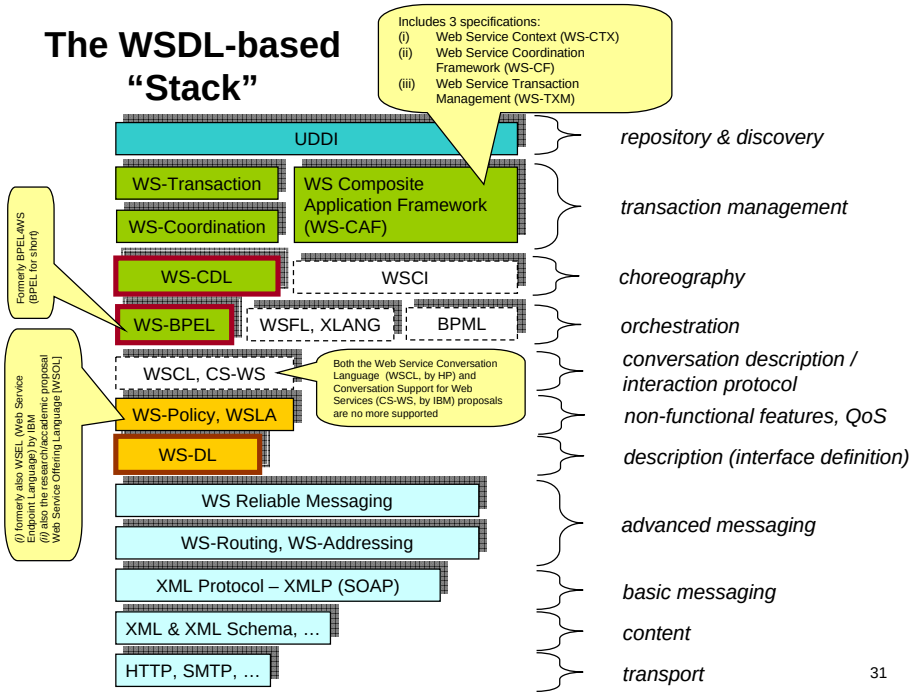
Putting All Together



The "Stacks" of Service Technologies



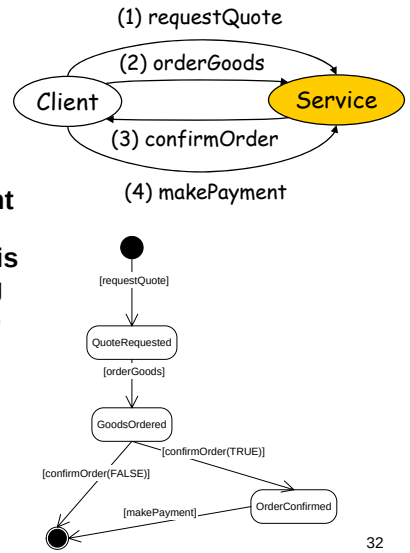
The WSDL-based "Stack"



Conversations

A service is not simply a set of independent operations

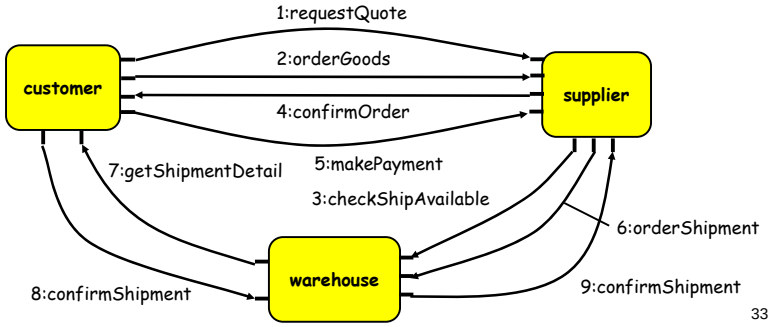
- **Using a service typically involves performing sequences of operations in a particular order (conversations)**
- **During a conversation, the client typically chooses the next operation to invoke (on the basis of previous results, etc.) among the ones that the service allows at that point**



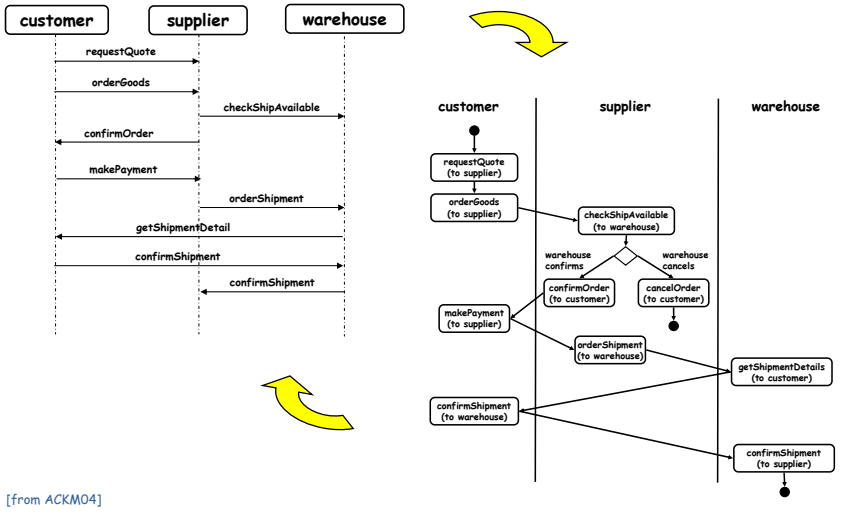
Choreography: Coordination of Conversations of N Services

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



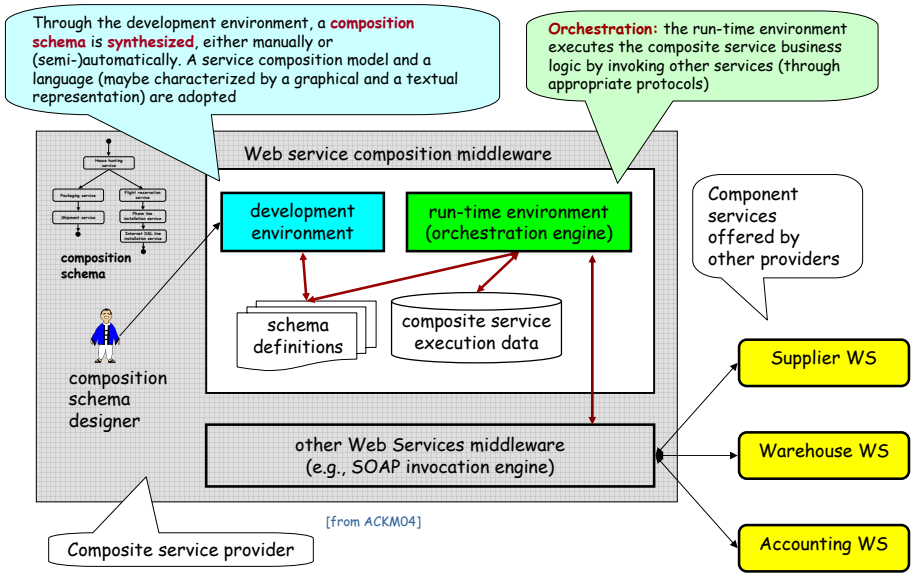
[from ACKM04]

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 ACKM04]

Synthesis and Orchestration

(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 Systems)**

Composition Schema

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)

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 choreographies it can participate in)**

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

Allows specification of composition schemas of Web Services

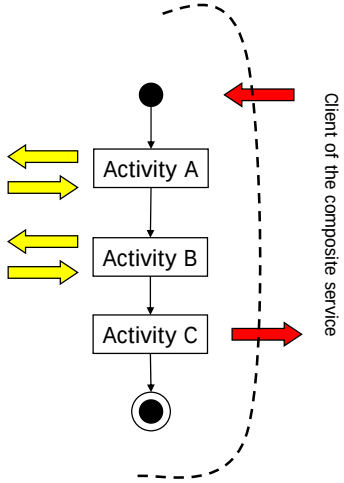
- **Business processes as coordinated interactions of Web Services**
- **Business processes as Web Services**

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

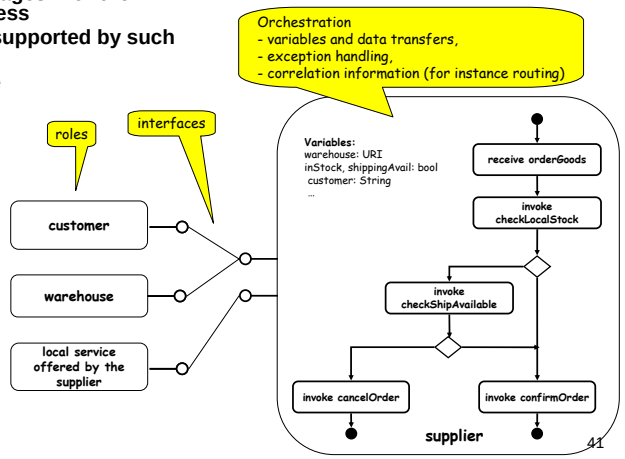
An XML document specifying

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)

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 same transactional context)**

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)

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)**

Choreography

(As Reported in Literature: Classical Ballet Style)

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)

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

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**
- **A Channel type MAY restrict its usage, by specifying the number of times a Channel can be used**

