Symbolic representation of knowledge has a long history in AI
- lisp: (list `ant `bat `cat)
- prolog: [ant, cat, bat]
  
mammal(X) :- cat(X).

Ontologies establish standards and conventions for domain knowledge
- Modelling: class, instance, relation
- Content: objects vs processes,
- Language: FOL, Description Logic
- Definitional not problem solving knowledge

Reminder

Application Ontologies
- Describe specific domains
  - Travel & Tourism; Medicine; Genetics
- Can have diverse uses
  - KBS (e.g. diagnosis); recommendation; web site / database design; data mining; NLP

Knowledge Representation Ontologies
- Define the formal vocabulary that specifies classes and relations
  - subClassOf; type
  - domain; range; subPropertyOf

Method 1
- ‘Methontology’ – see pdf files in ontology-reading.tar.gz

Method 2
- Protégé

Method 3
- Ontology design patterns
Methontology

- Emphasises
  - Ontology building as a craft
    » Considering Domain Ontologies cf KR or Generic Ontologies
  - Lack of tested and generalised methodologies
  - The developers move too quickly to implementation
    » Conceptual models get encoded in the implementation and not made explicit
    » Ontological commitments not made explicit
    » Experts are unable to critique formal ontologies
    » The ontology encoding language imposes a bias on what can be said, so developers do not consider what ought to be said

The answer - to build ontologies at the ‘knowledge level’, that is, at a level of abstraction above the encoding language

- The ontology evolves from a series of prototypes
  - Cyclical process
  - The ontology development process describes the activities to be performed during ontology engineering

- Lifecycle of development activities
  - Specification: identify uses and users
  - Conceptualisation: structures the domain knowledge
  - Formalisation: transforms conceptual model into a formal model
  - Implementation: build the computable model
  - Maintenance: updates and corrects the ontology

- Support activities
  - Knowledge acquisition
    » Predominantly at the beginning
    » Integration/Merging/Alignment
    » Bring in existing ontologies
  - Evaluation
    » Determine that modelling decisions are correct
  - Documentation

Methontology identifies the steps in the process
- Described as a life-cycle of evolving prototypes
- Beginning with knowledge acquisition and
- ending with a formal ontology

The Specification describes the
- Primary objective
- Purpose
- Granularity level
- Scope

Conceptualisation
- Organise the acquired knowledge
- Represent it in a way both domain expert and ontologist can understand
  » Intermediate representation - diagrams and tables
  » 11 steps that can be performed iteratively
1. Build the **Glossary of terms**, define each in natural language
   - carbon, hydrogen, element, gas, metal, conductor...
2. Build the **Concept taxonomy**:
   - Element
   - Reactiveness
   - Metal
   - Third transition series
3. Build the **Binary relations diagram**
4. Build the **Concept dictionary**

<table>
<thead>
<tr>
<th>Concept Name</th>
<th>Synonym</th>
<th>Acronym</th>
<th>Instances</th>
<th>Class Attributes</th>
<th>Instance Attributes</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Elm.</td>
<td></td>
<td>atomic-number</td>
<td>atomic-weight</td>
<td></td>
<td>has-structure/has-structure</td>
</tr>
<tr>
<td>Third transition series</td>
<td>3TS</td>
<td>Gold</td>
<td>Hafnium</td>
<td>Mercury</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concept dictionary in more detail:

- **Concept-Name**
  - A unique (singular) name for the concept
  - In certain cases identifiers are used GO:000999
- **Synonyms**
  - Alternative names for the concept
- **Instances**
  - Elements of the set represented by the concept
- **Relations**
  - Relations between (instances of) classes
  - E.g. `departurePlace(instance of Flight) <instance of Location)`
- **Class-Attributes**
  - Attributes that hold of the class
  - E.g. `companyName(KLM-Flight "KLM")`
- **Instance Attributes**
  - Attributes that can be specified for each instance
  - E.g. `weight(instance of Person) <weight-in-kg>`

