My definition of an ontology

- Short version: "a representation of the shared background knowledge for a community"
- Long version: "an implementable model of the entities that need to be understood in common in order for some group of software systems and their users to function and communicate at the level required for a set of tasks"
- ... and "it doesn't make the coffee"

Just one of at least three components of a complete system

Tools and downloads

- Proteg/Alpha
  - protege.stanford.edu
- GraphViz - required for OWL-Viz
  - http://www.graphviz.org/
- Tutorial handouts and Ontologies
  - http://www.cs.northwestern.edu/~mhe/Henrie/papers/MoscowMAd07
- Preparatory material
  - If you haven't done so already, please read the introduction to OWL and Protege-OWL

Ontologies" in Information Systems

- What information systems can say and say - "Models of Meaning"
  - Mathematical theories - although usually weak ones
  - evoked at the same time as Entity-Relation and UML style modeling
- Managing Scalability / complexity - "Knowledge driven systems"
  - Housekeeping tools for expert systems
  - Organising complex collections of rules, forms, guidelines...
- Interoperability
  - The common-grounding information needed to achieve communication
  - Standards and terminology
- Communication with users
  - Document design decisions
- Testing and quality assurance
  - sufficient comments to know when it breaks
  - - but "They don't make the coffee"

By way of User Centred Design

Knowledge Representation / ontologies was a solution, not a goal

Three Resources Developed independently

- Each with
  - Model
  - Knowledge/ content
  - Metadata
  - Interfaces to the others

Solution space
- Ontologies
  - Information sources
  - Logics
  - Rules
  - Frames
  - Planners
  - Logic programming
  - Bayesian nets
  - Decision theory
  - Fuzzy sets
  - Petri nets

Problem space
- Answer questions
- Allowing in actions
- Creating forms
- Discouraging resources
- Constraint actions
- Avoid risk
- ...
Topics for today
- Motivation
- Very Brief Review of OWL - a na"ive version of "pneumonia"
- Normalisation & Why Classify?
  - Why use a comparable subset of logic?
- Modularity - doing it in layers
- Anatomy, parts and Disorders - a less na"ive version of "pneumonia"
  - Pneumonia and pneumonias
  - A disorders of the lung
- Quantities and Units - if time
- Normal, NonNormal & Pathological
  - Using negation
- Summary

Assertion:
The arrival of computable logic-based ontologies/OWL gives new opportunities to make ontologies more manageable and modular
- Let the ontology authors
- create discrete modules
- describe the links between modules
- Let the logic reasoner
- Organise the result
- Let users see the consequences of their actions
  - very few people can do logic well
  - And almost none quickly

Fundamental problems:
Enumeration doesn’t scale

The scaling problem:
The combinatorial explosion
- It keeps happening!
  - "simple" logic based solutions do not scale up
  - Conditions + axioms + modifiers + axioms + context →
    large number of terms to author
    - software (ontologies)

Combination of things to be done & time to do each thing
- Terms and forms needed → Increase exponentially
- Effort per term → Most decrease in compensate
- To give the effectiveness we want
  - Or might accept
The exploding bicycle
- 1972 ICD-9 (E826) 8
- READ-2 (T30...) 81
- READ-3 87
- 1999 ICD-10 …

Diffusing the exploding bicycle:
500 codes in pieces
- 10 things to hit...
  - Pedestrian / cycle / motorcycle / car / HGV / train / unpowered vehicle / a tree / other
- 5 roles for the injured...
  - Driving/passerger / cyclist / getting in / other
- 5 activities when injured...
  - resting / at work / sporting / at leisure / other
- 2 contexts...
  - In traffic / not in traffic

V13.24 Pedal cyclist injured in collision with two- or three-wheeled motor vehicle, unpowered vehicle, street furniture, accident, while resting, sleeping, eating or engaging in other activities

Supports Loosely coupled distributed ontology development

Integrating rather than Cross Mapping

And generate it in language

User effort cut by 75% compared with manual methods
Mostly in reduced committee meetings & arguments
Rationale for Normalisation

- Maintenance
  - Each change is exactly one place
  - No "side effects"
- Modularity
  - Each primitive must belong to exactly one module
  - If a primitive belongs to two modules, it is not modular
  - Each primitive belongs to two modules. It probably confounds two essences.
  - Therefore concentrate on the "primitive skeleton" of the domain ontology
- Parsimony
  - Requires fewer axioms
Unified ontology after classification

