University of Pisa

MSc in Computer Engineering

Process-driven Information Systems

LECTURE 22

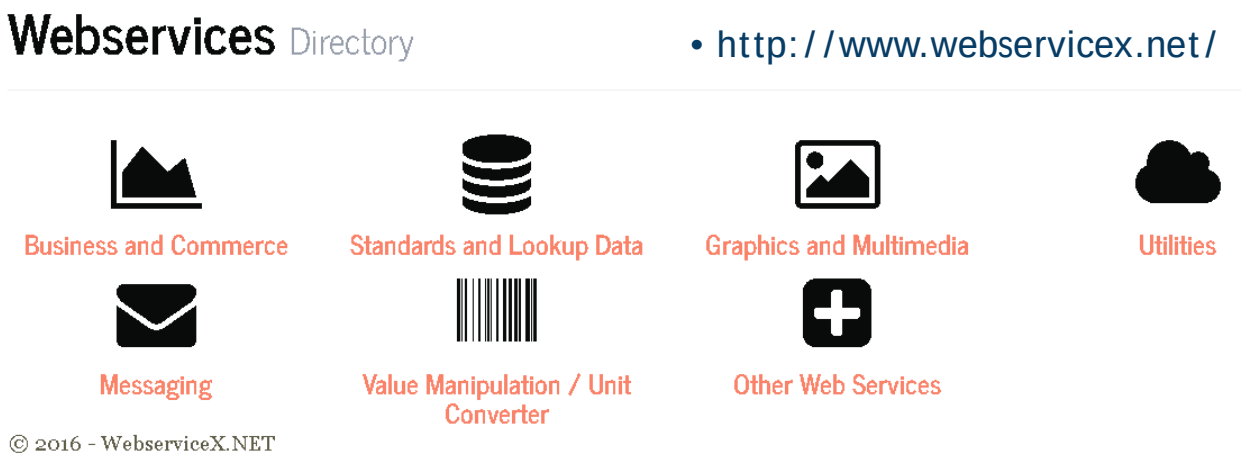
http://www.iet.unipi.it/m.cimino/wdis/

Mario G.C.A. Cimino

Department of Information Engineering

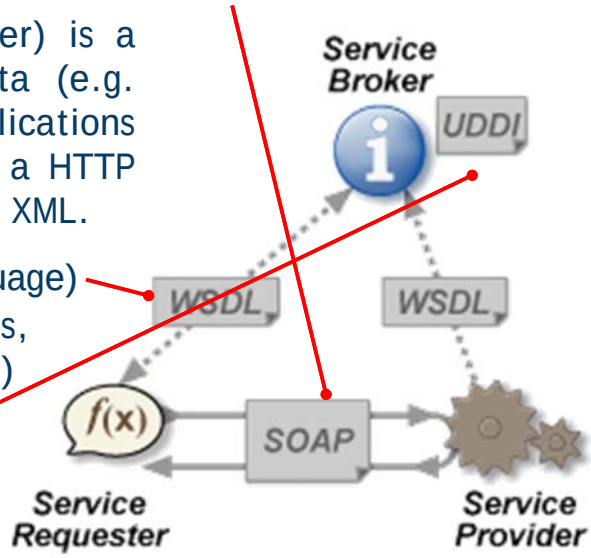
BP Management: Introduction to Web Service

- A **Web Service** (WS) is a software system designed to support interoperable **machine-to-machine** interaction over a network.
- Many companies today offer software on the Web as a service: *Google, Yahoo, Amazon, eBay*.
- Governments collect a lot of data opening up access to data via Web Services (WS). WS technology represents an important way for businesses to communicate with each other and with clients as well.