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**

Introducing Automatic Composition

Automatic Composition Synthesis (1)

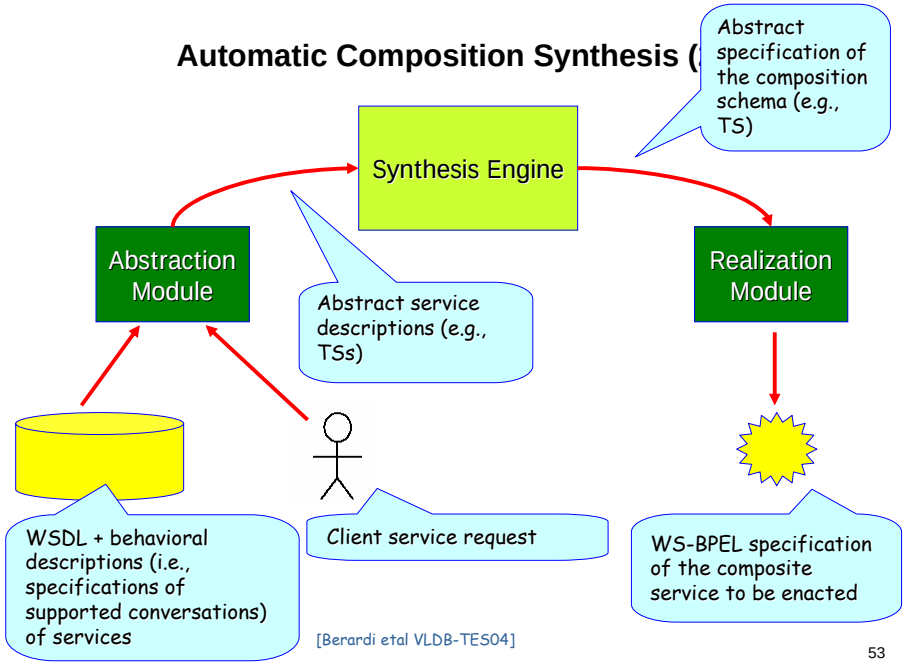
Given:

- **a set (S_1, …, S_n) of component services**
- **a client service request T**

Automatically build:

- **a composition schema CS that fulfills T by suitably orchestrating (S_1, …, S_n)**

Automatic Composition Synthesis (



[Berardi etal VLDB-TES04]

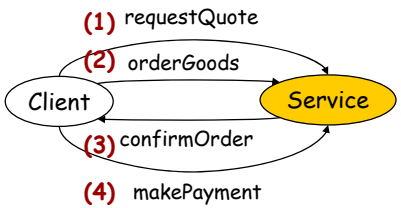
Abstracting over Technologies

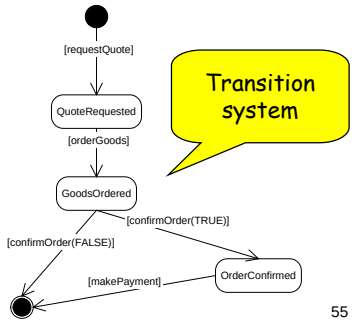
Modeling Services as Transition Systems

Services

- 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

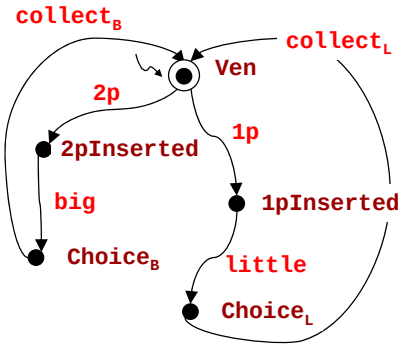




Transition Systems

A transition system (TS) is a tuple T = < A, S, S^0, δ, F > where:

- **A is the set of actions**
- **S is the set of states**
- **$S^0 \subseteq S$ is the set of initial states**
- **$\delta \subseteq S \times A \times S$ is the transition relation**
- **$F \subseteq S$ is the set of final states**



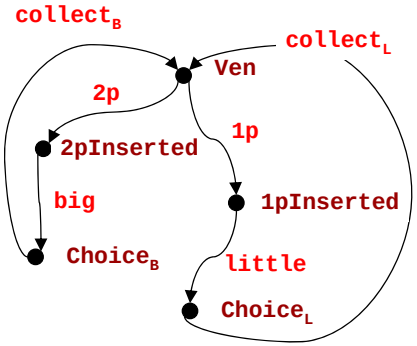
Process Algebras and TSs

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

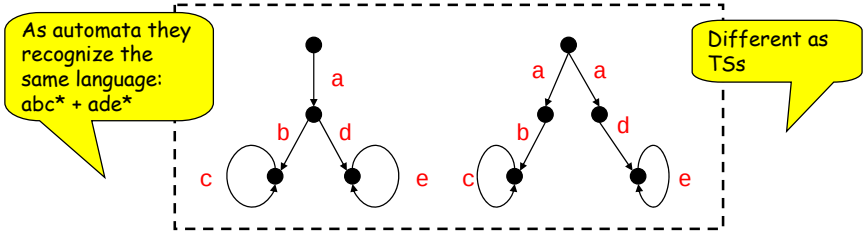
Automata vs. Transition Systems

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 …

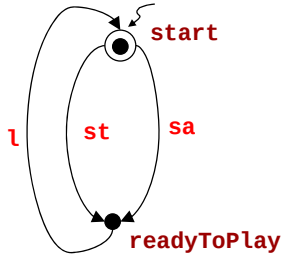
… thus obtaining the alphabet of actions

An Example

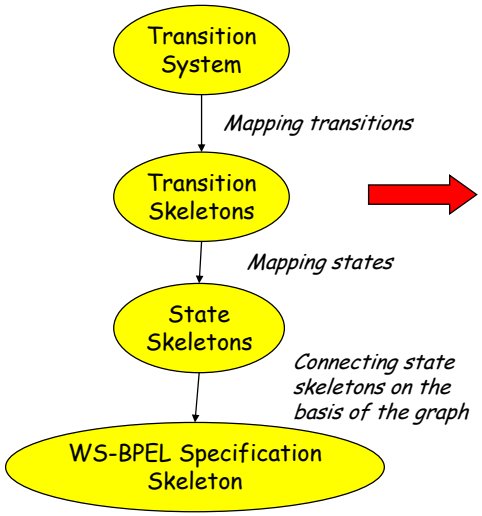
The MP3ServiceInterface defines 3 actions:

- search_by_title / st
- search_by_author / sa
- listen / l

Formally A = {st,sa,l}



From a TS to WS-BPEL (1)




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

   <variables>
    …
   </variables>

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

</process>
```


From a TS to WS-BPEL (2)

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 the graph in depth, 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.
(3) works for acyclic TS. See later for cycle management.

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>
</pick>
```

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**

Mapping Cyclic TSs

(Intuition)

Identify all the cycles

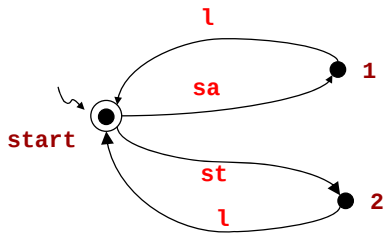
Enclose the involved state skeletons inside a `<while>` block controlled by a condition `(!exit)` (`exit` is a variable defined ad hoc)

- **`exit` is set to TRUE by any transition that "goes out" of the cycle**
- **The overall `<while>` block is connected to other state skeletons by appropriate `<link>`s**

Special cases:

- **A state S with self-transitions can be represented as a `<pick>` block enclosed in a `<while>` block controlled by a condition `(V_s)` (the variable `V_S` is set to FALSE by other non self-transitions)**
- **Cycles starting from the initial state should not be considered, as they can be represented as the start of a new instance**

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"
                          variable="output"/>
                 </sequence>
      </onMessage>
      … …
    </pick>
```

An Exa

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>
```

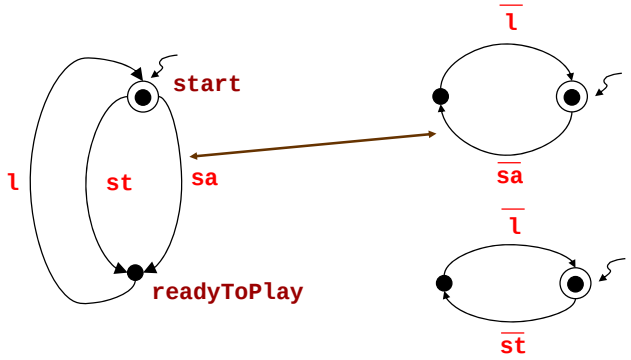

An Example (4)

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


TSs and Choreography

(only an intuition :-))

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



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

[Berardi etal WSCC04] - D. Berardi, R. Hull, M. Gruninger, S. McIlraith: Towards a First-Order Ontology for Semantic Web Services. Proc. W3C International Workshop on Constraints and Capabilities for Web Services (WS-CC), 2004, http://www.w3.org/2004/06/ws-cc-cfp.html

[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

[Berardi etal VLDB-TES04] - D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: Post-proc. of VLDB-TES'04

[Stirling Banff '96] - C. Stirling: Modal and Temporal Logics for Processes. Banff Higher Order Workshop, LNCS 1043, 1996. Available at: http://homepages.inf.ed.ac.uk/cps/banff.ps

