The Semantic Web
(Semantic Web Services)

Răzvan Popescu
University of Pisa

The Web Today

- The Web is syntactic; humans interpret the information.
- Link to UNIPI:
  - Humans: Click here to open the home-page of the University of Pisa
  - Computers: !@ #$ %^&*()_+
- How can we get the computers do the hard work?

Issues on the Web

There are (at least) two main problems with:

1. The information retrieval, and with
2. The Web as a computing paradigm

Information Retrieval

- Keyword-based matching systems
- Results (often) depend on parameters that have nothing to do with semantics (e.g., page rank, sponsored links, aso.)
- Sometimes the link of interest is not in the first 3 pages
- What about complex queries or finding multimedia files?
- Query construction ~ asm programming
- Complexity of info retrieval: O(exp(t))?

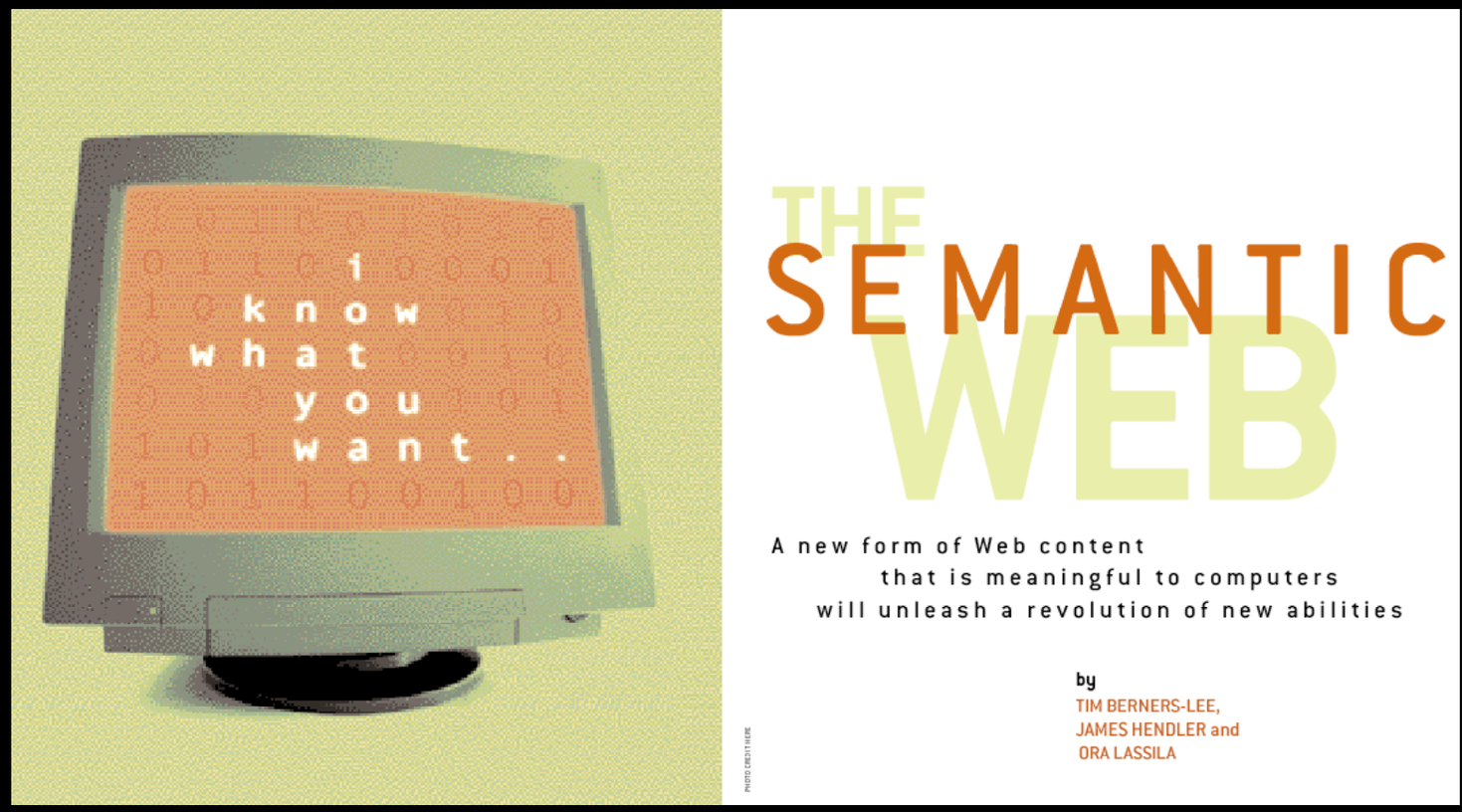
Web as a Computing Paradigm

- Use the output of a Web resource as the input of another Web resource
- Service-oriented Computing
  - Service Discovery (UDDI, keyword/category-based)
  - Composition (WS-BPEL, manual)
  - Adaptation (manual)

Problem and Solution

- Common Problem: Computers cannot extract semantics (i.e., meaning) from syntactic pages
- (Possible) solution/Patch: Add semantic annotations to Web resources