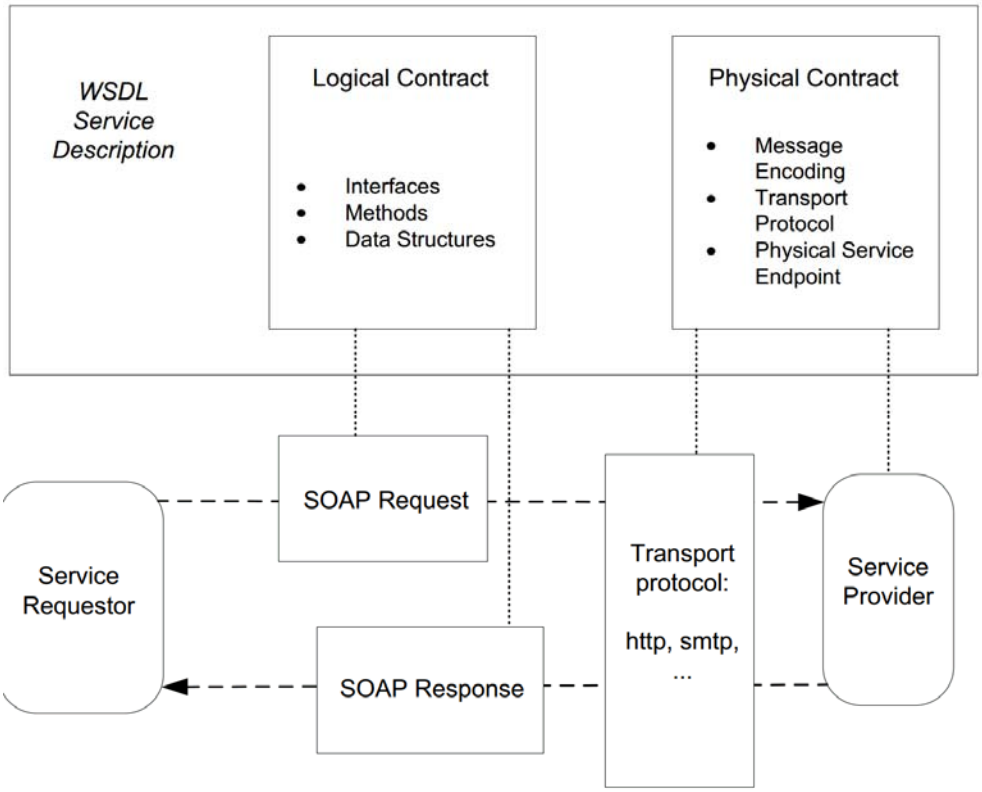


- http://www.webservicex.net/

- E.g. a purchase-and-ordering WS communicates to an inventory WS that specific items need to be reordered. Many WS can be chained to implement complicated workflows.
- W3C-XML protocols are used to interact with a WS.
- **SOAP** (Simple Object Access Protocol) over **HTTP** is used to exchange XML messages between the Requester and the Provider of the service.
- REST (REpresentational State Transfer) is a lightweight method to transfer data (e.g. between simple devices/client applications and a WS provider). IT is based on a HTTP request-response message, in JSON or XML.
- WSDL (Web Services Description Language) specifies the WS interface (operations, input/outputs, types, endpoints, ecc.)
- UDDI (Universal Description, Discovery and Integration) provides information to search and access the WS via a Registry