**Included here for completeness only**

---

**A legal ontology for Spanish law [Corcho]**

1. Build the Glossary of terms, define each in natural language: *defendant, court, person, private company, *
2. Build the Concept taxonomy:
   - **LegalEntity**
   - **JuridicialPerson**
   - **Company [exhaustive decomposition]**
     - **PrivateCompany**
     - **PublicCompany**
   - **PhysicalPerson [partitioned into]**
     - **Juvenile**
     - **PersonLegallyOfAge**
3. Build the Binary relations diagram
4. Build the Concept dictionary

**Juridical: of law or the administration of justice**
1. Glossary of terms
   - Types are: concept, constant, relation, instance attribute

<table>
<thead>
<tr>
<th>Name</th>
<th>Synonyms</th>
<th>Acronym</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult age in Spain</td>
<td></td>
<td></td>
<td>Adult age is 18</td>
<td>constant</td>
</tr>
<tr>
<td>Court</td>
<td>Judicial tribunal</td>
<td></td>
<td>Although court can be understood as a physical place or as a judge, we assume that a court is a judicial tribunal</td>
<td>concept</td>
</tr>
<tr>
<td>Birthday</td>
<td></td>
<td></td>
<td>The day when a person was born</td>
<td>instance attribute</td>
</tr>
<tr>
<td>is-defendant (person,lawsuit)</td>
<td></td>
<td></td>
<td>Identifies the lawsuit a defendant is involved in</td>
<td>relation</td>
</tr>
</tbody>
</table>

2. Concept taxonomy: subclasses; partitions; disjoint and exhaustive decompositions

3. Binary relations diagram
   - Ad-hoc relationships between concepts
   - Determine domains and ranges

4. Concept dictionary
   - Organise all concepts and relations in 2. and 3.
   - Rows list relations/attributes whose domain is the Concept

<table>
<thead>
<tr>
<th>Concept Name</th>
<th>Synonym</th>
<th>Acronym</th>
<th>Instances</th>
<th>Class Attributes</th>
<th>Instance Attributes</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Court</td>
<td></td>
<td></td>
<td>Constitutional Court National Court</td>
<td>number of members seat</td>
<td>hears</td>
<td></td>
</tr>
<tr>
<td>Lawsuit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has-defendant has-plaintiff is-heard</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Is-defendant Is-plaintiff</td>
<td></td>
</tr>
<tr>
<td>Physical Person</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>age birthday is-mother-of has-mother is-father-of</td>
<td></td>
</tr>
</tbody>
</table>
5. Binary relations table
   - Describe all the binary relations in the Concept Dictionary (4.)

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Source concept (Domain)</th>
<th>Cardinality</th>
<th>Target concept (Range)</th>
<th>Inverse relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-defendant</td>
<td>Person</td>
<td>N</td>
<td>Lawsuit</td>
<td>has-defendant</td>
</tr>
<tr>
<td>is-plaintiff</td>
<td>Person</td>
<td>N</td>
<td>Lawsuit</td>
<td>has-plaintiff</td>
</tr>
<tr>
<td>has-defendant</td>
<td>Lawsuit</td>
<td>N</td>
<td>Person</td>
<td>is-defendant</td>
</tr>
<tr>
<td>is-heard</td>
<td>Lawsuit</td>
<td>N</td>
<td>Court</td>
<td>heard</td>
</tr>
</tbody>
</table>

6. Instance attributes table
   - Attributes for instances of concepts (values differ for each instance)

<table>
<thead>
<tr>
<th>Instance Attribute Name</th>
<th>Domain Concept</th>
<th>Value type</th>
<th>Value range</th>
<th>Cardinality (min,max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>number-of-members</td>
<td>Court</td>
<td>Integer</td>
<td>1..</td>
<td>(1,1)</td>
</tr>
<tr>
<td>territorial-jurisdiction</td>
<td>Court</td>
<td>String</td>
<td>---</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>

7. Class attributes table - attributes applicable to the class (values apply to the class name)

<table>
<thead>
<tr>
<th>Class Attribute Name</th>
<th>Concept</th>
<th>Value type</th>
<th>Cardinality (min,max)</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type-of-control</td>
<td>Private</td>
<td>[private,public]</td>
<td>(1,2)</td>
<td>public</td>
</tr>
<tr>
<td>type-of-control</td>
<td>Public</td>
<td>[private,public]</td>
<td>(1,2)</td>
<td>public</td>
</tr>
</tbody>
</table>