Scientific American, May 2001



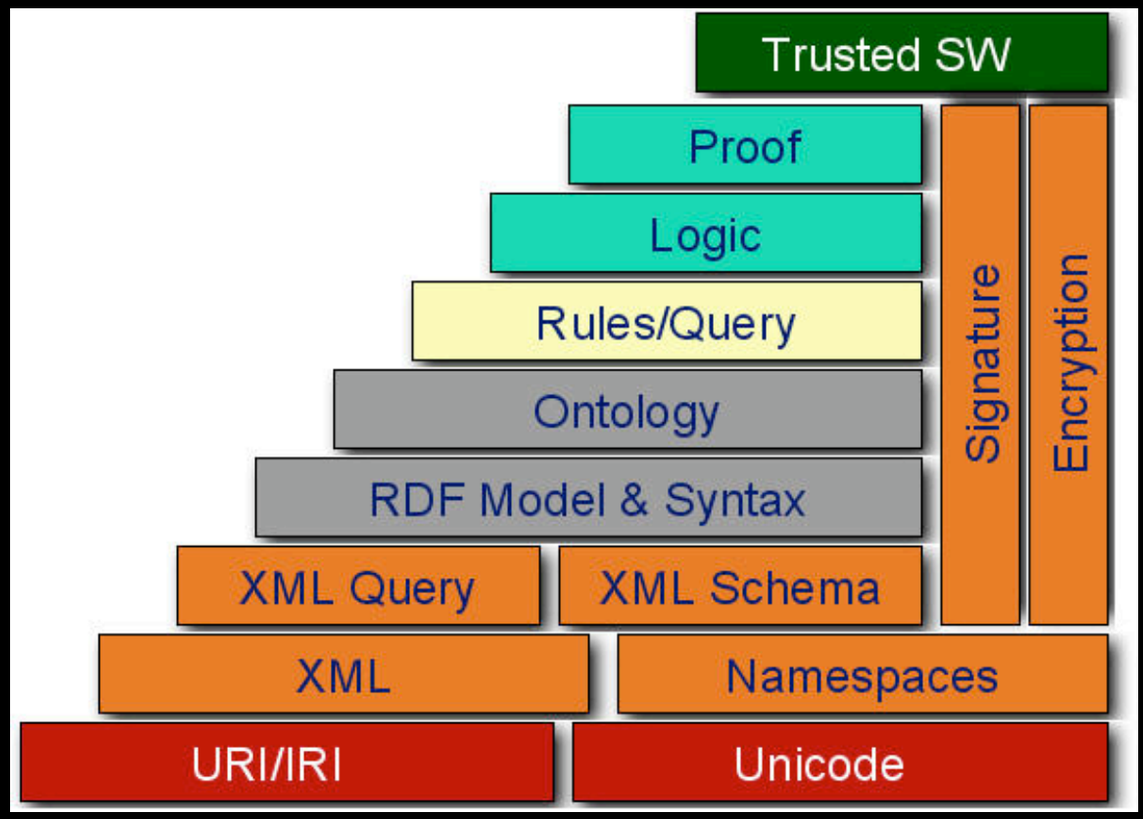
Semantic annotation shall enable machine processable data and the automation of processing the data on the Web.

Semantic Web - Vision

"... a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the machine-readable information space gave an accurate representation of the state of people's thoughts, interactions, and work patterns, then machine analysis could become a very powerful management tool, seeing patterns in our work and facilitating our working together through the typical problems which beset the management of large organizations." (Tim Berners-Lee)

W3C Semantic Stack

- XML: syntax for structured documents, no semantic constraints on the meaning of documents
- XML Schema: restricts the structure of XML documents
- RDF: data model for referring to objects ("resources") and how they are related
- RDF Schema: vocabulary for describing properties and classes of RDF resources, with a semantics for generalisation-hierarchies of such properties and classes
- OWL: adds more vocabulary for describing properties and classes -- relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes



XML

Humans:

```
<personalInformation>
    <name>Mario Rossi</name>
    <address>
        <street>Largo B. Pontecorvo 3</street>
        <town>Pisa</town>
        <country>Italy</country>
    </address>
    ...
</personalInformation>
```

Computers:

```
<!@#>
  <$%^>...</$%^>
  <&*(>
    ...
  </&*(>
  ...
</!@#>
```

XML: the means to describe and apply a tree-based structure to information.

Basic syntax:

<name attribute="value"*>content?</name>

What About Matching?

```
<personalInformation>
   <name>Mario Rossi</name>
   <address>
      <street>Largo B. Pontecorvo 3</street>
      <town>Pisa</town>
      <country>Italy</country>
   </address>
   ...
   ...
</personalInformation>
```

```
<buyer>
   <userName>Mario Rossi</userName>
   <deliveryAddress>
      <street>Largo B. Pontecorvo 3</street>
      <town>Pisa</town>
      <country>Italy</country>
   </deliveryAddress>
   ...
</buyer>
```

XML Strengths

- It is a simultaneously human- and machine-readable format
- It has support for Unicode, allowing almost any information in any human language to be communicated
- The ability to represent the most general computer science data structures: records, lists and trees
- The self-documenting format that describes structure and field names as well as specific values
- The strict syntax and parsing requirements allow the necessary parsing algorithms to remain simple, efficient, and consistent
- It is platform-independent, thus relatively immune to changes in technology

XML Weaknesses

- Its syntax is fairly verbose and partially redundant
- Parsers should be designed to recurse arbitrarily nested data structures and must perform additional checks to detect improperly formatted or differently ordered syntax or data
- The basic parsing requirements do not support a very wide array of data types so interpretation sometimes involves additional work in order to process the desired data from a document.
- Modelling overlapping (non-hierarchical) data structures requires extra effort.

XML Extensions

- XPath makes it possible to refer to individual parts of an XML document
- XQuery is to XML what SQL is to relational databases, although currently only for reading data
- XML namespaces enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring
- XML Encryption defines the syntax and processing rules for encrypting XML content
- ...

XML Schema and XSD

- XML Schema can be used to express a set of rules (a schema) to which an XML document must conform in order to be considered 'valid' according to that schema
- XSD is an instance of an XML schema. It defines a type of XML documents in terms of constraints upon what elements and attributes may appear, their relationship to each other, what types of data may be in them, and other things.

