

# **Linked Data for Digital Humanities**

- HUMN3001 and HUMN6003 -

## **Ontologies**

**Terhi Nurmikko-Fuller**



[terhi.nurmikko-fuller@anu.edu.au](mailto:terhi.nurmikko-fuller@anu.edu.au)

# Plan

- What's an ontology?
- Examples of existing ontologies
- RDFS, OWL
- Ontology design



Sorry, Parmenides, not you!

*So the Semantic Web is all about meaning...*

*How do we exchange meaning through RDF?*

- Ontologies.*

# What's an ontology?

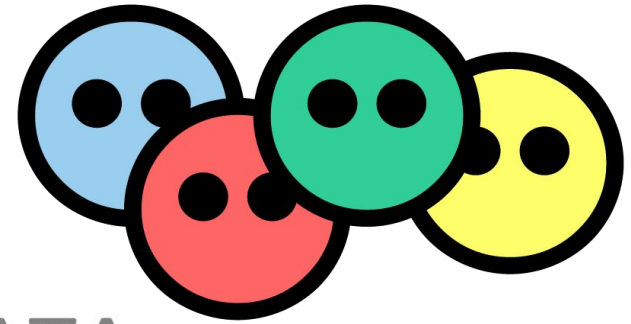
- An ontology is a description of concepts and their relationships.
- It enables us to build semantic models with RDF (more specific models than RDF itself).
- It's about adding meaning to your data so that it can be “understood” and reused by others.

# Appropriate ontologies

- Ontologies don't remove complexity, but they do enable us to scale it (relatively) gracefully.
- More than one “correct” ontology can be applicable to a resource – it depends what you're doing with it.
- Where available, use an applicable existing ontology (or extend it).
- Write the ontologies you need – you can extend them later. Better it be limited and right...
- It is probably unwise to expect an ontology for all Things...

# Ontologies encountered...

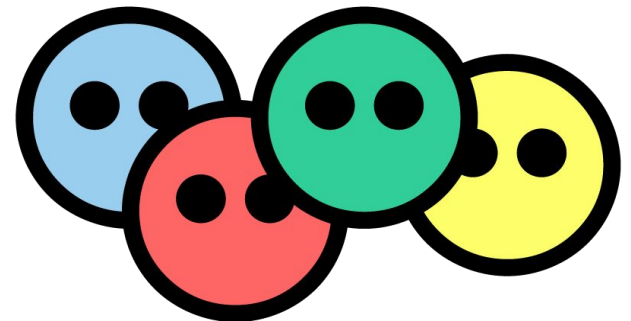
- Friend-of-a-Friend (FOAF)
- Dublin Core
- DBpedia
- MODS RDF
- MADS RDF
- BIBFRAME
- eeboo
- Linkedjazz
- JazzCats



...and more!

# Useful Ontologies

- **RDF Schema** <http://www.w3.org/TR/rdf-schema/>
- **XSD** [https://en.wikipedia.org/wiki/XML\\_Schema\\_\(W3C\)](https://en.wikipedia.org/wiki/XML_Schema_(W3C))
- **Dublin Core** <http://dublincore.org/>
- **SKOS** <http://www.w3.org/2004/02/skos/>
- **FOAF** <http://xmlns.com/foaf/spec/>



# Some in Cultural Heritage

- **CIDOC-CRM**

- [http://www.cidoccrm.org/official\\_release\\_cidoc.html](http://www.cidoccrm.org/official_release_cidoc.html)

- **FRBRoo**

- [http://www.cidoc-crm.org/frbr\\_drafts.html](http://www.cidoc-crm.org/frbr_drafts.html)