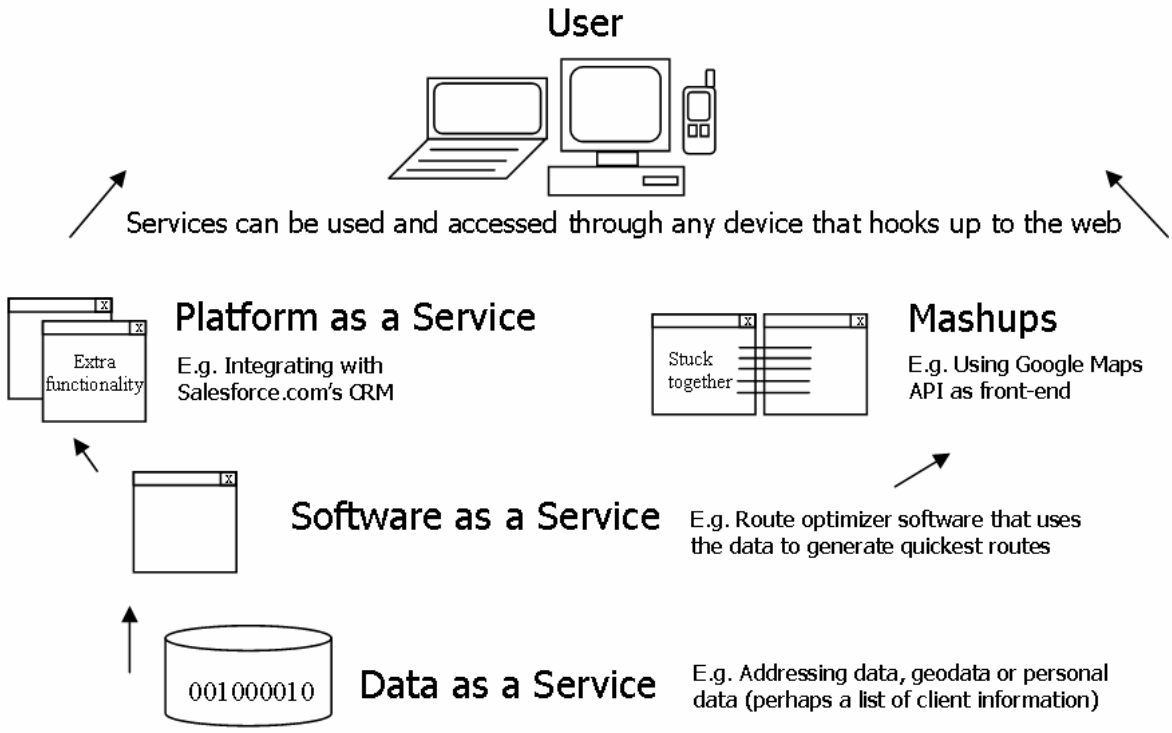
- The WS provider is responsible for preparing a WSDL file of the service
- The WS requestor need access to the WSDL file
- The requestor sends a SOAP message to the service endpoint using information in the WSDL
- The provider invokes the software and returns a SOAP message