XSD Example

studentInfo.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="studentInfo" type="Student"/>
    <xs:complexType name="Student">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="affiliation" type="xs:string"/>
            ...
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

XML document that conforms to the above schema:

```
<studentInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentInfo.xsd">
    <name>Mario Rossi</name>
    <affiliation>University of Pisa</affiliation>
    ...
</studentInfo>
```

Ontologies (the computer science view)

- Formal, explicit specification of a shared conceptualisation
- Vocabulary of terms, and some specification of their meaning
- Goal: create an agreed-upon vocabulary and semantic structure for exchanging information about that domain

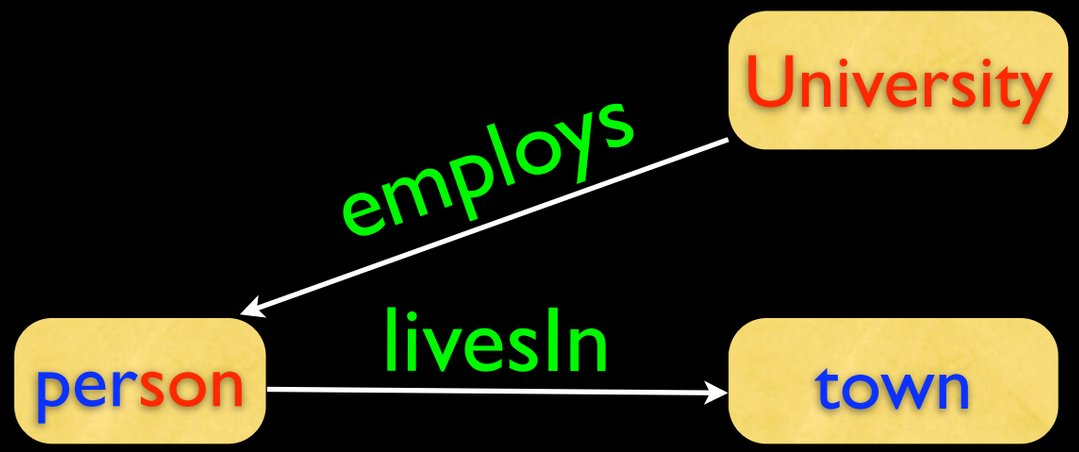
Taxonomies (vs. Ontologies)

- Taxonomy means classifying things
- Taxonomies are usually hierarchical (trees)
- Taxonomies are not enough to describe semantics

Ontology Langs for the SW

- Plethora of languages
- Most successful:
  - RDF (graphics based)
  - OWL (description logics based)
- Common core:
  - object/instance/individual (element of the domain),
  - type/class/concept (set of objects sharing certain characteristics),
  - relation/property/role (pair (tuple) of objects).

The *R*esource *D*escription *F*ramework



- Graphical formalism, where statements are <subject, predicate, object>
- Uses the notion of "resource" (anything pointed by an URI)
- Statements are properties of resources

RDF

- A Resource is anything that can have a URI
- A Property is a Resource that has a name and can be used as a property
- A Statement consists of the combination of a Resource, a Property, and a value

RDF Syntax

Well-defined XML syntax:

<description about=”subject”>

  <predicate resource=”object”/>

</description>

or

<description about=”subject”>

  <predicate> object </predicate>

</description>

RDF Schema

- RDFS = RDF + constructors (e.g., Class, subClassOf, Property, range, domain, aso.)
- Constructors are used to create vocabularies, e.g.:
  - <Professor, subClassOf, Person>
  - <employs, range, Person>
- No distinction btw. classes and instances or btw. constructors and vocabulary terms

RDFS Issues

- Cannot model:
  - properties with conditional domains/ranges (the range of "eats" is "meat" for "carnivores" and "vegetables" for "vegetarians")
  - existential/conditional constraints (any "computer" has a "CPU")
  - transitive, inverse, or symmetrical properties ("ancestor" is transitive)

OWL

- Markup language for publishing and sharing data using ontologies
- Developed by the WebOnt group
- Based DAML + OIL
- Extends a DL subset of RDF
- W3C Candidate Recommendation

OWL Versions

- OWL Full = OWL syntax + RDF
- OWL DL = extends a DL subset of RDF; restriction to FOL (based on SHOIN(D))
  - well-def semantics
  - complexity and decidability (for DL and Lite every statement can be decided in finite time)
  - reasoning algorithms
- OWL Lite = subset of OWL DL (based on SHIF(D))

OWL Lite

- It supports:
  - classification hierarchy
  - simple constraints (e.g., cardinality of 0/1)
- Low(er) formal complexity

OWL Lite

- RDFS features: Class, rdfs:subClassOf, rdf:Property, rdf:subPropertyOf, rdfs:domain, rdfs:range, individual
- Equality and inequality: equivalentClass, equivalentProperty, sameAs, differentFrom, AllDifferent
- Property characteristics: inverseOf, TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty
- Property restrictions: allValuesFrom, someValuesFrom
- Restricted cardinality: minCardinality, maxCardinality
- Class intersection: intersectionOf

OWL DL

- Maximum expressiveness while complete and decidable
- imposes restrictions on some OWL constructs (e.g., a class cannot be an instance of another class)
- OWL DL/Full add:
  - oneOf (e.g., dayOfTheWeek)
  - hasValue (a prop can be req to hava a certain individual as a value)
  - disjointWith (btw classes)
  - unionOf/complementOf/intersectionOf
  - minCardinality/maxCardinality/cardinality (full cardinality)
  - complex classes; classes as instances (OWL Full)

OWL Full

- Maximum expressiveness supporting RDF syntax yet without computational guarantees (e.g., a class can be an instance as well)
- No implementations yet!

Using OWL

1. Build an OWL ontology: create the ontology; name classes and properties and provide info about them
2. State facts about the domain (provide info about the individuals)
3. Reason about ontologies and facts (determine consequences about what was built and stated)

Building OWL Ontologies

- Determine the classes and the properties, as well as the domains and the ranges of the properties
- Add individuals and relationships
- Check whether the ontology is consistent and whether the classes are coherent

Creating an Ontology

- Ontology([name]owl:imports(<name>)...)
- OWL Classes:
  - collections of individuals (e.g., man)
  - classes can be equivalent (viz., same individuals), disjoint, intersected
  - Cardinality restrictions: minCardinality (on a prop w.r.t. a class)/maxCardinality
  - Class intersections: intersectionOf

    Class(pp:animal partial restriction(pp:eats someValuesFrom(owl:Thing)))

    Class(pp:person partial pp:animal)

    Class(pp:man complete intersectionOf(pp:person pp:male pp:adult))