8. Constants table - describe the constants in the Glossary (1.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value type</th>
<th>Value</th>
<th>Measurement unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult age in Spain</td>
<td>cardinal</td>
<td>18</td>
<td>year</td>
</tr>
</tbody>
</table>

9. Instances table - describe the instances in the Concept dictionary (4.)

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>Concept Name</th>
<th>Instance Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>National Court</td>
<td>Court</td>
<td>seat territorial-jurisdiction</td>
<td>Madrid Spain</td>
</tr>
</tbody>
</table>
Ontological modelling - Relations

Relations and their properties

- **Transitive**
  \[ P(x, y) \land P(y, z) \iff P(x, z) \]
  - \( \text{part-of}(a, b) \land \text{part-of}(b, c) \iff \text{part-of}(a, c) \)

- **Symmetric**
  \[ P(x, y) \iff P(y, x) \]
  - \( \text{connected-to}(a, b) \iff \text{connected-to}(b, a) \)

- **Reflexive**
  \[ P(x, x) \]

- **Irreflexive**
  \[ \neg P(x, x) \]

- **Functional**
  \[ P(x, y) \land P(x, z) \iff y = z \]
  - \( \text{hasMother}(a, b) \iff \text{motherOf}(b, a) \)

- **Inverse**
  \[ P(x, y) \iff Q(y, x) \]

Method 1: Methontology

- A structured methodology based around intermediate representations
  1. Glossary of terms
  2. Concept taxonomy
  3. Binary relations diagram
  4. Concept dictionary
  5. Binary relations table
  6. Instance attributes table
  7. Class attributes table
  8. Constants table
  9. Instances table
  10. Formula table
  11. Axioms table

Method 2: Protégé

- **Protégé**
  - Established, general purpose tool for ontology building
  - Install Protégé 4.1 (beta)
    - Download installer for linux
    - [dice]:sh install_protege_4.1.bin
    - [dice]: ./run.sh
    - (update the reasoners when prompted)

Protégé - Established, general purpose tool for ontology building

- Install Protégé 4.1 (beta)
  - Download installer for linux
  - [dice]:sh install_protege_4.1.bin
  - [dice]: ./run.sh
  - (update the reasoners when prompted)
Some development guidelines:
1. Determine the domain and scope of the ontology
   - Competency questions
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define classes and the class hierarchy
   - Bottom-up
   - Top-down
5. Define the properties of classes (slots=attributes or relations)
6. Define facets of slots
   - Domain and range
   - Cardinality
   - Slot value type: string, number, boolean, enumerated, instance (of Classes)
7. Create instances
8. Ensure the class hierarchy is correct
   - no cycles
   - subClassOf is transitive
9. Classes represent concepts NOT the words that denote the concepts
   - Synonyms are not different classes
   - Shrimps=Prawns
10. Too many siblings may indicate an intermediate class is needed
11. Multiple inheritance is allowed
12. Class or property value?
    - White-Wine or Wine & colour=White
13. Instance or class?
14. Singular or plural names for classes?

Logical partition:
\[ \text{type}(y, \text{ChildValue}) \Rightarrow \neg \text{type}(y, \text{AdultValue}) \]

Define Adult:
\[ \text{type}(x, \text{Adult}) :: \text{type}(x, \text{Person}) \text{ and hasAge}(x, y) \text{ and } \text{type}(y, \text{AdultValue}) \]

Content pattern:
- Precedes
- IsPrecededBy
- Ordering
- Paying
- Eating
- ToddlerValue
- InfantValue
- Value partition
- AdultValue
- ChildValue
- AgeGroupValue

Methodology
From this lecture, note:

- Methontology
  - Specification criteria
  - Conceptualisation process
    - Use of intermediate representations
    - Main steps (1-4) in some detail
    - The remainder in outline

- Protégé
  - Explore use with the example ontologies:
    family.owl cows.owl and simple-pizza.owl