[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/ws-bpel/, 2004

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/

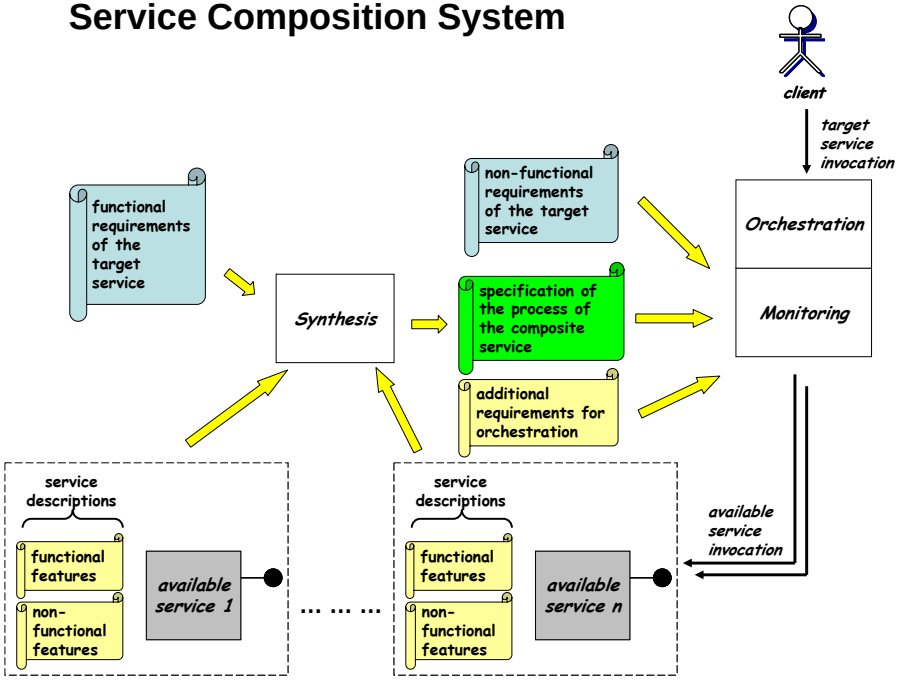
[WS-C] – Web Services Coordination (WS-C), http://www-106.ibm.com/developerworks/library/ws-coor/

[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/

State of the Art on Service Composition (approfondimento opzionale)

Service Composition System



Service composition

How to model client request ?

How to model available services ?

How to model the composite service ?

How to orchestrate the composite service ?

Composition Synthesis:

Input:

- client request
- set of available services

Output:

- specification of composite service

2. Orchestration:

Input:

- specification of composite service

Output:

- coordination of available services according to the composition schema
- data flow and control flow monitoring

Service description

Services export a view of their behavior

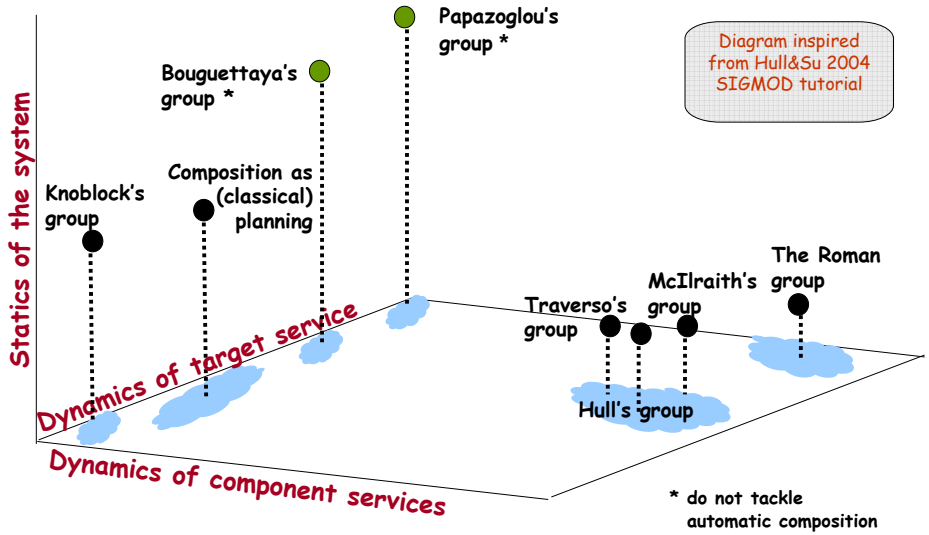
- I/O interface
  - Data Access
    focus on data
    for information gathering
  - Atomic Actions
    focus on actions
    world altering services
- Complex Behavioral Description
  (typically represented using finite states, e.g., TSs)

information oriented services

services as atomic actions

services as processes

The whole picture



Key dimensions in service composition (1)

Statics of the composition system (i.e., static semantics):

- e.g, ontologies of services (for sharing semantics of data/information), inputs and outputs, etc.

Dynamics of component services (i.e., dynamic semantics, process):

- e.g., behavioral features, complex forms of dataflow, transactional attitudes, adaptability to varying circumstances

Key dimensions in service composition (2)

Dynamics of the target service (i.e., dynamic semantics, process)

The target service exposed as:

- single step
- (set of) sequencial steps
- (set of) conditional steps
- while/loops, running batch
- while/loops, running under an external control

Key dimensions in service composition: the 4th dimension

Degree of (in)completeness in the specification of:

- Static Aspects (of the composition system)
- Dynamic Aspects (of component services)
- Target service specification

For simplicity not shown in the following slides

Note: Orthogonal to previous dimensions

What is addressed from the technical point of view?

Automatic composition techniques?

- Which formal tools?
- Sound and complete techniques?
- Techniques/Problem investigated from computational point of view?

Analyzed works

Papazoglou's group
Bouguettaya's group
(not automatic composition)

Knoblock's group (information oriented services)

Composition as Planning (services as atomic actions)

Traverso's group
McIlraith's group
Hull's group
The Roman group
(services as processes)

as called by Rick Hull in his SIGMOD 2004 tutorial

Papazoglou's group

J. Yang and M.P. Papazoglou: Service Components for Managing the Life-cycle of Service Compositions, Information Systems 29 (2004), no. 2, 97 – 125

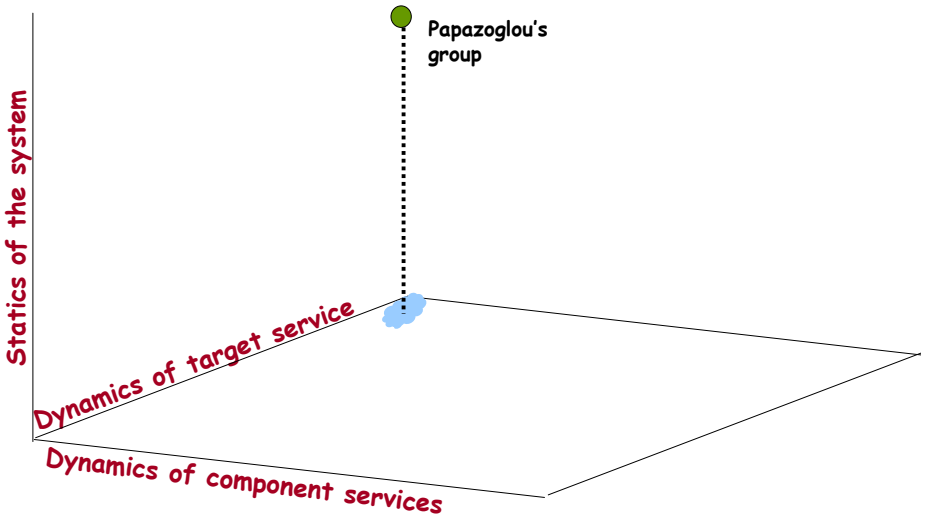
available services: I/O interfaces

- service component: simple or complex pre-existing service wrapped into a web component
- they are stored in a service component class library
- operations offered through a uniform interface

composite service: complex behavioral description

- set of service components (from service component class library) "glued" together by composition logics
  - composition logics defines execution order (either sequential or concurrent) of service components within composition, dependencies among input and output parameters, etc.
- support for manual composition: designer specifies composite service using the Service Scheduling Language and the Service Composition Execution Language

Papazoglou's group



Bouguettaya's group

B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid: Composing Web services on the Semantic Web, Very Large Data Base Journal 12 (2003), no. 4, 333–351

available services: atomic actions

- semantically described in terms of their I/O interfaces and non-functional properties such as their purpose, their category and their quality
- Available services stored into an ontology on the basis of their non-functional properties