Creating an Ontology

- OWL Properties:
  - collection of relationships among individuals (ObjectProperty) e.g., hasChild or among individuals and data (DataProperty), e.g., hasAge
  - properties are classes; a property can be a sub-property of another
  - Characteristics:
    - properties can be equivalent (viz., relate one individual to the same set of other individuals)
    - inverseOf/transitiveProperty (e.g., ancestor)/symmetricProperty (e.g., friend)
    - functionalProperty (viz., unique value)/inverseFunctionalProperty
  - Restrictions: allValuesFrom/someValuesFrom (viz., at least one value)

```
ObjectProperty(pp:eaten_by)
ObjectProperty(pp:eats inverseOf(pp:eaten_by) domain(pp:animal))

ObjectProperty(pp:has_pet domain(pp:person) range(pp:animal))

DataProperty(pp:naturals range(xsd:integer))
SubPropertyOf(pp:naturals pp:even)
```

Creating an Ontology

- OWL Individuals:
  - objects belonging to classes
  - related to other objects and data using props
  - sameAs/differentFrom/allDifferent

Individual(pp:Tom type(owl:Thing))

Individual(pp:Dewey type(pp:duck))

Individual(pp:Mick type(pp:male) value(pp:has_pet pp:Dewey))

Mad Cow's Contradiction

Class(pp:cow partial pp:vegetarian)

Class(pp:mad+cow complete intersectionOf(

pp:cow

restriction(pp:eats someValuesFrom(intersectionOf(

pp:brain

restriction(pp:part_of someValuesFrom pp:sheep))))))

Tom the Cat

```
ObjectProperty(pp:has_pet domain(pp:person) range(pp:animal))

Class(pp:old+lady complete intersectionOf pp:elderly pp:female pp:person))

Class(pp:old+lady partial intersectionOf(
    restriction(pp:has_pet allValuesFrom(pp:cat)
    restriction(pp:has_pet someValuesFrom(pp:animal)))))
Individual(pp:Minnie type(pp:elderly) type(pp:female) value(pp:has_pet pp:Tom))
```

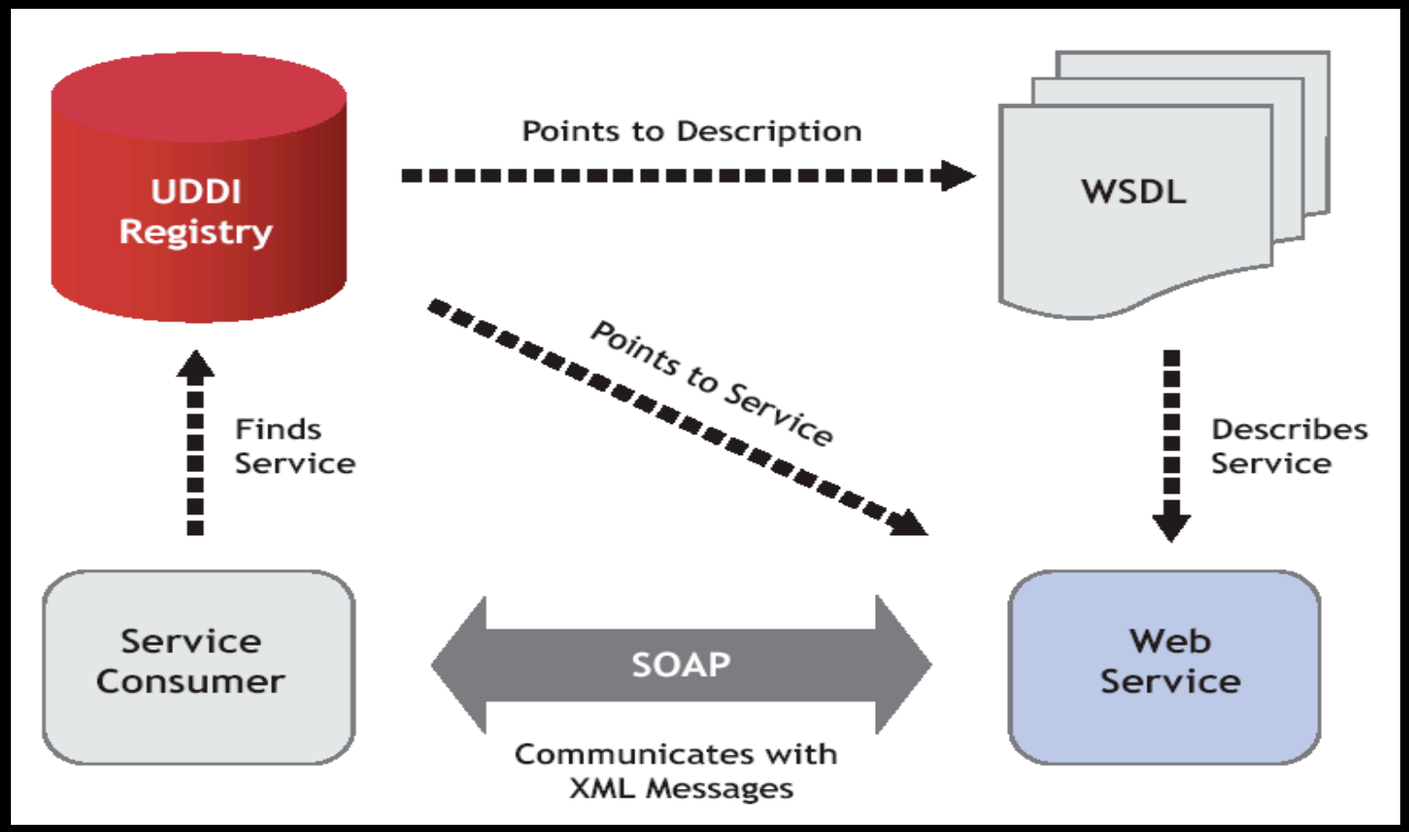
Airport.owl

```
<owl:Ontology about=""/>
      <owl:versionInfo/>...</owl:versionInfo>
      <rdfs:comment/>Airport</rdfs:comment>
</owl:Ontology>

<rdfs:Class id="Airport"/>
      <rdfs:subClassOf/>
            <owl:Restriction/>
                  <owl:onProperty resource="#name"/>
                    <owl:allValuesFrom resource="http://www.w3.org/2001/XMLSchema#string"/>
            </owl:Restriction>
      </rdfs:subClassOf>

<rdfs:subClassOf/>
      <owl:Restriction/>
            <owl:onProperty resource="#iataCode"/>
              <owl:allValuesFrom resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
</rdfs:subClassOf>

...
<owl:DatatypeProperty id="name"/></owl:DatatypeProperty>
...
```

