UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

DIPARTIMENTO DI INFORMATICA SISTEMISTICA E COMUNICAZIONE

Ph.D. course: Advanced Distributed Systems Development with Multiagent Systems

Dott. Daniela Briola

Prof.ssa Viviana Mascardi

Prof. Rafael Bordini

Lesson 2
A fast introduction to OWL ontologies

Daniela Briola
Daniela.briola@disco.unimib.it

Ph.D. Course: Advanced Distributed Systems Development with Multiagent Systems

What is an ontology?

- Many definitions. We adhere to the Grubers's one (1997):
  - An ontology is an explicit specification of a conceptualization
- An ontology should be:
  - Conceptual: an abstract model of the main concepts of a domain
  - Explicit: concepts, properties and constraints of the domain should be stated
  - Formal: in a machine readable format
  - Shared: should model a shared knowledge, and be accepted by the end users

Why using an ontology?

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge
- To give a Semantic to a syntax

- They are (it depends on language) more expressive than UML, ER etc, and not related to a language (Java, OO etc) nor to physical implementation (Relational Database etc)

Main entities

- Concepts (Class): the conceptual entities of a domain
- Individuals: concrete instances of the Concepts
- Relationships (between concepts)
- Attributes (of the concepts)
- Axioms (constraints over the concepts and, depending on the language, individuals)

- Formal ontologies support (with limits):
  - Verification and Validation (of the schema and of the instances)
  - Automatic inference
  - Reasoning

RDF: a simple [ontology] language

- class
  - subClassOf
- property
  - subPropertyOf
  - Domain and Range (of the property)
- IS-A (for defining instances)
- RDF models all as a triple (relation attribute-value)
- Very simple inference is supported
- Has a dedicated query language (SPARQL)

OWL

- Based on RDF
- W3C standard
- OWL and OWL2, based on Description Logic
- We focus on OWL DL (that is restricted to First Order logic)
  - Decidable
  - Based on long research on DL
  - Some reasoners available
  - API to programmatically use it
- It is the most widely adopted language for ontology development
- Often used for the Semantic Web

OWL (main features)

- Supported features:
  - Equivalent and Disjoint (for concepts and properties)
  - Same and Different (for instances)
- Facets of properties:
  - DataType (range into a literals, like int or String)
  - ObjectType (range into Concepts)
  - cardinality
- Properties main characteristics:
  - Symmetry
  - Transitivity
  - Inverse
  - Functional (min cardinality 0, max cardinality 1)

Methodology for developing ontologies

- There are many methodologies, from simple to complex ones
- We refer to the simplest and most intuitive one: 101 [Noy]
  - Step 1: Determine the domain and scope of the ontology
  - Step 2. Consider reusing existing ontologies
  - Step 3. Enumerate important terms in the ontology
  - Step 4. Define the classes and the class hierarchy
  - Step 5. Define the properties of classes—slots
  - Step 6. Define the facets of the slots
  - Step 7. Create instances

N. F. Noy, D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology", *Tech. Rep. March 2001*, [online] Available: http://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

Practical exercise: LEARN domain

- We are developing a MAS physical distributed over end user machine (one LEARN platform on each machine)
  - Platforms must have a name and an IP
- We can monitor one, or more, applications installed by the user on its pc
- An application can experience an internal state of error (not visible to the user but only to a monitor), which can then lead to a failure of the system (crash or wrong behaviour)
  - We need to keep the list of known failures and, if known, related state of errors

LEARN domain (cont)

- Agent services:
  - To provide the list ok known platforms
  - To provide the list of monitored applications
  - To provide the list of experienced failures or errors of one application
  - To notify about a new failure or error of one application
  - To add a new failure or error for an application

That was for the ontology…

- What about “when and why” invoking the services? In a proactive or reactive way? Subscribing to a service?
- How to let the receiver simply understand the type of message (a request to execute a service, an answer, a notification…)?
  - Speach at theory: communications not only transmit information, but represent actions which change the state of the world
  - For example:
    - Request
    - Inform
    - Failure
    - Query-If