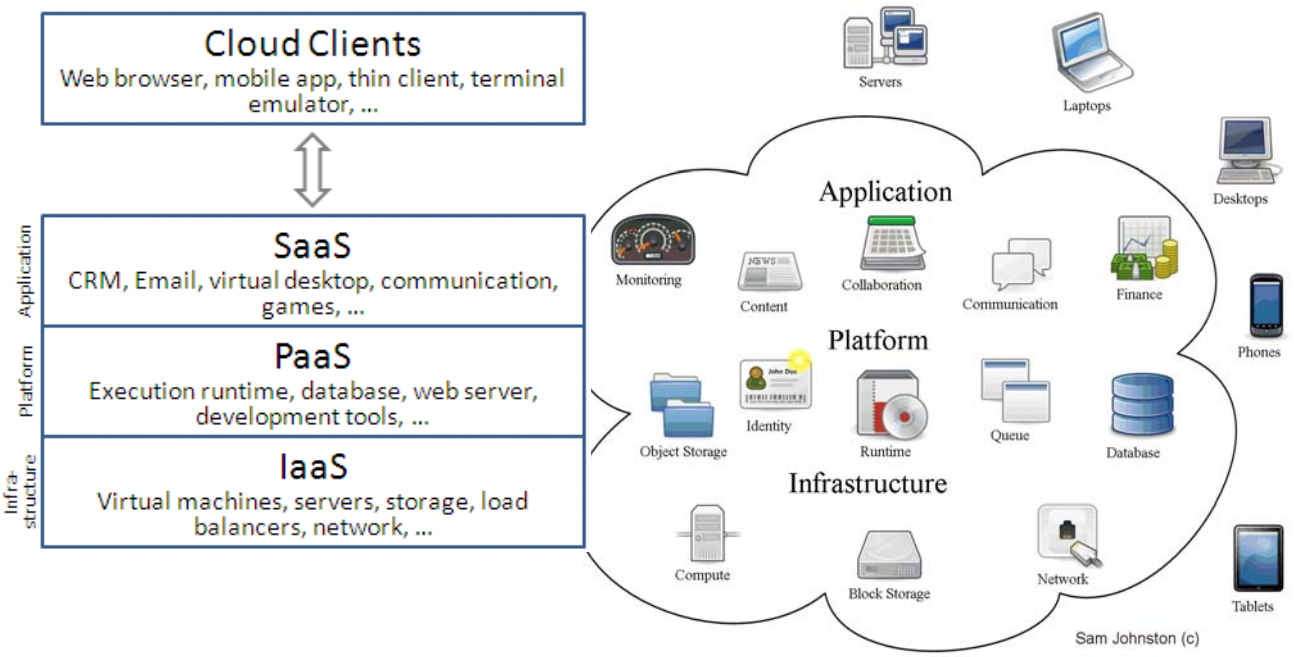
Role of WSDL in Web service invocation

Service-Oriented Architecture

A completely service-oriented model

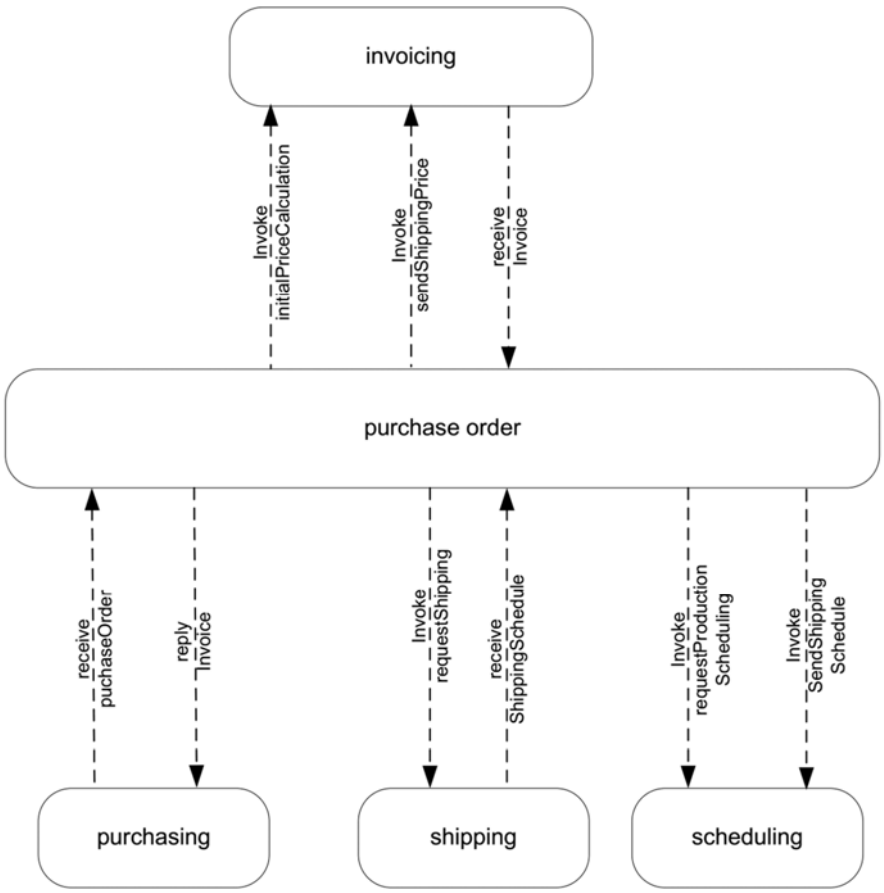


- The Service-Oriented Architecture (SOA) is one of the enablers of Cloud Computing: Internet-based computing providing on demand resources



⇒ +Agility, -Cost, -Maintenance, +Performance, +Productivity, +Reliability

- WS composition: describes how a set of services are related to each other. It is an implementation of system workflows.
- **WS-BPEL**: Business Process Execution Language for WS, is a related XML standard
- Example of a high-level purchase order composition showing the communication with each WS.