Bouguettaya's group

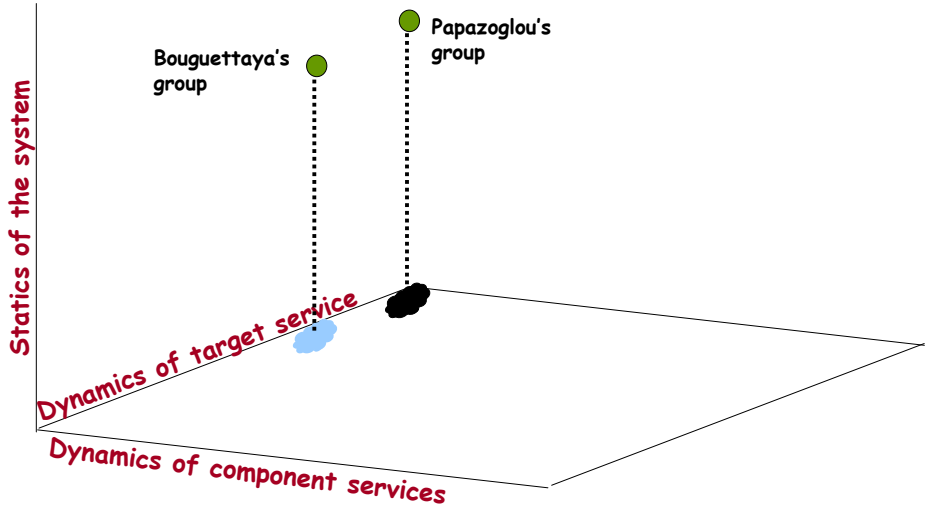
client request:

- **expressed in the Composite Service Specification Language (CSSL): it specifies the sequence of desired operations that the composite service should perform and control flow between operations**

service composition problem:

- Input: *(i)* **I/O descr. of available services**
  (ii) **client request expr. in CSSL**
- Output: **composite service as sequence of operations (semi-automatically) obtained from the client specification by identifying, for each operation, the operation(s) of available services that matches it, on the basis of their I/O interface and non-functional features**

Bouguettaya's group



Knoblock's group

M. Michalowski, J.L. Ambite, S. Thakkar, R. Tuchinda, C.A. Knoblock, and S. Minton: Retrieving and semantically integrating heterogeneous data from the web. IEEE Intelligent Systems, 19 (2004), no. 3, pp.72 – 79

available service: data query

- **basic idea: informative services as views over data sources**
- **each service described in terms of I/O parameters (of course, the latter being provided by the data source), binding patterns and additional constraints on the source**

client request:

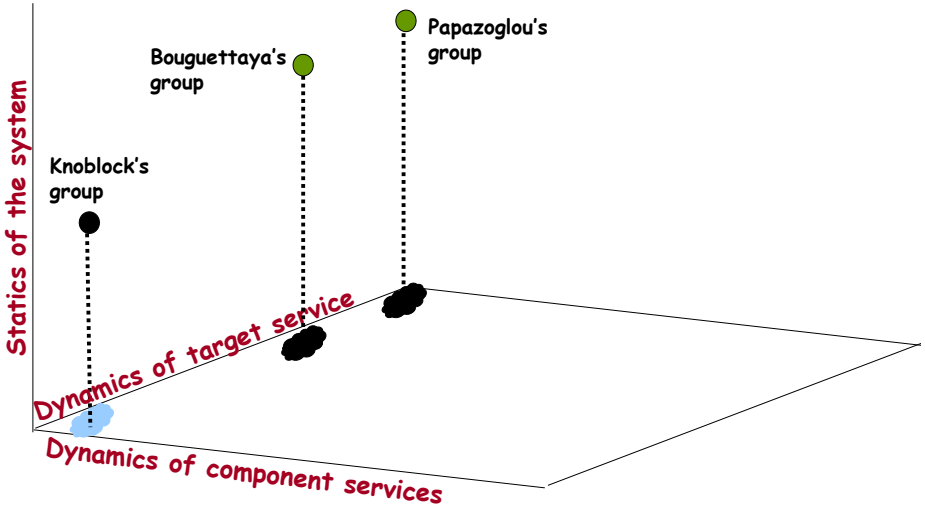
- **data query, expressed in terms of inputs provided by the client and requested outputs**

Knoblock's group

service composition problem:

- Input: *(i)* **available services modeled as data-sources, and** *(ii)* **client request as user query**
- Output**: (automatically obtained) composite service as integration plan for a *generalized* user query, so that all the user queries that differ only for intensional input values can be answered by the same (composite) service. Integration plan as a sequence of source queries, taking binding pattern into account**

Knoblock's group



Composition as Planning

available services: atomic actions

client request: client (propositional) goal

service composition problem: planning problem

- Input: *(i)* **client goal (also encodes initial condition)**
  (ii) **available services as atomic actions**
- Output: **composite service as a (possibly conditional) plan, i.e., sequence of actions that transform the initial state into a state satisfying the goal.**
  - ***Sirin, Parsia, Wu, Hendler & Nau** [Sirin etal ICWS03]*
  - **ICAPS 2003 Planning for Web Services workshop** ***[P4WS03]***
  - **ICAPS 2004 Planning for Web and Grid Services workshop** ***[P4WGS04]***

NOTE: the client has not influence over the control flow of the composite service

Example (1)

Component Services

- S_1: True → {S_1:bookFlight} FlightBooked ∧ MayBookLimo
  MayBookLimo → {S_1:bookLimo} LimoBooked
- S_2: True → {S_2:bookHotel} HotelBooked
  HotelBooked → {S_2:bookShuttle} ShuttleBooked
- S_3: True → {S_3:bookEvent} EventBooked

Ontology:

- TravelSettledUp ≡ FlightBooked ∧ HotelBooked ∧ EventBooked
- CommutingSettled ≡ ShuttleBooked ∨ LimoBooked ∨ TaxiAvailablilityChecked
- ...

Client Service Request:

- **Find a composition of the actions (i.e., a sequence, a program using such actions as basic instructions) such that a given property is fulfilled**

Example (2)

Component Services

- S_1: True → {S_1:bookFlight} FlightBooked ∧ MayBookLimo
  MayBookLimo → {S_1:bookLimo} LimoBooked
- S_2: True → {S_2:bookHotel} HotelBooked
  HotelBooked → {S_2:bookShuttle} ShuttleBooked
- S_3: True → {S_3:bookEvent} EventBooked

Ontology:

- TravelSettledUp ≡ FlightBooked ∧ HotelBooked ∧ EventBooked
- CommutingSettled ≡ ShuttleBooked ∨ LimoBooked ∨ TaxiAvailablilityChecked
- ...

Client Service Request:

- **Starting from: ¬FlightBooked ∧ ¬ HotelBooked ∧ ¬EventBooked ∧ ¬CommutingSettled**
- **Achieve: TravelSettledUp ∧ CommutingSettled**

Example (3)

Component Services
- S_1: True → {S_1:bookFlight} FlightBooked ∧ MayBookLimo
  MayBookLimo → {S_1:bookLimo} LimoBooked
- S_2: True → {S_2:bookHotel} HotelBooked
  HotelBooked → {S_2:bookShuttle} ShuttleBooked
- S_3: True → {S_3:bookEvent} EventBooked

Ontology:
- TravelSettledUp ≡ FlightBooked ∧ HotelBooked ∧ EventBooked
- CommutingSettled ≡ ShuttleBooked ∨ LimoBooked ∨ TaxiAvailablilityChecked
- ...

Client Service Request:

Starting from:
¬FlightBooked ∧ ¬ HotelBooked ∧ ¬EventBooked ∧ ¬CommutingSettled

achieve:
TravelSettedUp ∧ CommutingSettled

Compositions:
- S_1:bookFlight; S_1:bookLimo; S_2:bookHotel; S_3:bookEvent
- S_3:bookEvent; S_2:bookHotel; S_1:bookFlight; S_2:bookShuttle

Another Example (1)

Component Services:
- S_1: Registered → {S_1:bookFlight} FlightBooked
  ¬Registered → {S_1:register} Registered
- S_2: True → {S_2:bookHotel} HotelBooked
  HotelBooked → {S_2:bookShuttle} ShuttleBooked
- S_3: True → {S_3:bookEvent} EventBooked

Ontology:
- TravelSettedUp ≡ FlightBooked ∧ HotelBooked ∧ EventBooked

Client Service Request:

Starting from:
¬FlightBooked ∧ ¬ HotelBooked ∧ ¬EventBooked

Achieve:
TravelSettedUp

Another Example (2)

Client Service Request:
- Starting from: ¬FlightBooked ∧ ¬ HotelBooked ∧ ¬EventBooked
- Achieve: TravelSettedUp

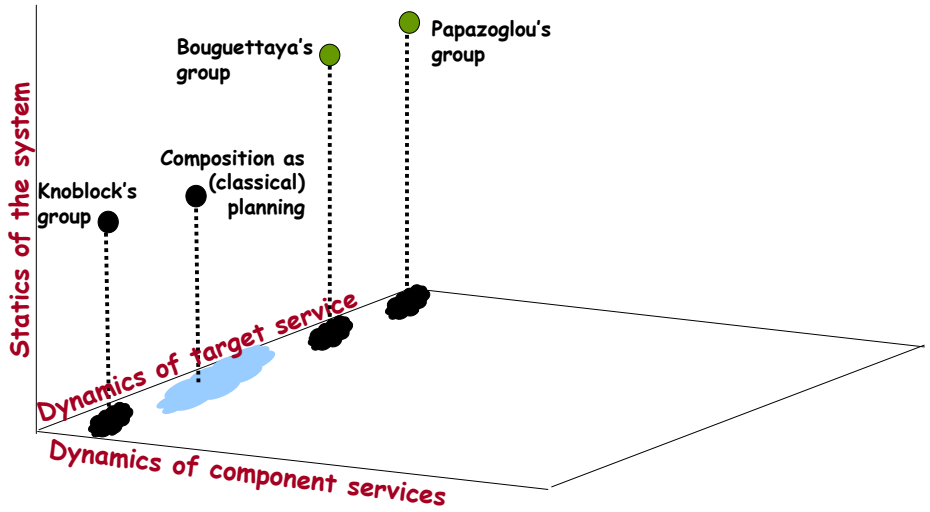
What about Registered?