Semantic Web Services

SoC Issues

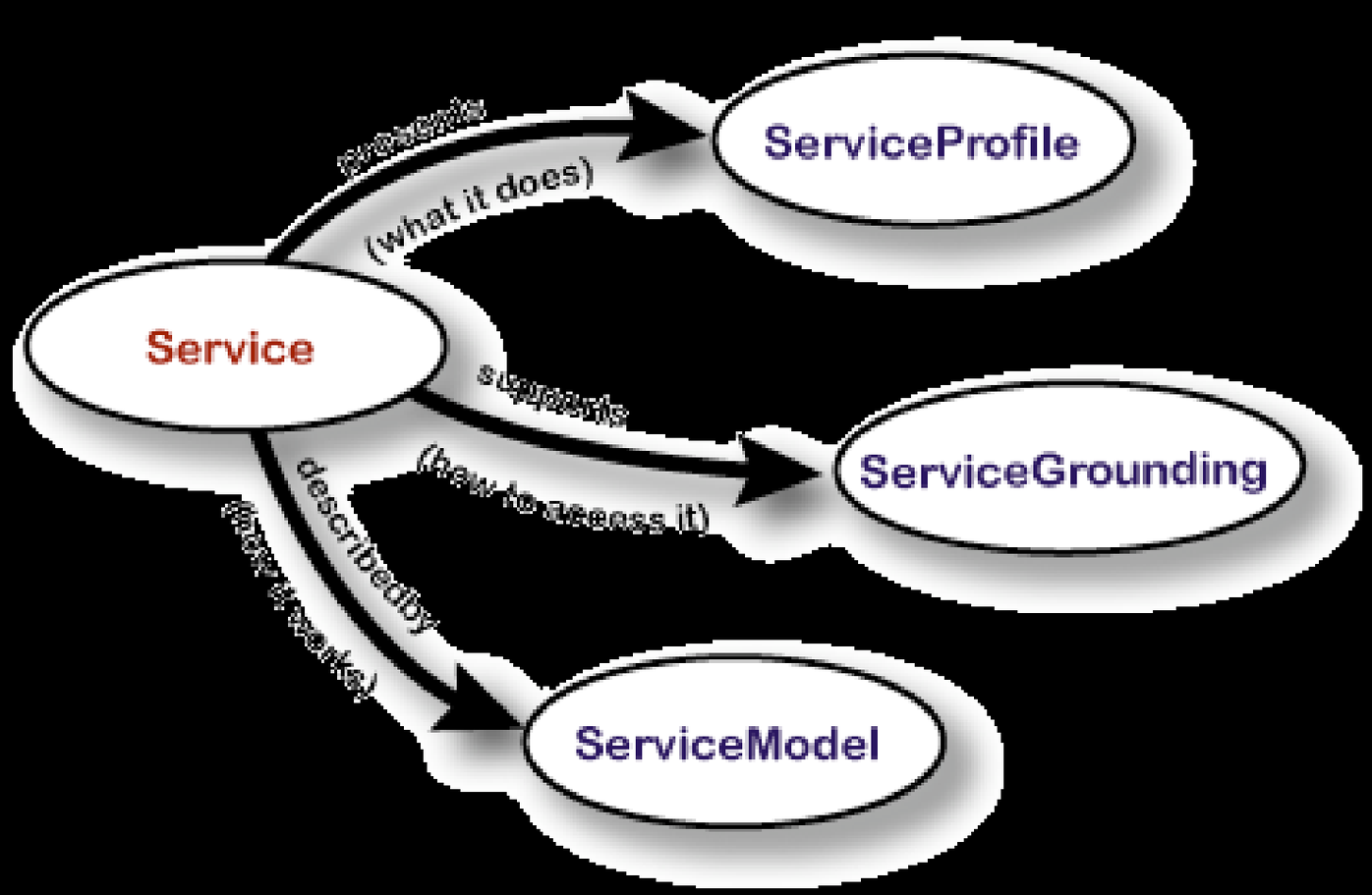
- WSDL -- purely syntactic
- UDDI -- keyword/ taxonomy based



- Signature mismatches affect: discovery/composition/adaptation
- Need for ontology info so as to match IOs/(sub-)services
- Benefits: automated techniques for the ontology-aware discovery of (composite) services, as well as for service composition and adaptation