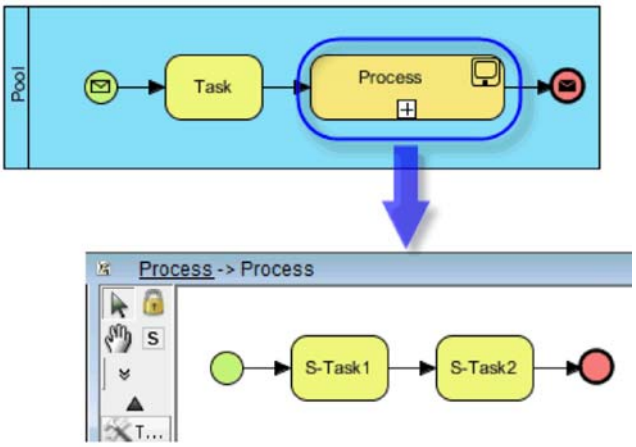


- BPEL can be generated from BPMN, under some limitation
- BPMN is a graph-oriented language in which control and action noted can be connected almost arbitrarily
- BPEL is a mainly block-structured language, an extension of imperative programming languages
- Since BPEL offers loops, if-then-else, XML data types, it is Turing complete

Sketch of a BPEL-to-BPMN mapping

	BPEL	BPMN
Basic Activities	*invoke*	sending/receiving task, message event
	receive (createInstance='no')	receiving task, message event
	reply	sending task, message event
	validate	—
	assign	assignment
	wait	timer intermediate event
	exit	termination end event
	throw, rethrow	error end event
	compensate, compensateScope	compensation events
Structured Activities	*sequence*	sequence flow
	if-elseif-else	excl. data-based gateway, default flow
	while, repeatUntil	standard loop activity
	foreach	multiple-instance loop activity
	pick (createInstance='no')	event-based gateway, message/timer event
	flow, control links	parallel gateway, inclusive gateway, complex gateway
	scope	embedded subprocess
	fault handlers	exception flow
	event handlers	—
	termination handler	—
	compensation handler	compensation activity, compensation event, compensation association
Generic	*variables*	data object
	correlation mechanism	property
	process instantiation	message events, excl. event-based gateway
	communication abstractions	participant, web service, role