The client does not know whether he/she/it is registered or not.

The composition must resolve this at runtime:

```
if (¬Registered){
  S1:register;
}
S1:bookFlight;
S2:bookHotel;
S3:bookEvent
```


Composition as Planning



Planning is a Rich Area!!!

Sequential Planning (plans are sequences of actions)
Conditional Planning (plans are programs with if's and while's)
Conformant Planning (plans the work in spite of incomplete -non observable- information)
Knowledge Producing Actions/Sensing (distinction between truth and knowledge)
Plan Monitoring
Interleaving Deliberation and Execution
Form of the Goals:

- **Achieve something**
- **Achieve something while keeping something else**
- **Temporal goals**
- **Main goal + exception handling**

References on Planning

Read and exploit planning and reasoning about actions literature!

Books

Chapters on Planning and on Reasoning about Actions in any Artificial Intelligence textbook.
[GNT04] M. Ghallab, D. Nau, P. Traverso. Automated Planning: Theory and Practice. Morgan Kaufmann, 2004.
[Reiter02] R.Reiter: Knowledge in Action. MIT Press, 2002.

Interesting papers

[Levesque AAAI/IAAI96] H. J. Levesque: What Is Planning in the Presence of Sensing? AAAI/IAAI, Vol. 2 1996: 1139-1146
[Bacchus&Kabanza AAAI/IAAI96] F. Bacchus, F. Kabanza: Planning for Temporally Extended Goals. AAAI/IAAI, Vol. 2 1996: 1215-1222
[Giunchiglia&Traverso ECP99] F. Giunchiglia, P. Traverso: Planning as Model Checking. ECP 1999: 1-20
[Calvanese etal KR02] D. Calvanese, G. De Giacomo, M. Y. Vardi: Reasoning about Actions and Planning in LTL Action Theories. KR 2002: 593-602
[De Giacomo&Vardi ECP99] G. De Giacomo, M. Y. Vardi: Automata-Theoretic Approach to Planning for Temporally Extended Goals. ECP 1999: 226-238
[Bylander IJCAI91] Tom Bylander: Complexity Results for Planning. IJCAI 1991: 274-279

See how other service-researchers have used it!

- **Proceedings of P4WGS – ICAPS Workshop 2004**
- **Proceedings of P4WS – ICAPS Workshop 2003**

Traverso's group

available services:

- **non-deterministic transition systems characterized by a set of initial states and by a transition relation that defines how the execution of each action leads from one state to a set of states**
- **among such services, one represents the client**

client request (called global goal):

- **it specifies a main execution to follow, plus some side paths that are typically used to resolve exceptional circumstances e.g.,** Do Φ else Try Ψ

Traverso's group

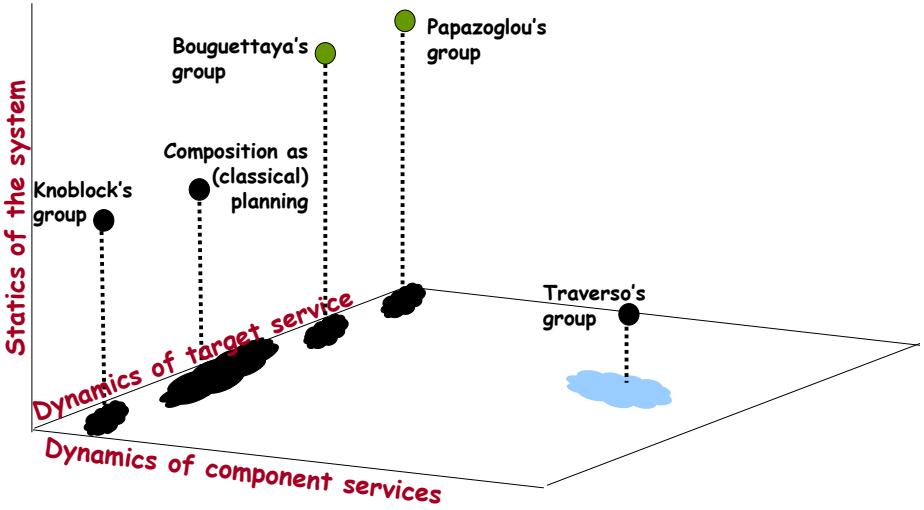
service composition problem: (extended) planning problem

- **Input: *(i)* a set of services, including the one representing the client (behavior), and *(ii)* the global goal,**
- **Output: a plan that specifies how to coordinate the execution of various services in order to realize the global goal.**

NOTE:

- **the composition is not tailored towards satisfying completely the client requested behavior, but concerns with the global behavior of the system in which some client desired executions may happen not to be fulfilled**

Traverso's group



References on Traverso's group

Papers on Planning as Model Checking

[Giunchiglia&Traverso ECP99]F. Giunchiglia, P. Traverso: Planning as Model Checking. ECP 1999: 1-20

[Pistore&Traverso IJCAI01] M. Pistore, P. Traverso: Planning as Model Checking for Extended Goals in Non-deterministic Domains. IJCAI 2001: 479-486

[Bertoli etal IJCAI01] P. Bertoli, A. Cimatti, M. Roveri, P. Traverso: Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking. IJCAI 2001: 473-478

[Dal Lago etal AAAI/IAAI02] U. Dal Lago, M. Pistore, P. Traverso: Planning with a Language for Extended Goals. AAAI/IAAI 2002: 447-454

[Cimatti etal AIJ03] A. Cimatti, M. Pistore, M. Roveri, P. Traverso: Weak, strong, and strong cyclic planning via symbolic model checking. Artif. Intell. 147(1-2): 35-84 (2003)

[Bertoli etal ICAPS03] P. Bertoli, A. Cimatti, M. Pistore, P. Traverso: A Framework for Planning with Extended Goals under Partial Observability. ICAPS 2003: 215-225

Papers on Service Composition

[Pistore&Traverso ISWC04] M. Pistore, P. Traverso: Automated Composition of Semantic Web Services into Executable Processes. ISWC2004.

[Pistore etal P4WGS04] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, P. Traverso: Planning and Monitoring Web Service Composition. P4WGS – ICAPS WS 2004

[Pistore etal AIMSA04] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, P. Traverso: Planning and Monitoring Web Service Composition. AIMSA 2004: 106-115

McIlraith's group

both available and composite service: behavioral description seen as procedures invokable by clients

- **Golog procedure, atomically executed, i.e., seen by its client as an atomic Situation Calculus action, presenting an I/O interface**
- **each service stored in an OWL-S ontology**

McIlraith's group

client request:

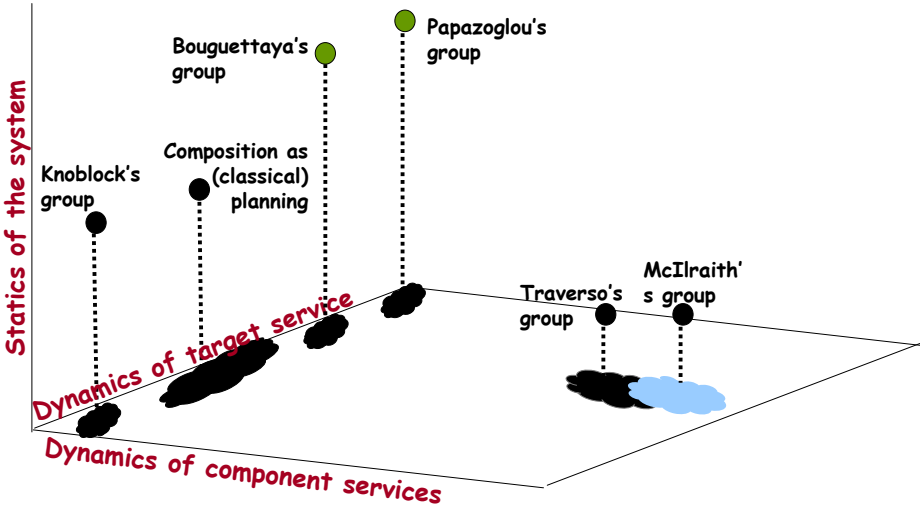
- **skeleton of a Golog procedure expressing also client constraints and preferences**

service composition problem:

- **Input: *(i)* OWL-S ontology of services as atomic actions, and *(ii)* client request**
- **Output: Golog procedure obtained by automatically instantiating the client request with services contained in the ontology, by also taking client preferences and constraints into account**

NOTE: the client has not influence over the control flow of the composite service

McIlraith's group



References on McIlraith's group

Background

[McCarthy IFIP62] J. L. McCarthy: Towards a Mathematical Science of Computation. IFIP Congress 1962: 21-28

[McCarthy&Hayes MI69] J. L. McCarthy and P. C. Hayes: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, 1969

[Reiter 2002] R. Reiter: Knowledge in Action. MIT Press, 2002.

