

UNIVERSITY OF
Southampton

School of Electronics
and Computer Science

Ontology Engineering

Topics

- This lecture:
 - Ontologies – a quick recap
 - What are they?
 - What type of ontologies are out there?
 - What are they used for?
 - Ontology Building Methodologies
 - Life-cycle
 - Existing methodologies
- Next lecture:
 - Design Patterns

Onto

Machine
readable

ontology should represent
a shared view of the domain

- Definition

- “a formal, explicit specification of a shared conceptualisation”

Gruber

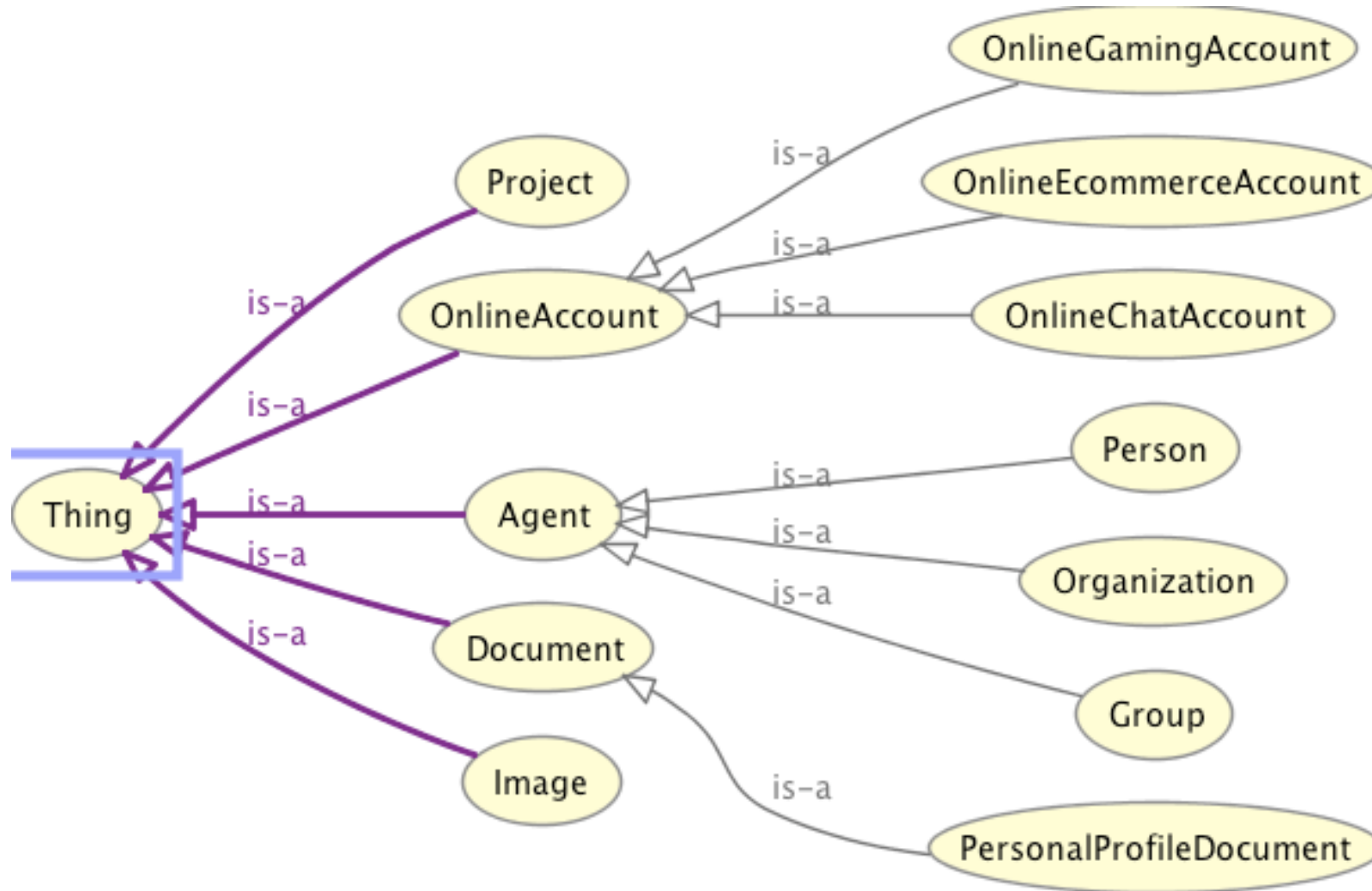
Representation of concepts
and constraints is
explicitly defined

modelling the concepts
and relations of the domain

- And in English ...