- **PerioDO**

- <http://perio.do/specs/>

- **Europeana Data Model**

- <http://pro.europeana.eu/page/edm-documentation>



# Bibliographical metadata

- **MODS/RDF**

- <http://www.loc.gov/standards/mods/modsrdf/>

- **MADS/RDF**

- <http://www.loc.gov/standards/mads/rdf/v1.html>

- **Bibframe**

- <http://www.loc.gov/bibframe/docs/>

- **BiBO**

- <http://bibotools.googlecode.com/svn/bibo-ontology/trunk/doc/classes/>

- **Schema.org**

- <http://schema.org/>

# Other examples (1)

- **Music Ontology**

- <http://musicontology.com/>

- **Timeline Ontology**

- <http://motools.sourceforge.net/timeline/timeline.html>

- **Event Ontology**

- <http://motools.sourceforge.net/event/event.html>

- **Media ontology**

- <http://www.w3.org/TR/mediaont-10/>

- **Prov**

- <http://www.w3.org/TR/prov-o/>

# Other examples (2)

- **RO**

- <http://www.researchobject.org/specifications/>

- **Open Annotation**

- <http://www.openannotation.org/spec/core/>

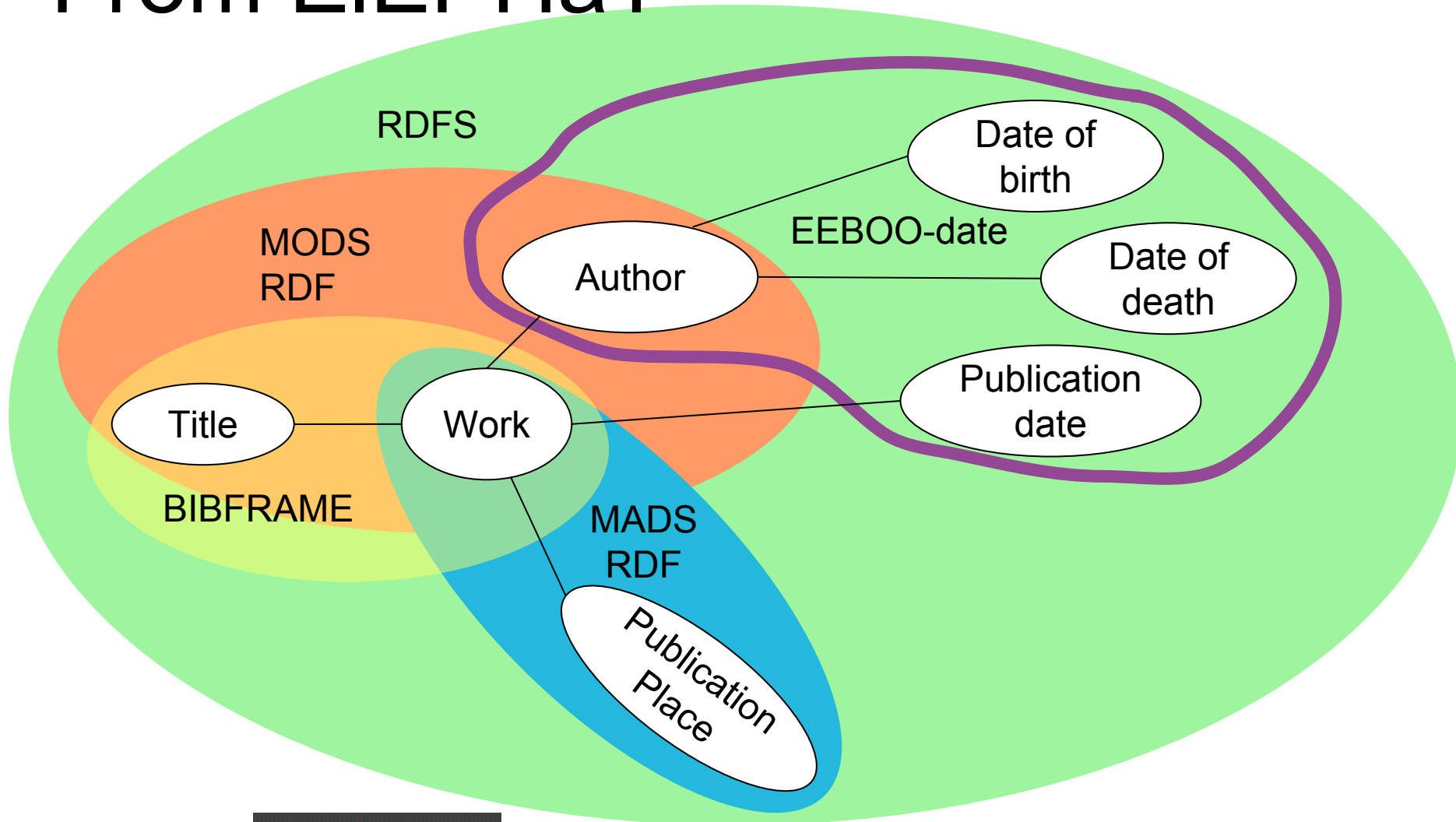
- **Owl Time**

- <http://www.w3.org/TR/owl-time/>

- **Segment ontology**

- <http://www.linkedmusic.org/ontologies/segment/>

# From EIEPHãT



# Ontologies in EIEPHãT

- Each of these conceptual areas is a specialisation
  - which might be the subject of scholarly study
  - or computational analysis
  - or crowdsourcing, etc.
- There will be overlap
  - one person's metadata is another person's data
  - we can build upon others specialisation and knowledge.
- We do not expect complexity to vanish