OWL-S 1.1: An upper ontology for Services

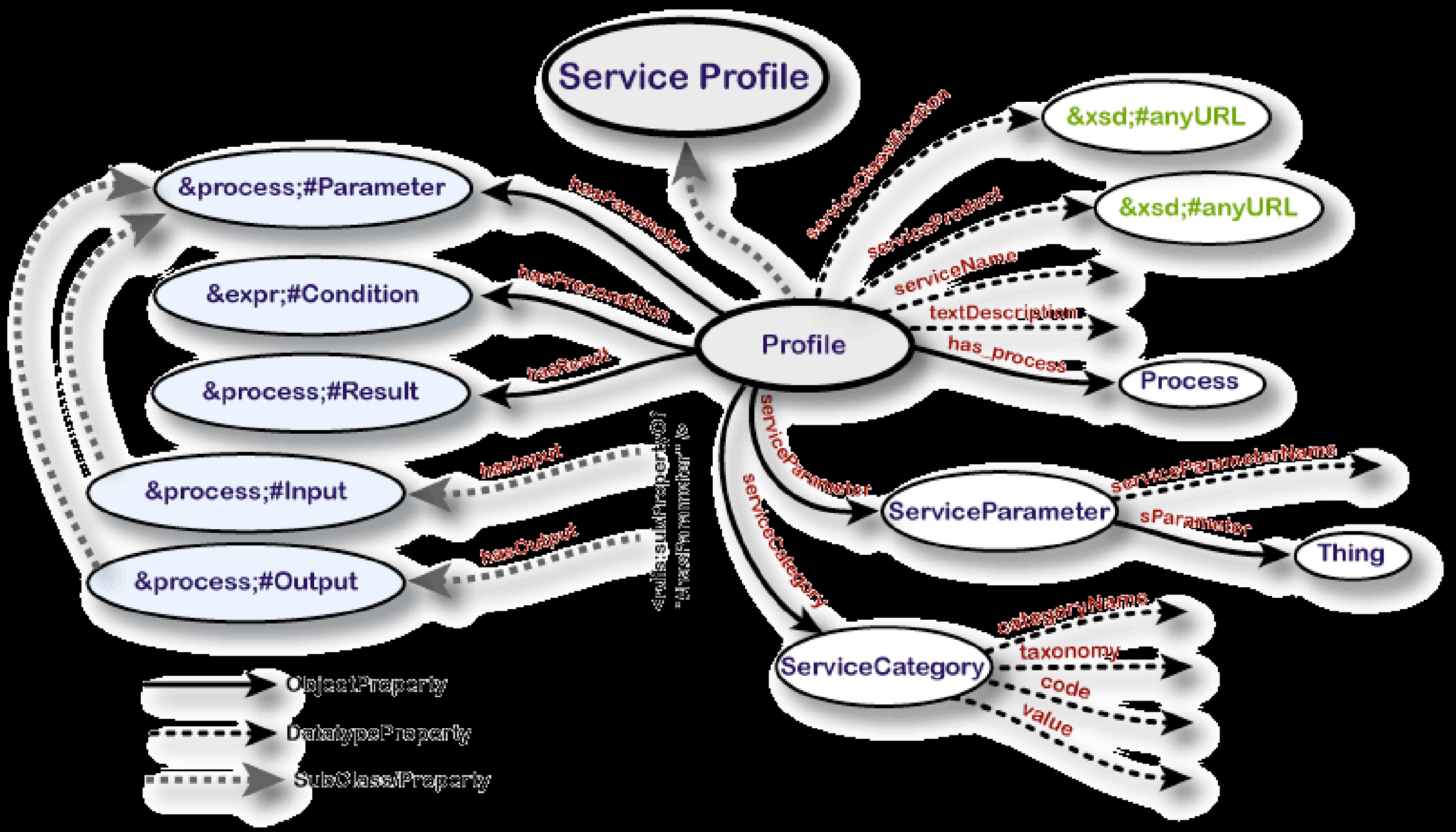
- Goals: automated service discovery/ composition/invocation/monitoring



OWL-S Service Profile

- Black-box view of the service:
  - provider info
  - functionality: IOPEs
  - Service features:
    - service category (e.g., UNSPSC)
    - QoS rating
    - max response time, geographic availability, aso.
- A service can have multiple service profiles

OWL-S Service Profile



OWL-S Service Profile (Profile.owl)

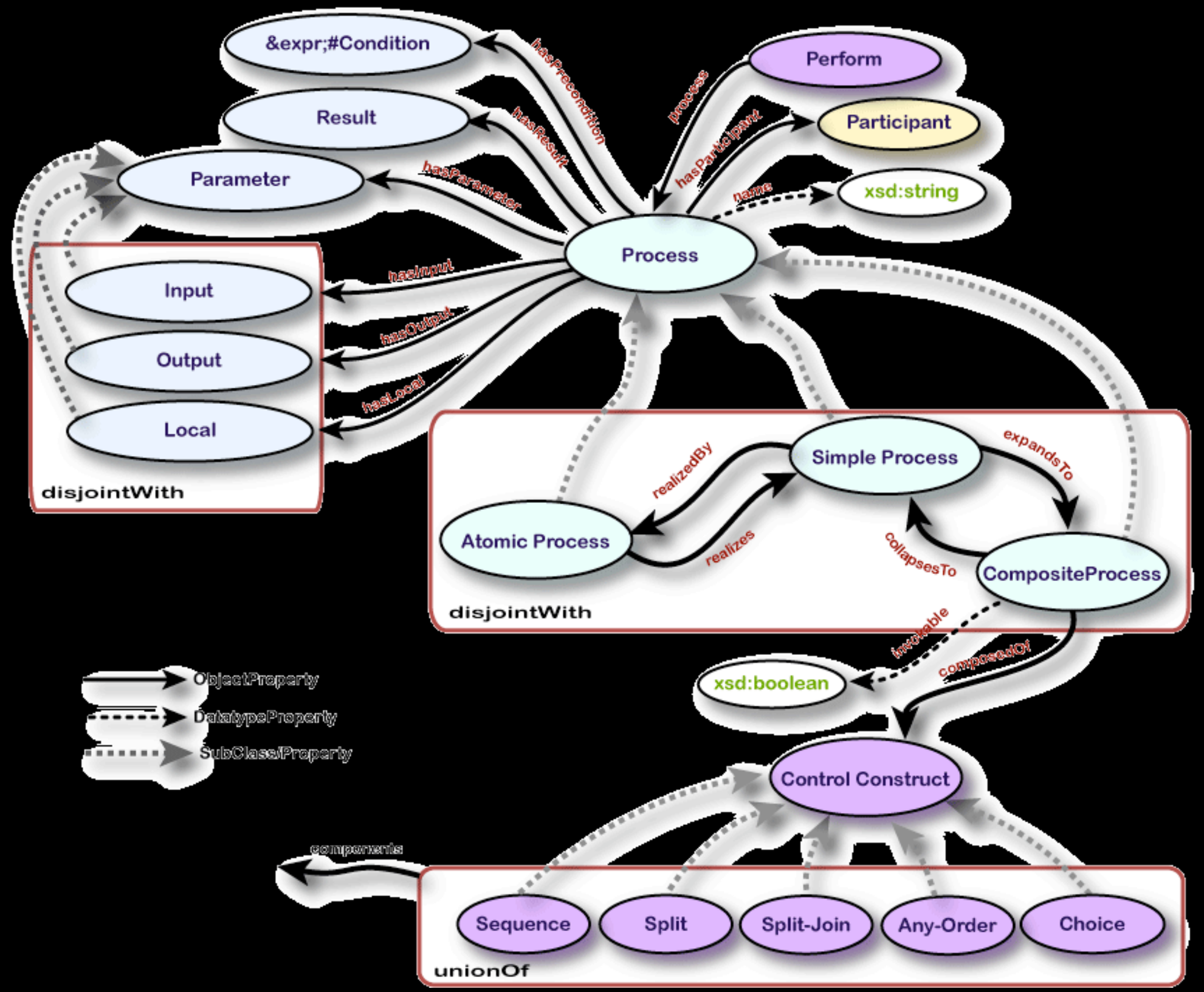
```
<owl:Class rdf:about="#Profile">
  <rdfs:comment>A profile can have only one name</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
          <owl:onProperty rdf:resource="#serviceName"/>
      <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
...
 <owl:ObjectProperty rdf:ID="hasParameter">
  <rdfs:domain rdf:resource="#Profile"/><rdfs:range rdf:resource="&process;#Parameter"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasInput">
 <rdfs:subPropertyOf rdf:resource="#hasParameter"/><rdfs:range rdf:resource="&process;#Input"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasOutput">
 <rdfs:subPropertyOf rdf:resource="#hasParameter"/><rdfs:range rdf:resource="&process;#Output"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasPrecondition">
 <rdfs:domain rdf:resource="#Profile"/><rdfs:range rdf:resource="&expr;#Condition"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasResult">
 <rdfs:domain rdf:resource="#Profile"/>
 <rdfs:range rdf:resource="&process;#Result"/>
</owl:ObjectProperty>
```