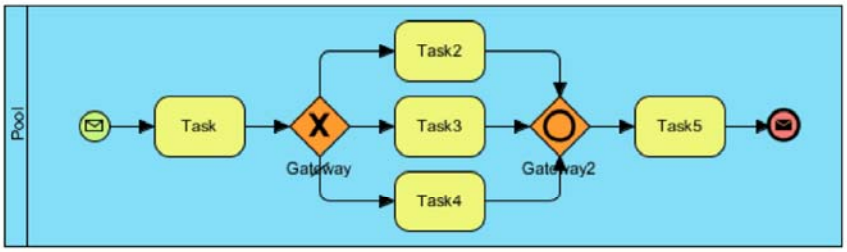
- Examples of BPMN-to-BPEL transformations made by the Visual Paradigm modeler suite
- In the BPEL, the flow modeled in cub-process is merged to the ordinary flow: the activities *STask1* and *STask2* are modeled in the sub-process diagram, following *Task*




```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="...
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType"..
    <empty name="Task"/>
    <empty name="STask1"/>
    <empty name="STask2"/>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```

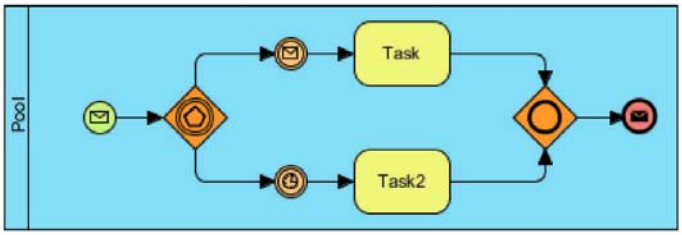

- Example 2: the BPMN XOR gateway is translated into a BPEL switch

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="http://schemas.xmlsoap.org/ws
business-process/" xmlns:Pool="http://mypool" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <empty name="Task"/>
    <switch name="Gateway">
      <case condition="true()">
        <sequence>
          <empty name="Task2"/>
        </sequence>
      </case>
      <case condition="false()">
        <sequence>
          <empty name="Task3"/>
        </sequence>
      </case>
      <otherwise>
        <sequence>
          <empty name="Task4"/>
        </sequence>
      </otherwise>
    </switch>
    <empty name="Task5"/>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```



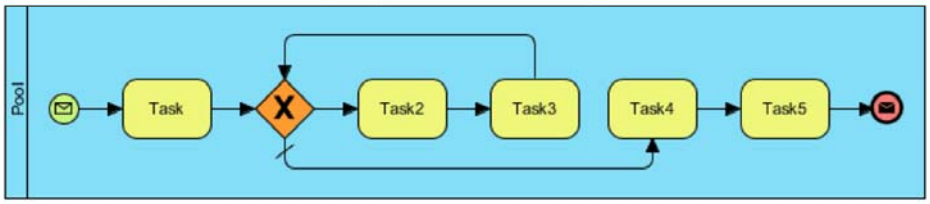
- Example 3: the BPMN event-driven XOR gateway becomes a BPEL pick, which provides two branches, each one with a condition. The branch that has its condition satisfied first is executed.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="...
business-process/" xmlns:Pool="http://b" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="...
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType" partnerRole="requester"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <pick createInstance="no">
      <onMessage operation="continue" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable">
        <sequence>
          <empty name="Task"/>
        </sequence>
      </onMessage>
      <onAlarm for="'PT5S'">
        <sequence>
          <empty name="Task2"/>
        </sequence>
      </onAlarm>
    </pick>
    <invoke inputVariable="Variable" operation="performCallback" partnerLink="Pool" portType="Pool:RequesterPortType"/>
  </sequence>
</process>
```

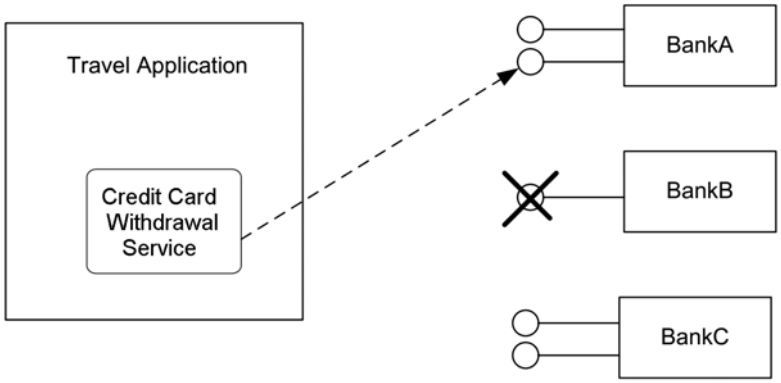


- Example 4: in BPMN having a sequence flow back into a gateway, forming a loop, means in BPEL to repeat the flow while a condition is satisfied.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="http://schemas.xmlsoap.org/ws/...
business-process/" xmlns:Pool="http://mypool" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <empty name="Task"/>
    <while condition="false()">
      <sequence>
        <empty name="Task2"/>
        <empty name="Task3"/>
      </sequence>
    </while>
    <empty name="Task4"/>
    <empty name="Task5"/>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```