  - but where it has been studied it should be scaled, shared, and linked.

*How do we express our ontologies?*

*- Why, in RDF of course!*

# RDFS and OWL (intro)



- RDFS: RDF Schema
  - The basics required to structure
    - an ontology and
    - exchange vocabularies
  - Classes and properties, super- and sub-classes, range and domain.
- OWL: Web Ontology Language
  - More sophisticated structures
  - Constraints for existence and cardinality, transitive, inverse, symmetrical properties, ...

# Structuring RDF

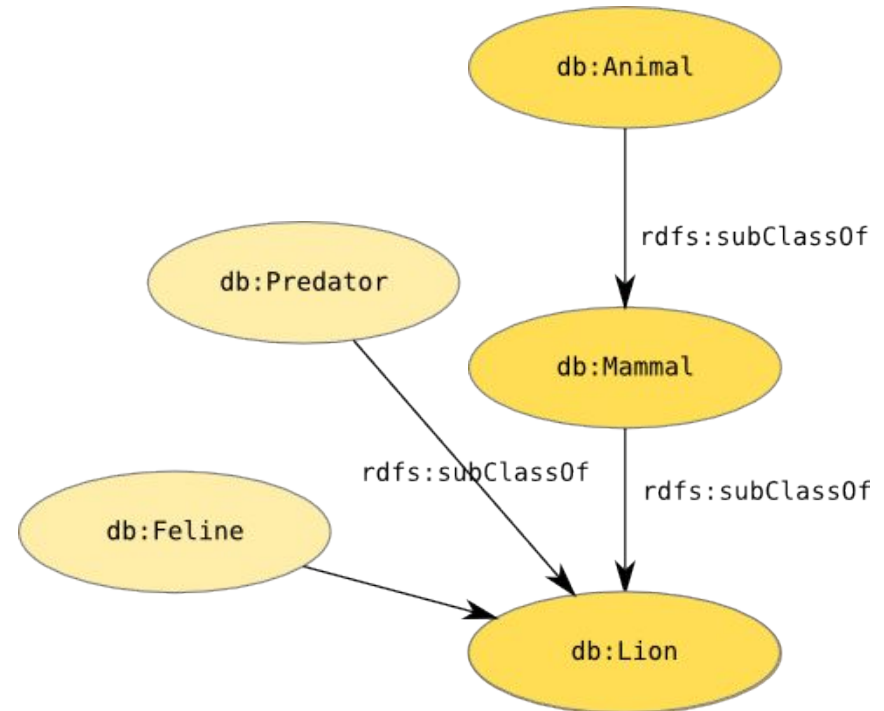
- Little value if there's no structure at all.

- RDF Schema

- Define Classes of resources
- Define Properties
- SubClasses

- the isA relationship

- Represented in RDF itself



db: → <http://live.dbpedia.org/resource/>

rdfs: → <http://www.w3.org/2000/01/rdf-schema#>



# RDF Schema (detail)

<code>rdf:type</code>	property to state that a resource is an instance of a Class
<code>rdfs:Class</code>	resources that are RDF classes ( <code>rdfs:Class</code> is an instance of <code>rdfs:Class</code> )
<code>rdf:Property</code>	class of RDF properties ( <code>rdf:Property</code> is an instance of <code>rdfs:Class</code> )
<code>rdfs:subClassOf</code>	isA relationship between classes
<code>rdfs:subPropertyOf</code>	isA relationship between properties
<code>rdfs:label</code>	a human-readable version of a resource's name
<code>rdfs:seeAlso</code>	a resource which might provide more information about the subject
...	