[Levesque etal JLP2000] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, R. B. Scherl: GOLOG: A Logic Programming Language for Dynamic Domains. J. Log. Program. 31(1-3): 59-83 (1997)

[De Giacomo etal AIJ2000] G. De Giacomo, Y. Lespérance, H. J. Levesque: ConGolog, a concurrent programming language based on the situation calculus. Artif. Intell. 121(1-2): 109-169 (2000)

[De Giacomo etal KR02] G. De Giacomo, Y. Lespérance, H. J. Levesque, S. Sardiña: On the Semantics of Deliberation in IndiGolog: From Theory to Implementation. KR 2002: 603-614

[Scherl&Levesque AIJ03] R. B. Scherl, H. J. Levesque: Knowledge, action, and the frame problem. Artif. Intell. 144(1-2): 1-39 (2003)

Papers

[McIlraith etal IEEE01] S. A. McIlraith, T. Cao Son, H. Zeng: Semantic Web Services. IEEE Intelligent Systems 16(2): 46-53 (2001)

[Narayanan&McIlraith WWW02] S. Narayanan, S. A. McIlraith: Simulation, verification and automated composition of web services. WWW 2002:

[McIlraith&Son KR02] S. A. McIlraith, T. Cao Son: Adapting Golog for Composition of Semantic Web Services. KR 2002: 482-496

[Burstein etal ISWC02] M. H. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. V. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, K. P. Sycara: DAML-S: Web Service Description for the Semantic Web. International Semantic Web Conference 2002: 348-363

[Narayanan&McIlraith CN03] Srini Narayanan, Sheila A. McIlraith: Analysis and simulation of Web services. Computer Networks 42(5): 675-693 (2003)

[McIlraith&Martin IEEE03] S. A. McIlraith, D. L. Martin: Bringing Semantics to Web Services. IEEE Intelligent Systems 18(1): 90-93 (2003)

Hull's group

both available and composite service (peer): behavioral description

- **Mealy machine, that exchanges messages with other peers according to a predefined communication topology (channels among peers)**
- **peers equipped with (bounded) queue to store messages received but not yet processed**
- **Conversation: sequence of messages exchanged by peers**
- **At each step, a peer can either *(i)* send a message, or *(ii)* receive a message, or *(iii)* consume a message from the queue, or *(iv)* perform an empty move, by just changing state**

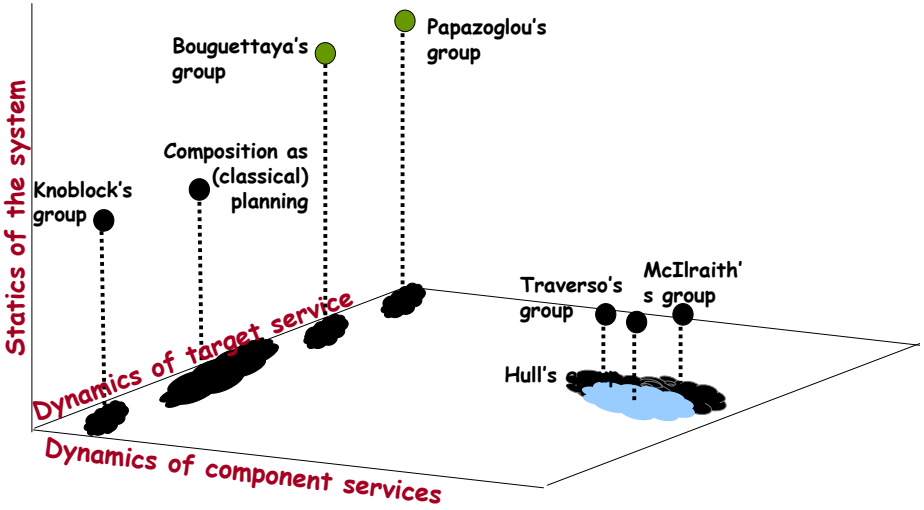
Hull's group

Choreography mapping problem:

- **Input: *(i)* a desired global behavior (i.e., set of desired conversations) as a Linear Temporal Logic formula, and *(ii)* an infrastructure (a set of channels, a set of peer names and a set of messages)**
- **Output: Mealy machines (automatically obtained) for all the peers such that their conversations are compliant with the LTL specification**

NOTE: not yet a "jam session style" choreography

Hull's group



References on Hull's group

[Hull etal PODS03] R. Hull, M. Benedikt, V. Christophides, J. Su: E-services: a look behind the curtain. PODS 2003: 1-14

[Hull etal SIGMOD03] R. Hull, J. Su: Tools for Design of Composite Web Services. SIGMOD Conference 2004: 958-961

[Bultan etal WWW03] T. Bultan, X. Fu, R. Hull, J. Su: Conversation specification: a new approach to design and analysis of e-service composition. WWW 2003: 403-410

The Roman group

available service: behavioral description

- service as an interactive program: at each step it presents the client with a set of actions among which to choose the next one to be executed
- client choice depends on outcome of previously executed actions, but the rationale behind this choice depends entirely on the client
- behavior modeled by a finite state transition system, each transition being labeled by a deterministic (atomic) action, seen as the abstraction of the effective input/output messages and operations offered by the service

The Roman group

client request (target service):

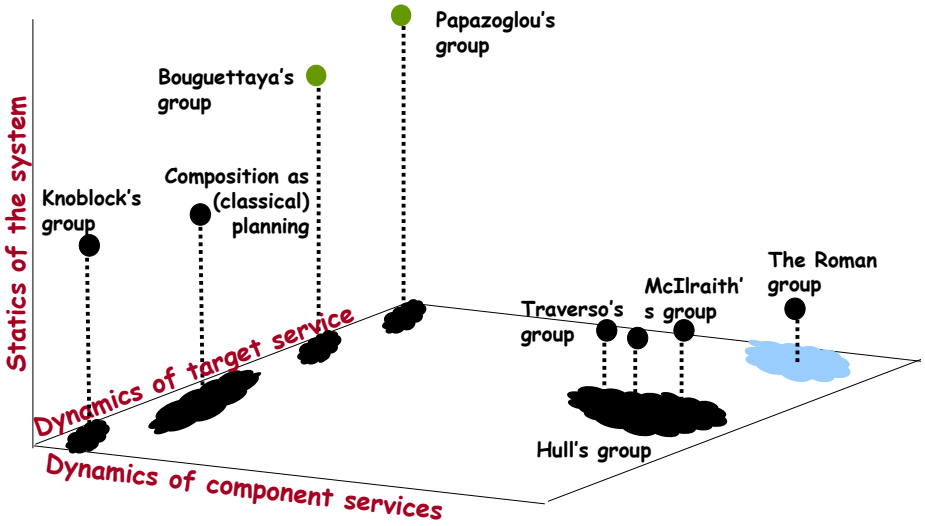
- set of executions organized in a (finite state) *transition system* of the activities he is interested in doing

service composition problem:

- Input: *(i)* finite state transition system of available services, and *(ii)* finite state transition system of target service
- Output: (automatically obtained) composite service that realizes the client request, such that each action of the target service is delegated to at least one available service, in accordance with the behavior of such service.

NOTE: the client "strongly" influence the composite service control flow

The Roman group



References on the Roman group

[Berardi etal ICSOC03] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: Automatic Composition of E-services That Export Their Behavior. ICSOC 2003: 43-58

[Berardi etal ICSOC04] D. Berardi, G. De Giacomo, M. Lenzerini, M. Mecella, D. Calvanese: Synthesis of Underspecified Composite e-Services based on Automated Reasoning. ICSOC 2004

[Berardi etal WES03] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: A Foundational Vision of e-Services. WES 2003: 28-40

[Berardi etal P4WS03] D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella: Composing e-Services by Reasoning about Actions, ICAPS 2003 Workshop on Planning for Web Services (P4WS03).

[Berardi etal DL03] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: e-Service Composition by Description Logics Based Reasoning. Description Logics 2003

[Berardi etal P4WGS04] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: Synthesis of Composite e-Services based on Automated Reasoning. ICAPS 2004 Workshop on Planning and Scheduling for Web and Grid Services (P4WGS04).

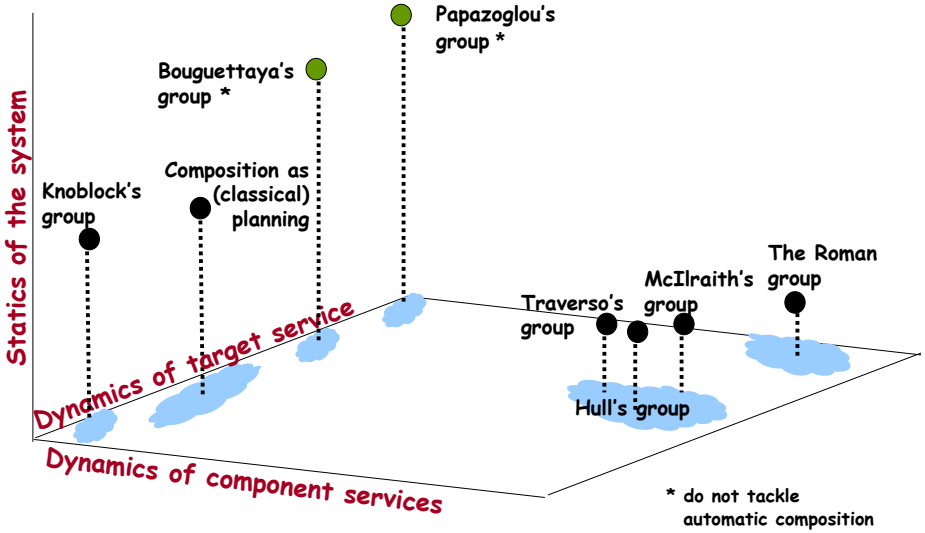
[Berardi etal TES04] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: ESC: A Tool for Automatic Composition of e-Services based on Logics of Programs, VLDB-TES 2004

[Berardi Ph.D] D. Berardi Automatic Service Composition.Models, Techniques and Tools. Ph.D. thesis, Dipartimento di Informatica e Sistemistica – Universita' di Roma "La Sapienza", Rome, Italy, 2005.