Module Structure

Normalisation: Criterion 1
The skeleton should consist of disjoint trees
- Every primitive concept should have exactly one primitive parent
- All multiple taxonomies result in ill-defined reasoning

Normalisation Criterion 2:
No hidden changes of meaning
- Each branch should be homogeneous and logical
- Hierarchical principle should be substitution
- Otherwise we are “going to the right”
- The criteria for differentiation should follow consistent principles in each branch
  e.g. structure XOR function XOR cause

Normalisation Criterion 3
Distinguish “Self-standing” and “Refining” Concepts
- “Qualities” vs “Everything else”
  - Self-standing concepts
    - Rayleigh Weyl & Guanrino’s “contrast”
    - parent, ski, plant, secretary, desk...
  - Refining concepts – depend on self-standing concepts
    - e.g. weather: sunny, rainy, foggy...
    - Rayleigh Weyl & Guanrino’s non-contrasts
    - Closely related to Sowa’s “flat partitions”
    - Usually thought of as Value Types by engineers
    - For us an engineering distinction...

Normalisation Criterion 3a
Self-standing primitives should be globally disjoint & open
- Primitives are atomic
  - If primitives overlap, the overlap contains implicit information
- A list of self-standing primitives can never be guaranteed complete
  - How many kinds of person? of plant? of contradicts? of belief?
  - Can’t infer – parent & “red”, “blue”, “green”, red
- Heuristic:
  - Diagnoses by exclusion of self-standing concepts should NOT be part of standard ontological reasoning

Normalisation Criterion 3b
Refining primitives should be locally disjoint & closed
- Individual values must be disjoint
  - But can be hierarchical
    - e.g. “very hot”, “moderately severe”
- Each list can be guaranteed to be complete
  - Cannot have “Parent & ‘red’, ‘blue’, ‘green’, red”
- Value types themselves need not be disjoint
  - “Being hot” is not disjoint from “being severe”
- Moving values/types in overlap is a matter of logic, e.g.
  - Restriction has state sometimtes is from (severe and hot)

Normalisation Criterion 4
Axioms
- No axiom should formalise the ontology
- No axiom should imply that a primitive is part of more than one branch of primitive skeleton
- If all primitives are disjoint, no such axioms will make that primitive untestable
- Axiom for non-composition
- Closest reason to decomposition of primitives which do not subsume each other
- If any are valid, the ontology is not normalised
A real example: 
Build a simple tree

If you want more abstractions, just add new definitions (re-use existing data)

And again – even for a quite different category

Summary: Why Normalise?

Why use a Classifier?

Now: How to do it in OWL -
A quick review

Basic OWL
(In Manchester Syntax)
Primitve and Defined classes

- Primitives
  - Named things that are described
    - Lung
    - is_contained_in SOME Chest
- Defined classes
  - Things for which we have a sufficient (and necessary) definition
    - Pneumococcal_pneumonia
    - Prevalent_RTD
    - It means that is a pneumococcal and it is caused by Pneumococcal
  - And (1)Bld (that is a pneumococcal and it is caused by Pneumococcal)

Domain and Range constraints

- Domain and range constraints are really disguised universals
- has_part DOMAIN Anatomical_structure
- RANGE Anatomical_structure
- Thing is_part_of ONLY Anatomical_structure
- BEWARE: Domain and Range constraints can effect classification

Lack of Unique Name Assumption

- In OWL, anything may be the same until we say it is disjoint or different
- For classes
  - Micro-organism
  - Bacterium
  - Virus
  - (Bacterial) Bacterium, Virus
- Otherwise may overlap
- For individuals
  - a(staff wants) female

Open World Reasoning

- Everything is true unless it can be proved false
- False means "probably false"
- (Open World Assumption)
  - All ODL relations and descriptions explicitly contain "among other things"
  - Pneumococcal_pneumonia is a pneumococcal that among other things is caused by pneumococcal
- Most other systems use "negation as failure"
  - If you can't find it, it must be false
  - Closed World Assumption

The OWL Reasoner

- Organises the subclass ("subsumption") hierarchy according to the definitions (and other axioms)
- A specialized theorem prover
- AWA "classifier"
- Several available
  - FaCT++ and Pellet are built into ProtegeKappa
  - FaCT++ in Luxemburg later
  - Pellet is more casual
  - Pellet will soon have better debugging facilities