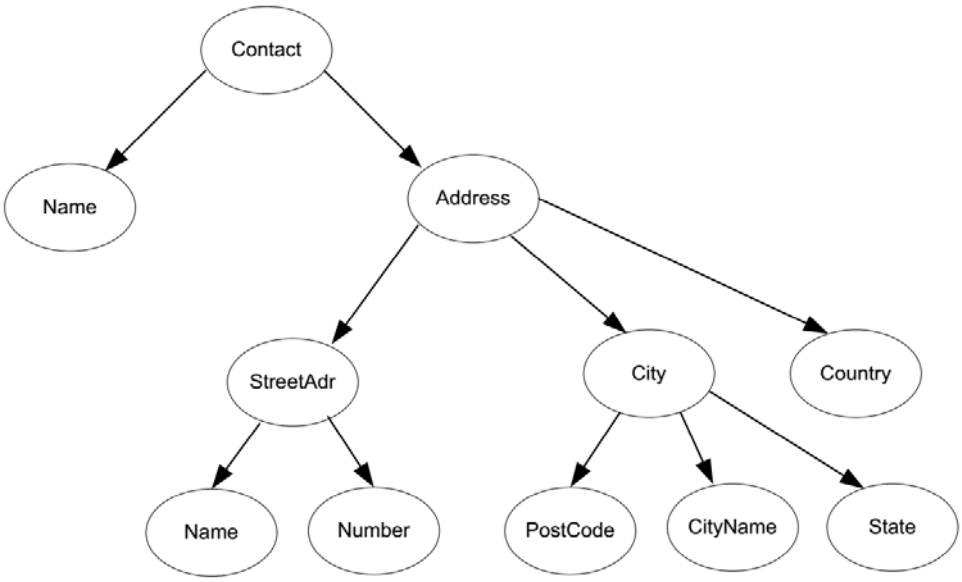


- Let us consider an example of a **travel application**, allowing customers to select trips, make reservations, and confirm reservations by providing credit card information.
- To allow this composition of services, the travel application invokes a credit card withdrawal service provided by a bank. In static binding the two services are bound at **development time.** However, this is not effective in environments with dynamic service landscape
- In dynamic binding, service implementations can be discovered at runtime: the application (or a supporting middleware) asks the service registry for a list of suitable services, selects and binds to one of them.
- Service matchmaking: the process of selecting a set of services that fit a service request. It is made by the service broker.
- It depends on rich semantic annotation of services



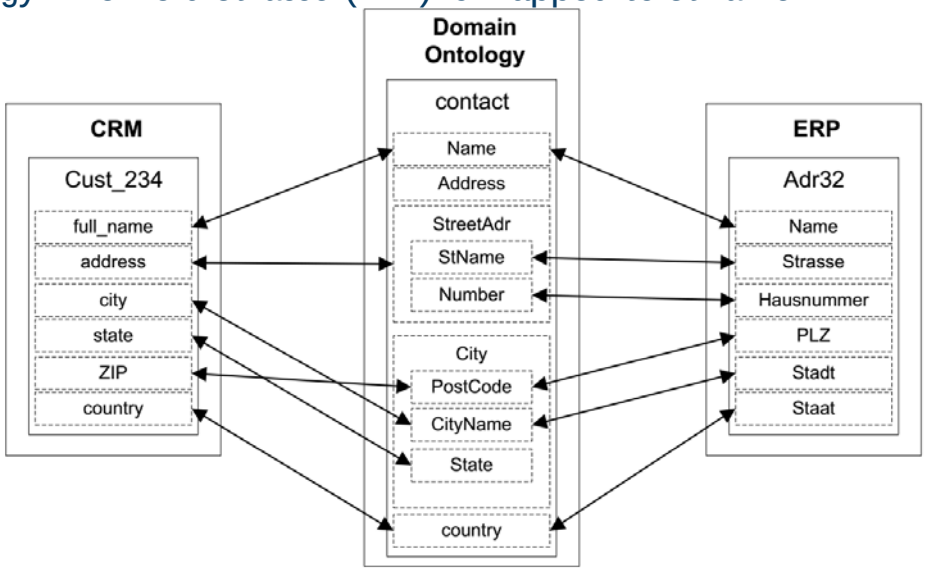
- Services can be composed in a correct way if they operate on the same domain concepts. The simplest case occurs when consequent services operate on the same domain concept: e.g. a service returning customer data can be combined with a service taking customer data as an input.
- Since web services are usually developed independently of each other, the WSDL data types in most cases do not match.
- Typically syntactic differences are solved by system architects and software developers, using data mapping techniques. With compositions of many services this approach introduces a considerable overhead due to heterogeneous data types.
- Even with similar parameter names like *europrice* and *price*, the user of the service cannot be sure that the price is really the euros currency. For example there might be semantic differences: one service returns 120 and the other 118. Since the concept of price is not agreed upon the providers, the price 120 includes value-added tax (VAT), while the price 118 does not.
- To solve this problem, data should be semantically annotated by using **Semantic Web** standard, to be automatically compared and integrated. (RDF, resource Description Framework, OWL: ontology Web langauge)