[IJCIS 2004] D. Berardi, G. De Giacomo, M. Lenzerini, M. Mecella, D. Calvanese: Automatic Service Composition based on Behavioral Description. To appear in IJCIS 2005

[Gerede etal ICSOC04] C. E. Gerede, R. Hull, O. H. Ibarra, J. Su: Automated Composition of E-services: Lookaheads. ICSOC 2004

The whole picture



Other Relevant Works

Approaches proposing interesting conceptual models for services, not targeted towards composition:

- **Vianu 's group**
- **Benatallah & Casati's group**

Vianu's group

A. Deutsch, L. Sui, and V. Vianu: Specification and Verification of Data-driven Web Services, In Proceedings of the 23nd ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2004), ACM, 2004, pp. 71–82

available service: data query + behavioral descr.

- **service as a data-driven entity characterized by a database and a tree of web pages**
- **At each step, set of input choices presented to client: some generated as queries over the database; specific client data treated as constants. The client chooses one of such inputs, and in response, the service produces as output updates over the service database and/or performs some actions, and makes a transition from a web page to another**

automatic verification of service properties:

- **both over runs (linear setting) and over sets of runs (branching setting)**
- **they characterize the complexity of verifying such properties for various classes of services**

Benatallah & Casati's group

B. Benatallah, F. Casati, and F. Toumani:
Web services conversation modeling: The Cornerstone for E-Business Automation.
IEEE Internet Computing, 8 (2004), no. 1, pp.46 – 54

available service: behavioral description

- **behavior of a service as finite state transition system in terms of message exchanged with the clients (conversations)**
- **transitions labeled by messages, and states labeled with the status of the conversation (e.g., effect of the message exchange leading to it, if clearly defined)**

they study how to automatically generate the skeleton of a BPEL4WS spec. starting from the transition system modeling the service behavior

they also study properties of service behavior in order for two services to correctly interact

(Only) Orchestration

Two main kinds of orchestration *[Hull etal PODS03]* :

- ***(i)* the mediated approach, based on a hub-and-spoke topology, in which one service is given the role of process mediator/delegator, and all the interactions pass through such a service, and**
- ***(ii)* the peer-to-peer approach, in which there is no centralized control**

Mediated Orchestration Engines

***e*-Flow *[Casati & Shan, IS01]* :**

- **Platform for specifying, enacting and monitoring composite service**
- **Composite *E*-Service (CES) is a service process engine offered as (meta-) service that performs coordination of services, with some process adaption/evolution mechanisms**
- **A provider can offer a value added service as coordination of different services: it registers the new service to the CES and let the CES enact its execution**

AZTEC *[Christophides etal TES01]* :

- **Framework for orchestration of session-oriented, long running telecommunication services is studied. It is based on active flowcharts thus coping with asynchronous events that can happen during active telecom sessions**

Mediated Orchestration Engines

WISE *[Lazcano etal CSSE2000]* :

- **Orchestration engine that coordinates the execution of distributed applications (virtual processes), and a set of brokers enables the interaction with already existing systems that are to be used as building blocks.**
- **Process meta-model based on Petri Nets, with the possibility to add Event-Condition-Action (ECA) rules**

MENTOR-lite *[Shegalov etal VLDBJ01]* :

- **Workow management system based on a XML mediator for coordinating services which are distributed among different organizations and deployed on heterogeneous platforms**
- **Process meta-model is based on a specific statechart dialect**

Peer-to-Peer Orchestration Engines

Self-Serv *[Benatallah etal IEEE03]* :

- **Platform for composing services and executing new composed services in a decentralized way, through peer-to-peer interactions**
- **Composite service modeled as an activity diagram**
- **Its enactment carried out through the coordination of different state coordinators (one for each service involved in the specification and one for the composite service itself)**

PARIDE Orchestrator *[Mecella etal VLDB-TES02]* :

- **A composition schema, modeled as a specific Coloured Petri Net, is orchestrated by a set of organizations, which moves it (as a "token") along the execution**
- **Separation between the responsibility of the orchestration and the providing of services (suitable in specific scenarios)**
- **Services can be substituted with other compatibles**

References

[Casati & Shan, IS01] - F. Casati and M.C. Shan, Dynamic and Adaptive Composition of e-Services, Information Systems 6 (2001), no. 3, 143 – 163.

[Christophides etal TES01] - V. Christophides, R. Hull, G. Karvounarakis, A. Kumar, G. Tong, and M. Xiong. Beyond Discrete e-Services: Composing Session-oriented Services in Telecommunications. In Proc. of VLDB-TES, 2001.

[Lazcano etal CSSE2000] - A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler, The WISE approach to Electronic Commerce, International Journal of Computer Systems Science & Engineering 15 (2000), no. 5

[Shegalov etal VLDBJ01] - G. Shegalov, M. Gillmann, and G. Weikum, XML-enabled Workflow Management for e- Services across Heterogeneous Platforms, Very Large Data Base Journal 10 (2001), no. 1, 91–103.

[Benatallah etal IEEE03] - B. Benatallah, Q. Z. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. IEEE Internet Computing, 7(1):40–48, 2003

[Mecella etal VLDB-TES02] – M. Mecella, F. Parisi Presicce, B. Pernici: Modeling e-Service Orchestration Through Petri Nets. Proc. VLDB-TES 2002, LNCS 2444. An extended version as M. Mecella, B. Pernici: Building Flexible and Cooperative Applications Based on eServices, Technical Report 21-02, DIS Univ. Roma "La Sapienza", 2002

ebXML

(approfondimento opzionale)

ebXML

ebXML is more a standardized "conceptual framework", a "reference model", than a real stack of standard technologies

- ***Stable version in 2001/2002***
  - **Technical Architecture Specification (v1.04)**
  - **Business Process Specification Schema (v1.01)**
  - **Registry Information Model (v2.0)**
  - **Registry Services Specification (v2.0)**
  - **Requirements Specification (v1.06)**
  - **Collaboration-Protocol Profile and Agreement Specification (v2.0)**
  - **Message Service Specification (v2.0)**

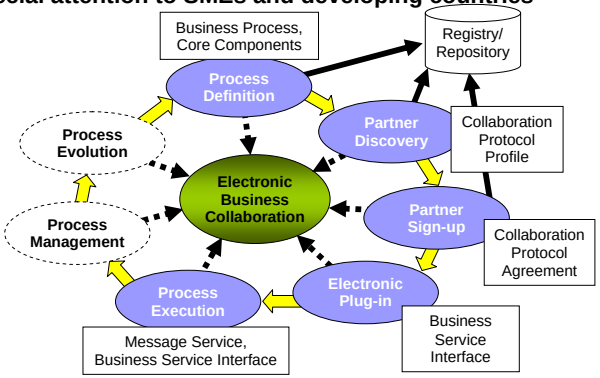
Currently in revision

- **Indeed, many Technical Committees (TCs) are working in synergy with the promoters of the W3C/WSDL-based "stack"**
  - **E.g., UDDI v2 has been developed in the context of ebXML/OASIS, currently WS-BPEL and WS-CAF are being evolved/developed in the context of specific TCs, etc.**

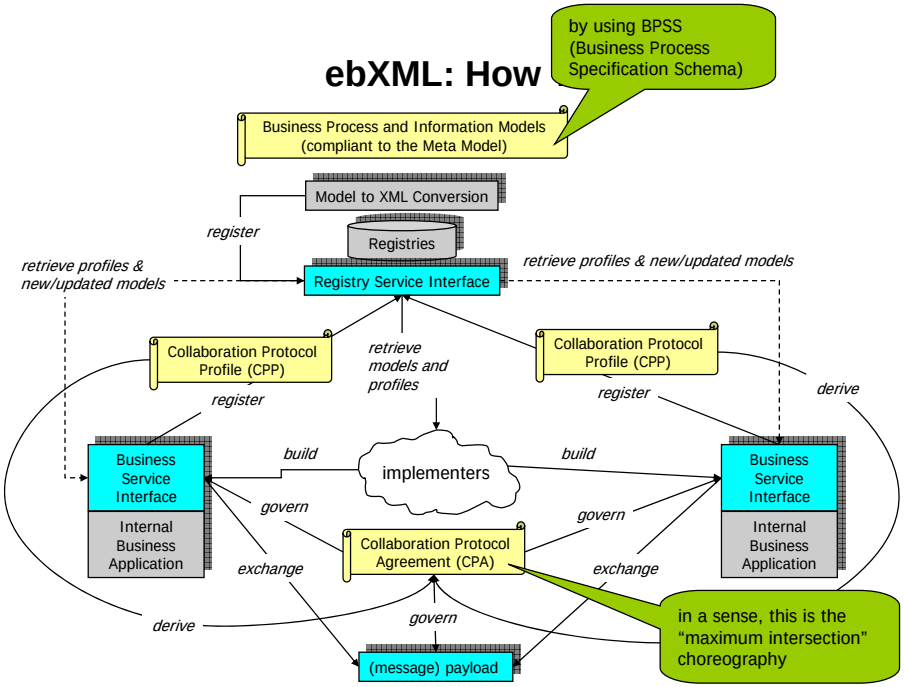
ebXML: Aims

To define an open & public infrastructure, based on XML, for distributed electronic commerce