Basic steps in building an ontology: 1

Gather your terms

<table>
<thead>
<tr>
<th>Lung</th>
<th>Pneumonia</th>
<th>Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pneumonia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Micro-organism</td>
<td>Virus</td>
</tr>
<tr>
<td></td>
<td>Pneumococcal</td>
<td>Haemophilus</td>
</tr>
<tr>
<td></td>
<td>Chest</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pneumonia</td>
<td>Antibiotic</td>
</tr>
</tbody>
</table>

Basic steps in building an ontology 2:

Organise your terms

<table>
<thead>
<tr>
<th>Lung</th>
<th>Pneumonia</th>
<th>Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pneumonia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Micro-organism</td>
<td>Virus</td>
</tr>
<tr>
<td></td>
<td>Pneumococcal</td>
<td>Haemophilus</td>
</tr>
</tbody>
</table>

Basic steps in building an ontology 3

Represent your terms

- Perhaps first using mind maps or CMAPS
- Then when you understand them in OWL
- Including definitions
- And not throwing away the original source material
- ... So now on to OWL
  - First, a simplistic version of Pneumonia & Pneumonitis
Bacterial infection

- Any infection caused by some bacterium

- Note that it is a defined (equivalent) class.

Mixed infection

- How will bacterial infection, viral infection, and mixed infection classify?
  - Run the classifier and see

A "Pure bacterial infection"

- How will these classify? Is a pneumococcal and haemophylyus infection a kind of pure bacterial infection? - Both are bacteri

Closure Axioms

- How will these classify? Why?

Trivial satisfiability

- Two common errors

After classification

- Why?
After classification (Explain it)

Trivial satisfaction
- Bacterium and Virus are disjoint
  - Nothing is both a bacterium and a virus
  -marking = (bacterium AND virus)
- ONLY NOTHING = NOT SOME THING
  - Infection THAT is caused by ONLY something = Infection THAT NOT in caused by SOME Thing
- ONLY does not mean SOME:
  - Infection THAT is caused by ONLY Bacterium = Infection THAT NOT in caused by SOME NOT Bacterium
- An infection not caused by anything is a kind of infection not caused by anything except bacteria.
  - Check definition of "Pure bacterial infection"

Modularisation: towards assembling ontologies from reusable fragments

Why use modules
- Re-use
  - e.g. annotations, quantities, upper ontologies
- Coherent extensions
  - Localisation & Views
  - Local normal ranges, value sets, etc. under generic headings
  - Experimentation and add list
  - e.g. add in national examples without changing basic structure
- Logical separation
  - e.g. avoid confusing medicine and medical records
- but managing modularised ontologies is more work
  - More things to remember
  - More things to get right

Modules and imports
- Key notions:
  - "Base URI" - the identifier for the ontology
  - Is the base of a URI path usually put an UI
  - Is the start of a URI path usually put an UI
  - Physical location
  - Where the module is actually stored
  - Update your local directory for this version of the ontology
- Our conventions:
  - Ontologies stored on root of modules in a single directory
  - Start-Here.owl tells you where to load and load everything else
  - The "Active ontology" is the one you are editing
  - Active ontology items are shown in bold

Protege-OWL import mechanism
- Importer looks for a file with the correct identifying "Base URI"
- Written into the header of the XOML
  - NOT the physical location
- Order of search
  - (specified file)
  - The local directory
  - Local libraries
  - Global libraries
  - The internet at the site indicated by the Base URI

Items from active ontology are in bold

If you can, load the tutorial ontology nowontology!
- Open ...

© University of Manchester
Anatomy and Disorders
A more realistic version of Pneumonis
- Disorders have a locus in an anatomical structure of physiological process
- Disorder has locus SOME Anatomical structure
- Disorders are anything which is described as pathological
- has_normal_quality SOME Pathological
- To be explained in detail later
- Parts and wholes
- A whole field "morphology"
- Multiple views - functional / clinical's view different from structural anatomist's view

Import disorders.owl
- Go to the Active Ontology Tab
- Click the plus icon for imported ontologies
- Select import an ontology that has already been loaded
  - Import an ontology that is contained in one of the ontology libraries.
  - Import an ontology that has already been loaded.
  - Import an ontology contained in a document, located on the web.
  - Import an ontology contained in a specific file.
- Select disorders.owl and press finish

First definition of pneumonitis
- "Inflammation of the lung"
- First information
  - CTRL or CMD-F in class hierarchy
  - "Pneumonitis" is in class hierarchy
  - "Pneumonitis" is part of the lung
- Add the restriction
  - has_locus SOME Lung
- Make it a defined class