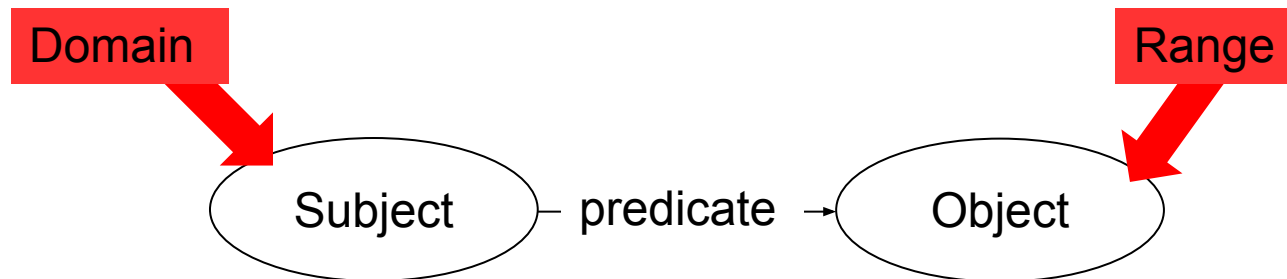
`rdf:` → <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

`rdfs:` → <http://www.w3.org/2000/01/rdf-schema#>

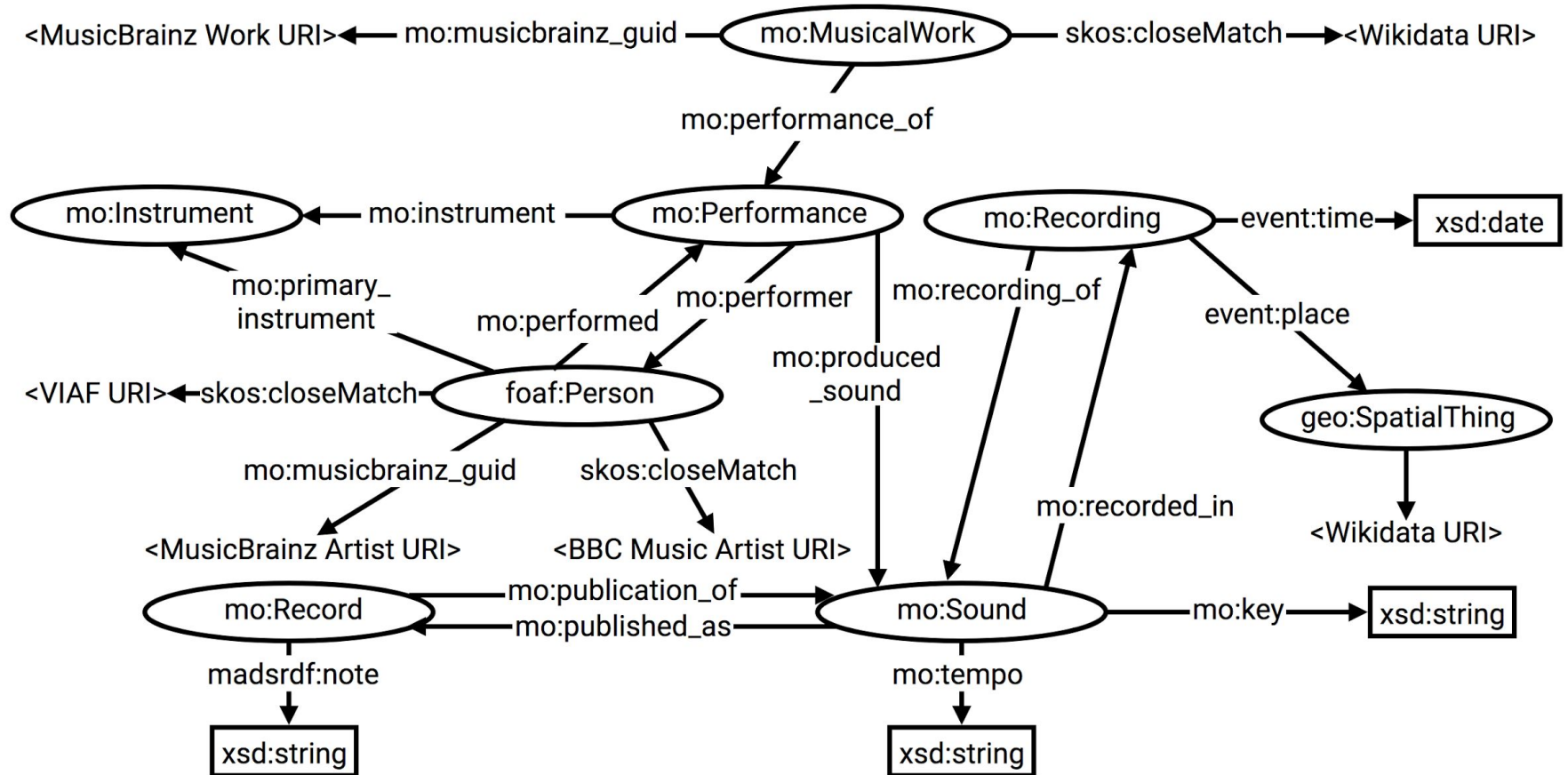
# Domain + Range

- A Property can define a Domain and Range
  - Domain is the Class of the subject.
  - Range is the Class of the object.

Properties run from the Domain, to the Range.



# Example of the JazzCats ontological structure



Music Ontology

SKOS

Event ontology

Geo ontology

MADS/RDF

XSD

The rest are specifics URIs for resources and data entities, or strings, or integers.