OWL-S Process Model

- A service has a unique Process Model
- Service as a process tree
- Process Types: atomic*/simple/composite
- Composite procs: sequence, split, choice, repreat-until, aso

* the only processes directly invocable by the

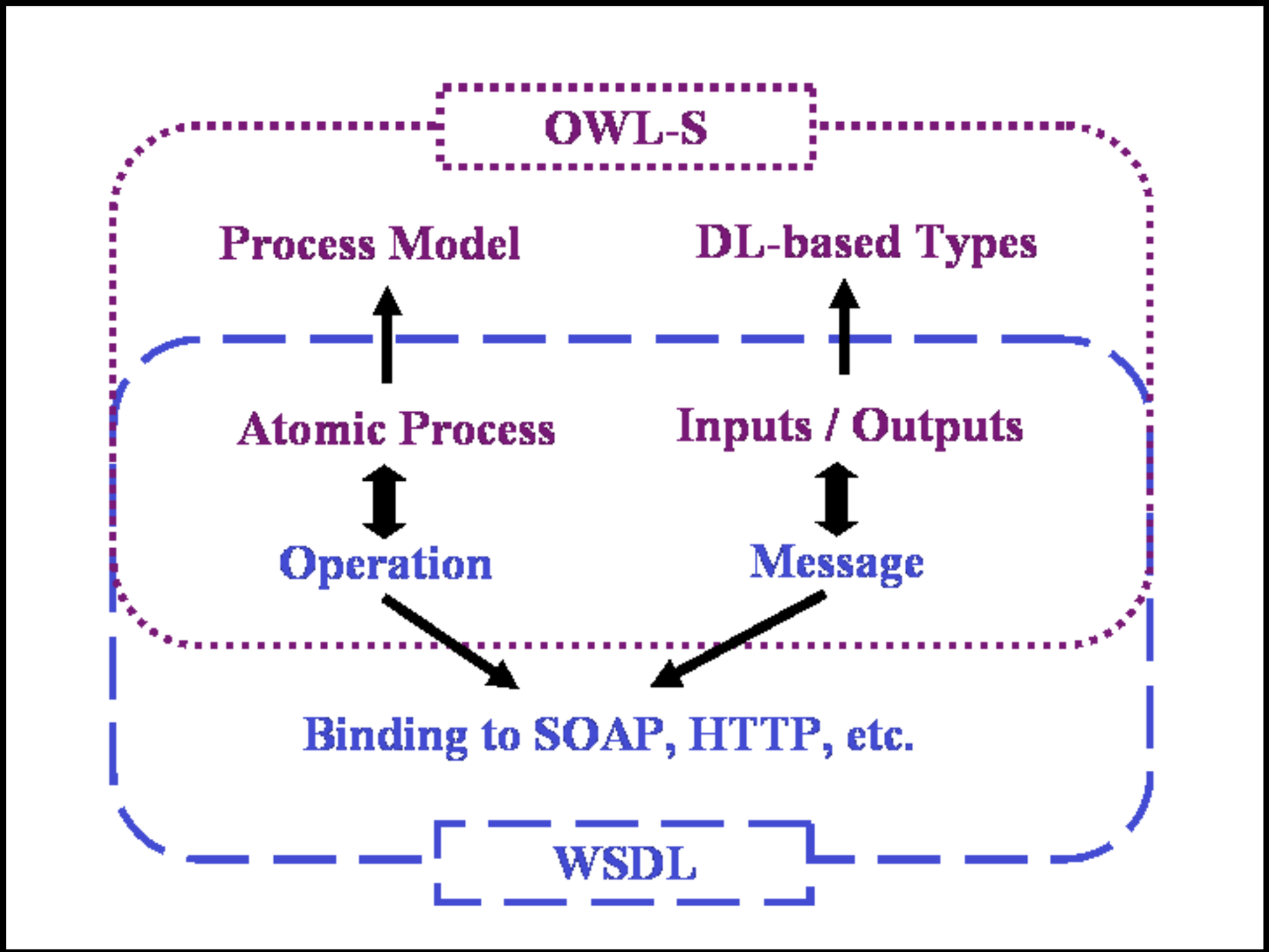
OWL-S Process Model



OWL-S Grounding

- Concrete specialisation of the service
- Detailed info on how to access and invoke the service -- protocol and message info
- IOs of atomic procs mapped to WSDL
- A service has a unique grounding