Module list
- Annotation - the annotation properties needed
- generative structures
- relations & numbers
- e.g. - the upper ontology
- e.g. - the use adnected for biorepositoriye
- Anatomy, Physiology, Biochemistry, Organism
- The root topics
- Qualities
- Functional qualities of those topics
- Disorders
- General patterns for disorders
- Specific disorders
- Examples for this tutorial
- Situations
- Disorders in context of patients and observations
Create pneumonia
▲ "Pneumonia" is a pneumonitis is the outcome of an infection.
▲ In this ontology we use has_outcome_of for "causes"
▲ "Pneumonia is a pneumonitis due to the outcome of an infection".
▲ Is outcome of some infection
▲ Information: (If you have a pneumonia, then you will have ...

Bacterial pneumonia
▲ First attempt
▲ "Pneumonia caused by a bacteria"
▲ But need to rephrase to fit the ontology
▲ "Pneumonia that is the outcome of an infection by bacteria"
▲ In this ontology, by "outcome" is the property "has_outcome_of"
▲ "Pneumonia have acts and objects"

By analogy make viral pneumonia and mixed pneumonia
▲ Mixed pneumonia is a pneumonia that is caused by both virus and pneumonia
▲ How to say this

Classify and check
▲ Be sure that all classes are defined
▲ ▲ defined
▲ ▲ primitive
▲ To convert from primitive to defined, cmnd-d or ctrl-d (Mac or PC)

Should get
▲ "Pneumonia"
▲ "Bacterial pneumonia"
▲ "Viral pneumonia"
▲ "Pneumonitis"

What about "left lower lobe pneumonia"?
▲ First define lobar pneumonia as
▲ "Pneumonia that has locus in a lobe of a lung"
▲ "Lobe that is a subdivision of SOME Lung"
▲ But what if I define an inflammation of a lobe of the lung
▲ "Inflammation: THAT has locus SOME (Lobe THAT is a subdivision of SOME Lung)"
▲ The classifier might make it for us

OWL means what it says
▲ Lobes are not lungs!
▲ Our definition of lung disorder is too narrow
▲ Almost done
▲ Disorders of parts are disorders of the whole
▲ A broader definition of "Disorder of Lung"
▲ Disorder THAT has locus SOME (Lung OR is a clinical part of SOME Lung)
▲ Almost OK, but still inflammation of lobe of lung is not a pneumonitis

Make the pattern consistent
▲ Redefine Pneumonitis
▲ "An inflammation of the lung or any clinical part of the lung of the lung"
Almost correct, but...

- What about "Bronchitis"?
  - An inflammation of the bronchi (or any of their parts)
  - Try a filter.
  - information
  - Bronchi, bronchitis, disease, symptoms, etc.

Definition of "Pneumonitis" is now too broad
- Not just any part of the lung, but the "parenchyma" of the lung
- lobe, quadrants, bases, apices, etc.

Now reclassify

- Bronchitis is now a disease of the lung ("lung disease") but not a pneumonia
- As required.

Also affects modularity

- We have chosen to model functional parts with physiology rather than with anatomy.
- To link with the FHM view so as to preserve in the anatomy module.
- So we add the fact that the pleura is a functional part of the lung in the physiology processes module rather than the anatomy module.
- Might even have a separate functional module:
  - A set of descriptions for a new module
  - Add to model of physiology processes module.

Normality and Negation

- What does it mean to be normal or abnormal?
- We implement two notions:
  - Normal - nothing noteworthy
  - Abnormal - requiring medical intervention
- Producing "yes" and "no" answers for each question/
- "Abnormal" is not "physically abnormal" but not normal in CEBL

The property hierarchy allows multiple views

- The bronchus is a "component" of the lung
- The lobe is a subdivision of the lung
- Redefine pneumonia as an inflammation of the lung or a subdivision of the lung

Clinical partonomy and pleuritis

- To an anatomist, the pleura are different organs from the lungs.
- To a clinician, pleuritis should be classified as a "lung disease" or "disorder of the Pleura" - inflammation of the pleura.
- The Pleura:
  - Section as part of the lung
  - Even though they are not physically part of the lung
  - The property hierarchy expects both views.
  - Anything that is structurally a part of something is a clinical part of it.
  - Anything that functionally a part of something is a clinical part of it.