# How do we know what they mean?

## Specifications and Documentations

- Yes, this does mean you have to spend (a really long!) time reading these documentations with your human eye balls and human brain.
- This is your responsibility as a human user to find out and use the right Class or property. The technology won't stop you from doing something stupid.

e.g. Music Ontology: <http://motools.sourceforge.net/doc/musicontology.html>

*This is awesome, but I haven't found an ontology that matches my data and research aims perfectly. What can I do?*

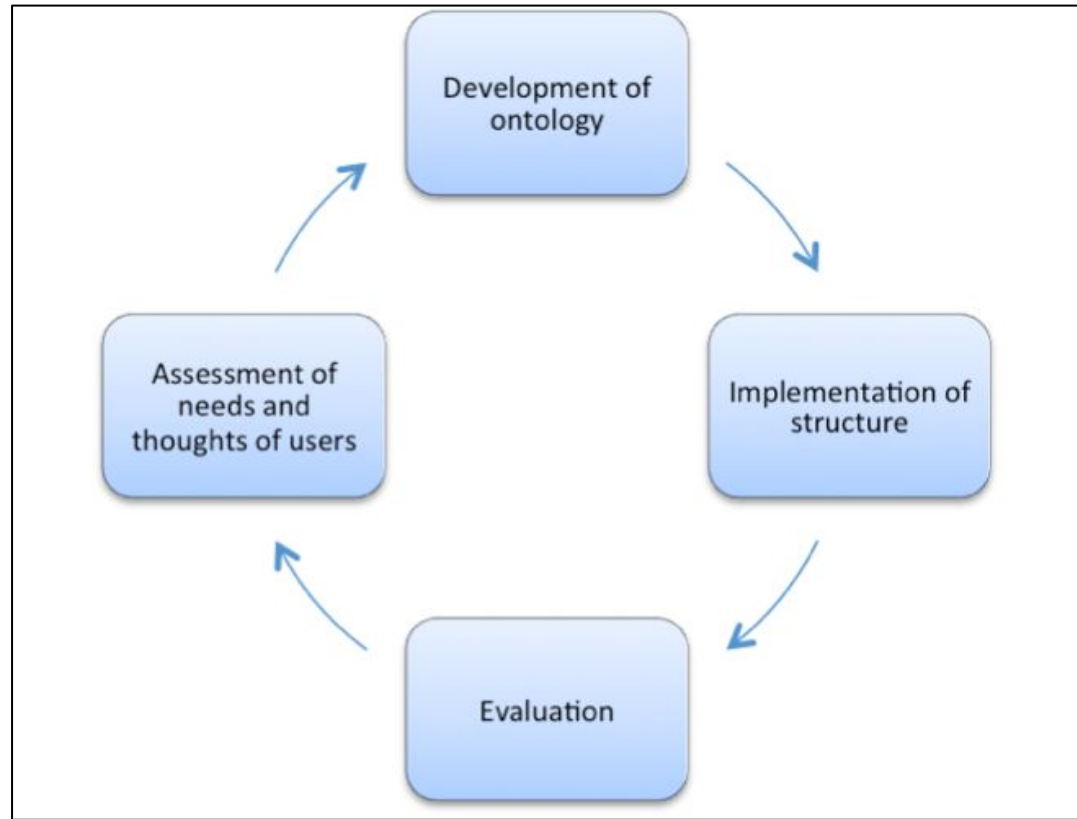
*- Well, you can design your own ontology!*

# You just need to...

- Get a model right...