- **Special attention to SMEs and developing countries**



ebXML: How



ebXML: BPSS, CPP e CPA (1)

BPSS is used for modeling a business process, thus obtaining a BPS (Business Process Specification)

- **Partners, roles, *collaborations* and document exchanges (*business transactions*)**
- **Collaboration: set of *activities*; an activity is a business transaction or again a collaboration**
- **Business transaction: a partner is the requester, the other is the responder, in a *business document flow***

CPP: expresses the capabilities of a partner in partecipating in a BPS

ebXML: BPSS, CPP e CPA (2)

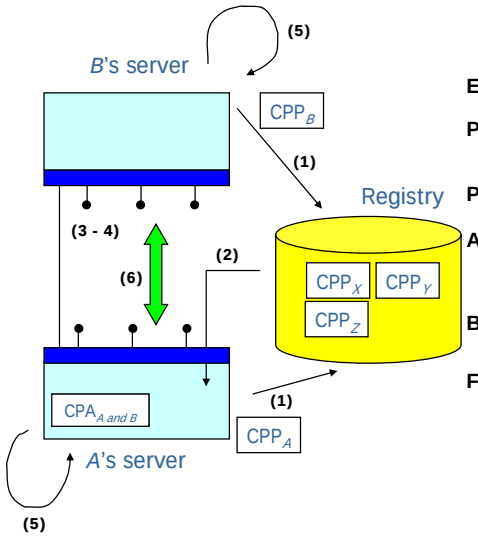
$\mathcal{A}$ wants to make electronic business with $\mathcal{B}$; $\mathcal{A}$ is the acquirer and $\mathcal{B}$ the vendor; the process underlying the business is already defined in a BPS

$\mathcal{A}$ discovers the $\mathcal{B}$'s CPP in a registry

A CPA is created, as the intersection of $\mathcal{A}$'s CPP and $\mathcal{B}$'s CPP

On the basis of the CPA, the $\mathcal{A}$'s and $\mathcal{B}$'s business service interfaces are configured in order to support the business transactions

ebXML: BPSS, CPP e CPA (3)



Each partner has registered its own CPP in the registry
Partner $\mathcal{A}$ discovers $\mathcal{B}$ in the registry and download $CPP_{\mathcal{B}}$ on its system
Partner $\mathcal{A}$ creates $CPA_{\mathcal{A}\text{ and }\mathcal{B}}$ and sends it to $\mathcal{B}$
After a negotiation (both manual or automatic), both $\mathcal{A}$ and $\mathcal{B}$ register identical copies of the agreed upon $CPA_{\mathcal{A}\text{ and }\mathcal{B}}$ in their systems
Both $\mathcal{A}$ and $\mathcal{B}$ configure their systems for runtime on the basis of $CPA_{\mathcal{A}\text{ and }\mathcal{B}}$
Finally $\mathcal{A}$ and $\mathcal{B}$ engage their e-Commerce process

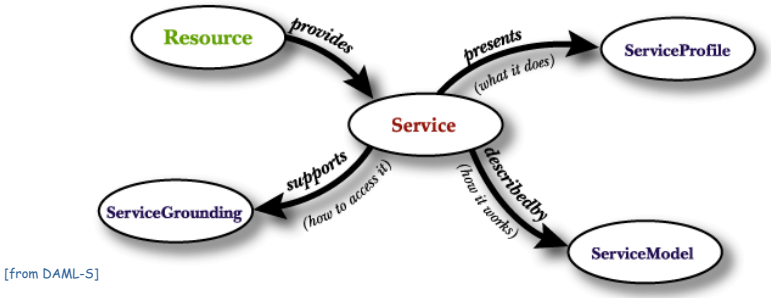
Semantic Web Services (approfondimento opzionale)

OWL-S (formely DAML-S)

An emerging standard to 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]

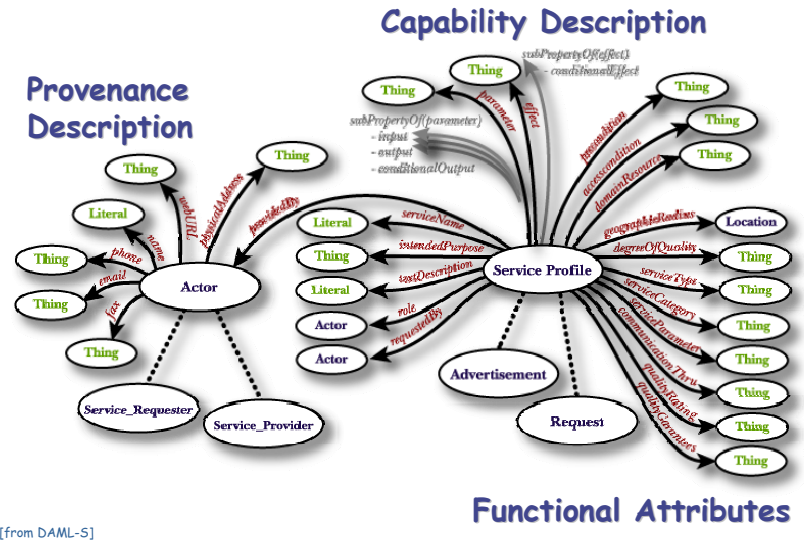
OWL-S Service Profile
(What it does)

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

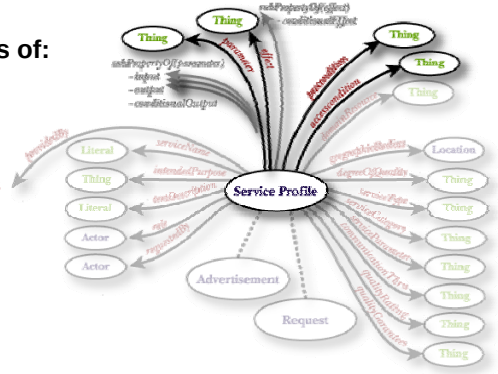


[from DAML-S]

Capability Description

Specification of what the service provides

- **High-level functional representation in terms of:**
  - **preconditions**
  - **inputs**
  - **(conditional) outputs**
  - **(conditional) effects**



IOPE

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

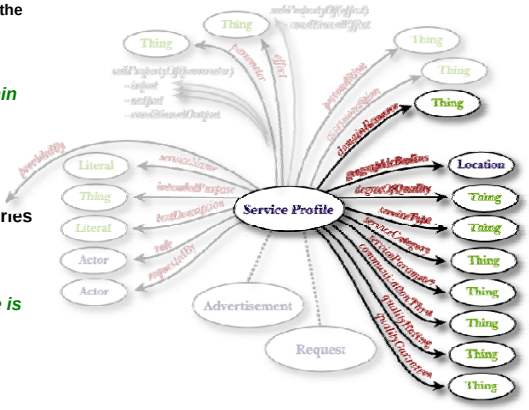
Preconditions
- Set of conditions that should hold prior to service invocation

(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

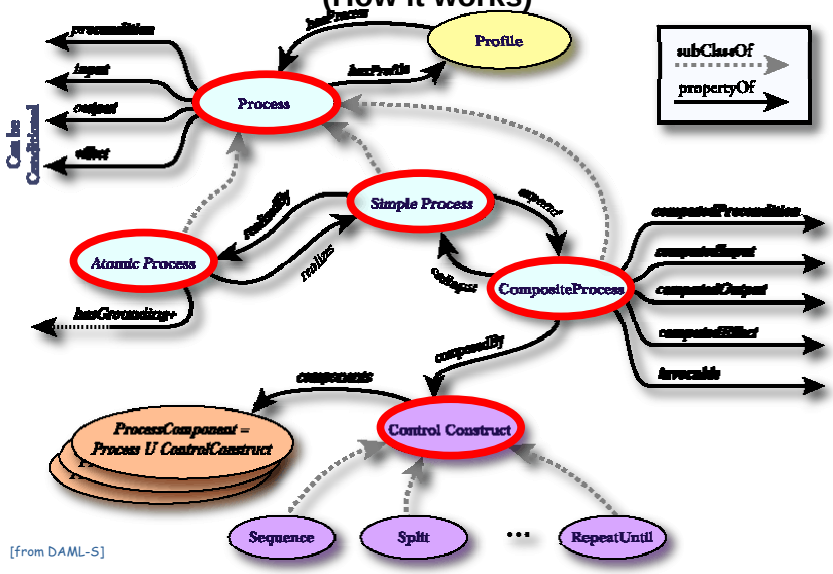
Functional Attributes

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

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

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

Recent proposals aim at improving and detailing process modeling and dynamic semantics …

- **WSMO (Web Service Modeling Ontology)**
- **SWSL (Semantic Web Service Language)**

… work in progress !!