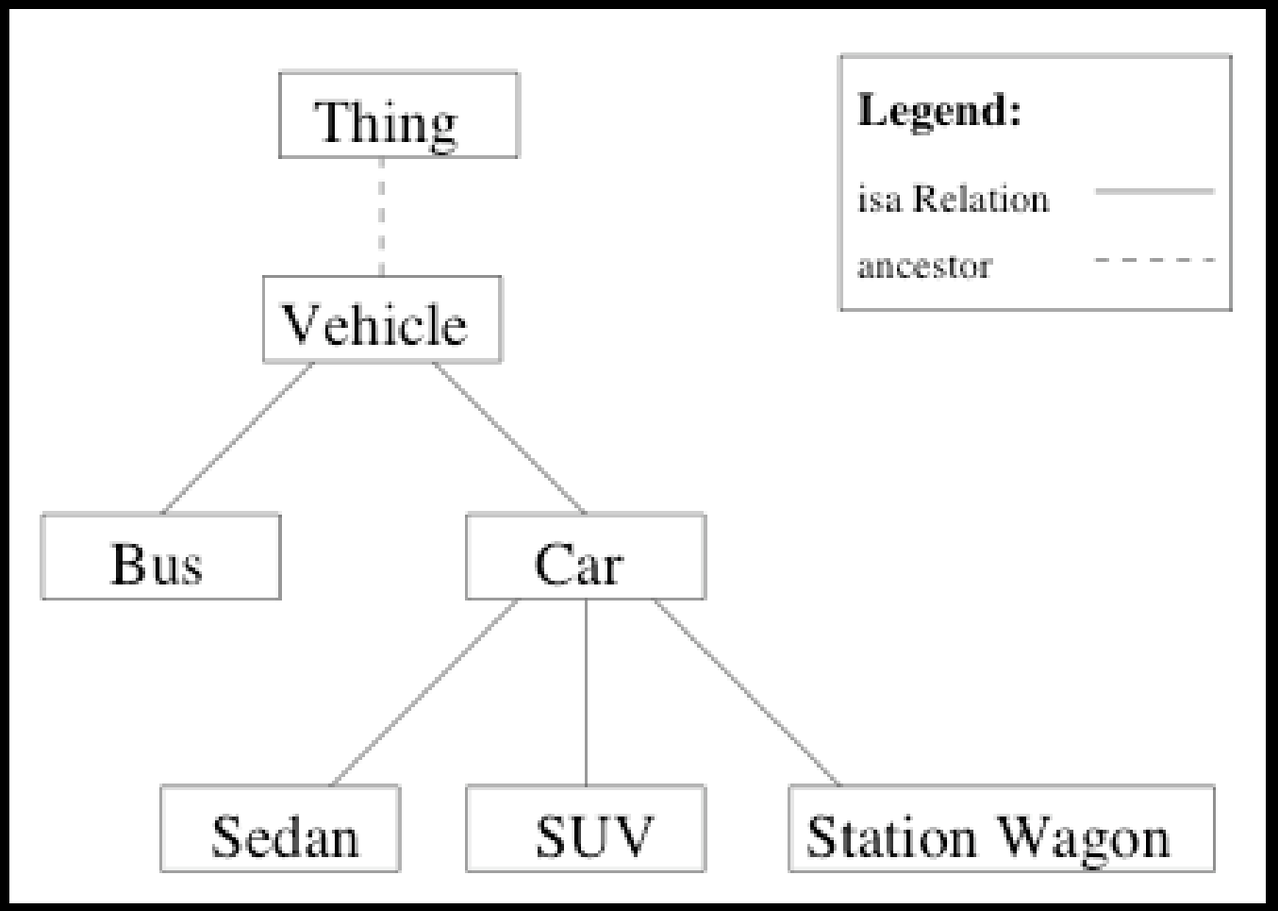
OWL-S Grounding



Semantic Matching of WS Capabilities (Paolucci et al.)

- IO-based (single) service matching
- Match condition: Ireq ∈ Iadv and Oadv ∈ Oreq
- degreeOfMatch(o_req, o_adv)=
  - *exact* if o_req=o_adv or o_req subclassOf o_adv
  - *plugIn* if o_adv subsumes o_req
  - *subsumes* if Oreq subsumes Oadv

degreeOfMatch Example



Other Approaches to SWS

- SWSF (SWSL and SWSO): Semantic Web Services Framework, http://www.daml.org/services/swsf/1.0/overview/
- WSMO (WSML): Web Service Modeling Ontology, http://www.wsmo.org/
- METEOR-S: Semantic Web Services and Processes, http://lsdis.cs.uga.edu/projects/meteor-s/
- WSDL-S: Web Service Semantics, http://www.w3.org/Submission/WSDL-S/

+/-

+ Give semantics (meaning) to information

+ Ontology-aware matching enhances info retrieval and service discovery/ composition/adaptation

+ Ontology tools (e.g., Protégé)

+ APIs for reasoning (e.g., Java DL reasoners OWLJessKb, DAMLJessKb, DAML4Jess)

- Unrealistic to believe that everybody will use the same ontology(ies); diff. to achieve cross-ontology mapping

- Reasoning is a time-consuming process

- Too many service profiles to represent all possible sub-services

- XML ~ ASM

- Is parameter ontology enough? (e.g., what is f(data, location):price?)

References

- The Semantic Web: Berners-Lee, T., Hendler, J., Lassila, O., "*The Semantic Web*", Scientific American, May 2001
- RDF: http://www.w3.org/RDF/
- RDF Schema: http://www.w3.org/TR/rdf-schema/
- OWL: http://www.w3.org/TR/owl-features/ (overview)
- OWL-S: http://www.daml.org/services/owl-s/1.1/overview/ (technical overview)
- Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. 2002. "*Semantic Matchmaking of Web Services Capabilities*". In First International Semantic Web Conference on The Semantic Web, LNCS 2342, I. Horrocks and J. Hendler, Eds. Springer-Verlag, 333–347
- Presentation partially based on slides from:
  - *Current Efforts towards Semantic Web Services (SWS): OWL-S and WSMO,* BIT-Seminar, 16/03/2005, Bolzano, by Axel Polleres
  - *Tutorial on OWL + An Example OWL Ontology*, ISWC, Sanibel Island, Florida, USA, 20th October, 2003, by Sean Bechhofer, Ian Horrocks, and Peter F. Patel-Schneider
- www.wikipedia.com

The Semantic Web
(Semantic Web Services)

Răzvan Popescu
University of Pisa