# Ontology design (1)



# Ontology design (2): 6 steps

## 1. Specification

- reasons and aims of the ontology are assessed and determined

## 2. Conceptualisation

- structure, classes and properties are planned out

## 3. Formalisation

- ideas are realised in a model, and the hierarchy of concepts is defined

## 4. Implementation

- requires the selecting of
  - the language (e.g. RDFS, OWL),
  - the editor software (e.g. Protégé),
  - if applicable, and the reasoner (e.g. FaCT++ )

## 5. Evaluation

- the ontology is tested against SPARQL queries or through an online validator

## 6. **Documentation**

- information regarding the design decisions and the rationale are outlined for the benefit of other users.



The mind map illustrates the workflow for audio annotation. The central node is 'Audio Annotation'. It branches into three main categories: 'Audio Input', 'Annotation Process', and 'Annotation Output'. 'Audio Input' includes 'Microphone', 'Recorder', and 'Audio File'. 'Annotation Process' includes 'Annotation Software', 'Annotation Tools', and 'Annotation Methods'. 'Annotation Output' includes 'Annotation File', 'Annotation Report', and 'Annotation Summary'. The diagram is color-coded with green for input, yellow for process, and blue for output.

# Ontology design (4)

The screenshot displays the Protégé ontology editor interface. The top menu bar includes File, Edit, Project, OWL, Code, Tools, Window, and Help. The toolbar contains various icons for file operations and editing. The main window is divided into several panes:

- SUBCLASS EXPLORER:** Shows the asserted hierarchy for the project 'pizza.owl'. The hierarchy starts with 'owl:Thing' at the root, followed by 'DomainConcept', 'Country', 'IceCream', and 'Pizza'. Under 'Pizza', there are several subclasses: 'CheeseyPizza', 'InterestingPizza', 'MeatyPizza', 'NamedPizza', 'NonVegetarianPizza', 'RealItalianPizza', 'SpicyPizza', 'SpicyPizzaEquivalent', 'VegetarianPizza' (which is selected), 'VegetarianPizzaEquivalent1', and 'VegetarianPizzaEquivalent2'. Below these are 'PizzaBase', 'PizzaTopping', and 'ValuePartition'.
- CLASS EDITOR:** Shows the editor for the class 'VegetarianPizza'. It includes a table for properties and their values, and a section for asserted conditions.

**CLASS EDITOR Details:**

For Class: **VegetarianPizza** (instance of owl:Class) ☐ Inferred View

Property	Value	Lang
rdfs:comment	Any pizza that does not have fish topping and does not have meat topping is a VegetarianPizza. Members of this class do not need to have any toppings at all.	en
rdfs:label	PizzaVegetariana	pt

**Asserted Conditions:**

- NECESSARY & SUFFICIENT:**
  - Pizza**
    - not** (hasTopping **some** FishTopping)
    - not** (hasTopping **some** MeatTopping)
  - hasBase some PizzaBase** [from Pizza]

The bottom status bar shows the current view is 'Logic View' (selected) and 'Properties View' is also available.

# Protégé

1. Free Ontology editor developed at Stanford
  - a. <http://protege.stanford.edu/>
2. Has OWL support
3. Describing full use beyond today's session
  - a. <http://protegewiki.stanford.edu/wiki/Protege4GettingStarted>
    - i. for info on installing and running
  - b. <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>
    - i. Tutorials

# To Do Well on this Project:

- 50% on your ontology work.
- 50% on the RDF you produce next week using the ontology.

## You need to:

- Design an ontology that is *not wrong*.
  - There are many possible correct models. There is no one right answer.
  - There can be mistakes! A page is part of a book, not a type of book.
- Document your ontology.
- Add or include parts from an existing ontology/ontologies.

**What you call a specific Class or property is not that important. The documentation must makes the meaning clear, whether you call something a “work”, a “book”, or “12345”.**