Create pleuritis and classify

- Classify and check results:
  - A disease of the lung but not a bronchitis or pneumonitis.
  - Anatomists and clinicians use each base their own terms:
    - Bronchitis
    - Pneumonitis

Normality and negation

- Basic rules:
  - Pathological = normal
  - Normal = NOT pathological
  - Pathological = NOT pathological
- Assumptions:
  - Abnormal means "necessarily abnormal"
  - Normal = Pathological = NOT normal pathological
  - See the definitions of non-normal quality in disorders and...
Defining “disease” or “disorder”

- Hard, probably futile
- The words are used in many different ways
- Things referred to cross ontological boundaries
- Processes - e.g. infection or inflammation
- Qualities - e.g. deformity, mathematical, situation...
- Best just to say what is pathological
- let the classifier gather them up
- Also classify along multiple dimensions
- include as many abstractions as are useful, no more and no less

Example from tiny tutorial ontology

Defining “disease” or “disorder”

- Hard, probably futile
- The words are used in many different ways
- Things referred to cross ontological boundaries
- Processes - e.g. infection or inflammation
- Qualities - e.g. deformity, mathematical, situation...
- Best just to say what is pathological
- let the classifier gather them up
- Also classify along multiple dimensions
- include as many abstractions as are useful, no more and no less

Example:

Head Trauma with/intracranial bleeding with/without Skull Fracture

Situations & Codes

- The package unit of information in electronic health records is a “Situation”
- A patient as observed by a clinician at a time in a setting
- Adding a code to a event in a record is to assert that the event is an instance of that kind of situation
- The event has at least all of the assertions comprising the individual codes.
- Allows easy expression of negation and classification of results with recognition of equivalence

Quantities and Units

- A pervasive issue, so we shall take a brief look now
- Make data-entities.owl the active ontology

Quantities

- As real as numbers, matrices, or any other mathematical structure
- “Real” numbers usually used for healthcare (T)
- “Tempered” numbers have been and continue to be used
- “Tempered” numbers need not have units (e.g. percentages, probabilities, etc.)
- Quantitative Free

Example:

Head Trauma with/intracranial bleeding with/without Skull Fracture

Classification - Note automatic inversion of negatives

Note automatic inversion of negatives
Simple structure in tutorial ontology

- Dimension implicit in classification of quantities
- Unit as an object
- Classifier: values of units
- Domain and range constraints
- Making disjoint explicit

Typical quantities

- Specific value
  - Concentration, quantity THAT
    has_units SOME mg_per_l
    has_magnitude VALUE 145
- Value range (OWL 1.1 / new version only)
  - Concentration, quantity THAT
    has_units SOME mg_per_l
    has_magnitude SOME Q (>=13, <=15)
- NB units mg_per_l will cause quantity to be classified as a MgAs_concentration_quantity
- Try it in DL query text!

Example of use

"Hemoglobin 13 mg/l"
- Secrox, hemoglobin THAT
  has_unit SGOM
  Concentration, quantity THAT
  has_magnitude VALUE 13
  has_units SOME mg_per_l

Summary: Building Ontologies in OWL-DL

- Start with a taxonomy of primitive classes
  - Should form pure trees
  - Remember, to make disjointness explicit
- Use definitions and the classifier to create multiple hierarchies
  - Use existential (some/related/to) restrictions by default
  - Things will only be classified under defined classes
- Be careful with
  - Open world reasoning
  - Use classes, entities when needed
    - "same" and "only" - some/related/to/other/related/to
    - domain and range constraints
    - making disjoint explicit

Extensions to quantities

- Compatible java packages for units and conversion
  - Really explicit to disappear into datatypes
  - Just a www.url/xyz...
<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>► Knowledge is fractal</td>
</tr>
<tr>
<td>► Enumeration is never ending</td>
</tr>
<tr>
<td>► The closure of UML / SWL in composition and classification</td>
</tr>
<tr>
<td>► Normalise ontologies for re-use and maintenance</td>
</tr>
<tr>
<td>► Build DAML nets out of Trees using classification</td>
</tr>
<tr>
<td>► Use Modules to separate views and then link them</td>
</tr>
<tr>
<td>► Diseases of the parts are diseases of the whole</td>
</tr>
<tr>
<td>► but must be careful</td>
</tr>
<tr>
<td>► The property hierarchy can be used to support multiple views</td>
</tr>
<tr>
<td>► Some notions defy definition - e.g. &quot;Disease&quot;</td>
</tr>
<tr>
<td>► When in doubt describe, classify and collect until bottom up</td>
</tr>
<tr>
<td>► Much more in the comments in the tutorial ontology</td>
</tr>
</tbody>
</table>

► and remember: "Ontologies are just one part of a system."