- A domain ontology is associated with a set of stakeholders, who need to agree on the domain ontology. In an ontology, concepts are represented by ellipses, and the relationships are represented by directed arcs.
- Example of simple domain ontology for contacts.



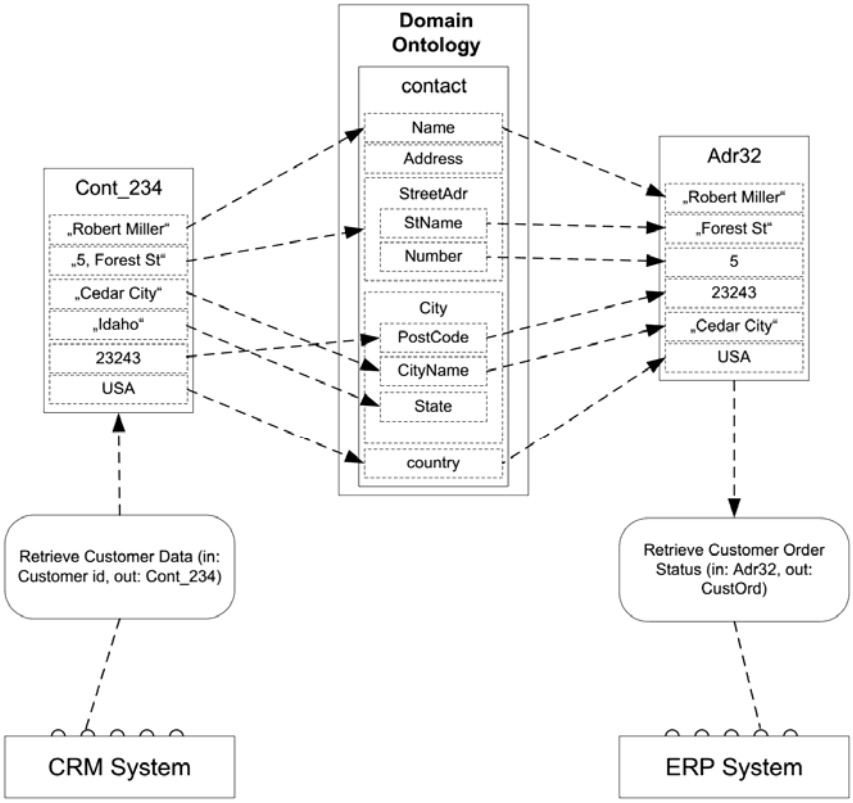
Domain ontology for contacts

- The Contact domain ontology can be used to integrate a Customer Relationship Management (CRM) and an Enterprise Resource Planning (ERP) with different data structures.
- E.g. the *full_name* field of the CRM is mapped to the *Name* concept in the domain ontology. The field *Strasse* (ERP) is mapped to *StName.*
- If each data field is mapped to the domain ontology, the mapping of the data between two system can be achieved automatically at runtime.



Domain ontology facilitates data mapping

- A service of the CRM returning a parameter of data type Cust_234 can be fed into a service taking a parameter of data type Adr32 if the appropriate ontology is provided.



Domain ontology used for automatic data mapping of CRM customer data to ERP customer data