- An ontology is the combination of concepts and relationships required to model a knowledge domain in a human *and* machine understandable format

Example: FOAF Ontology



When to Use an Ontology?

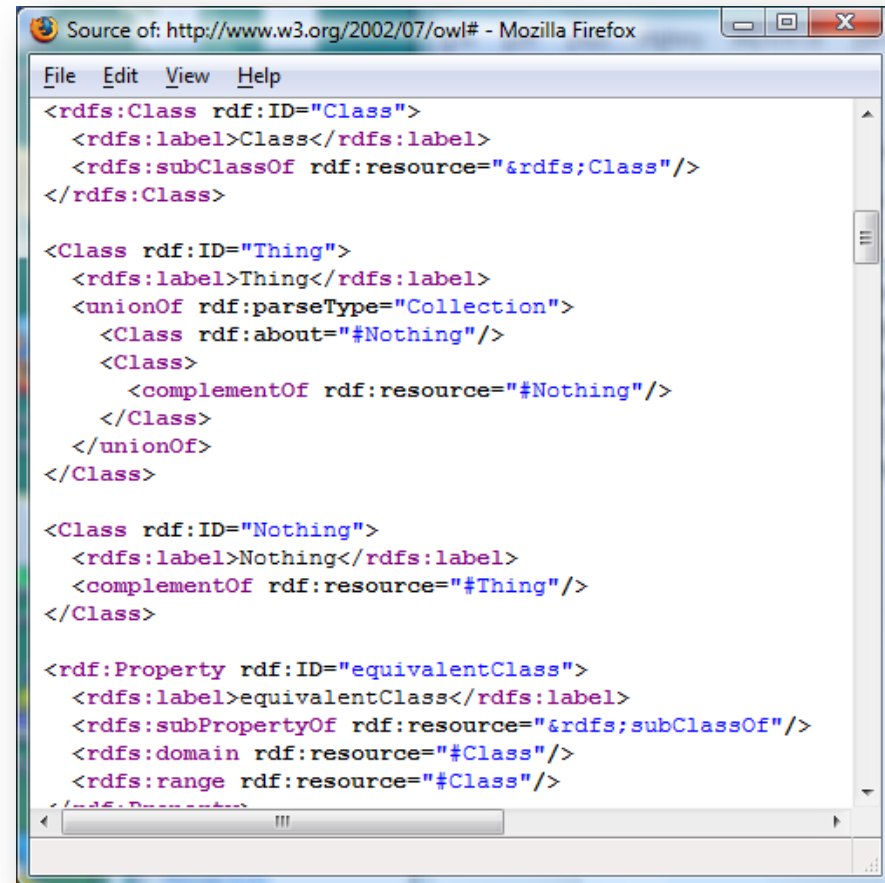
- Knowledge management
 - Control vocabulary
 - Making domain assumptions more explicit
 - Separate the metadata structure from the data itself
 - Change in metadata does not necessarily require change in the data
- Knowledge sharing
 - The clear model of your data enables other machines and people to understand it, and thus use and reuse it
- Knowledge integration
 - Ontologies can bridge between several data sources
- Knowledge analysis
 - Using a rich data model enables more complex analysis to be made on the data (eg for knowledge discovery)

Type of Ontologies

- There are four main type of ontologies:
 - Representation ontologies
 - General or upper-level ontologies
 - Domain ontologies
 - Application ontologies

Representation ontologies

- Describe low level primitive representations
 - Such as semantic web languages
- Example ontologies:
 - OWL, RDF, RDFS
- Usual size: small, a few dozens of concepts and relations



```
Source of: http://www.w3.org/2002/07/owl# - Mozilla Firefox
File Edit View Help
<rdfs:Class rdf:ID="Class">
  <rdfs:label>Class</rdfs:label>
  <rdfs:subClassOf rdf:resource="#rdfs;Class"/>
</rdfs:Class>

<Class rdf:ID="Thing">
  <rdfs:label>Thing</rdfs:label>
  <unionOf rdf:parseType="Collection">
    <Class rdf:about="#Nothing"/>
    <Class>
      <complementOf rdf:resource="#Nothing"/>
    </Class>
  </unionOf>
</Class>

<Class rdf:ID="Nothing">
  <rdfs:label>Nothing</rdfs:label>
  <complementOf rdf:resource="#Thing"/>
</Class>

<rdf:Property rdf:ID="equivalentClass">
  <rdfs:label>equivalentClass</rdfs:label>
  <rdfs:subPropertyOf rdf:resource="#rdfs;subClassOf"/>
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
```

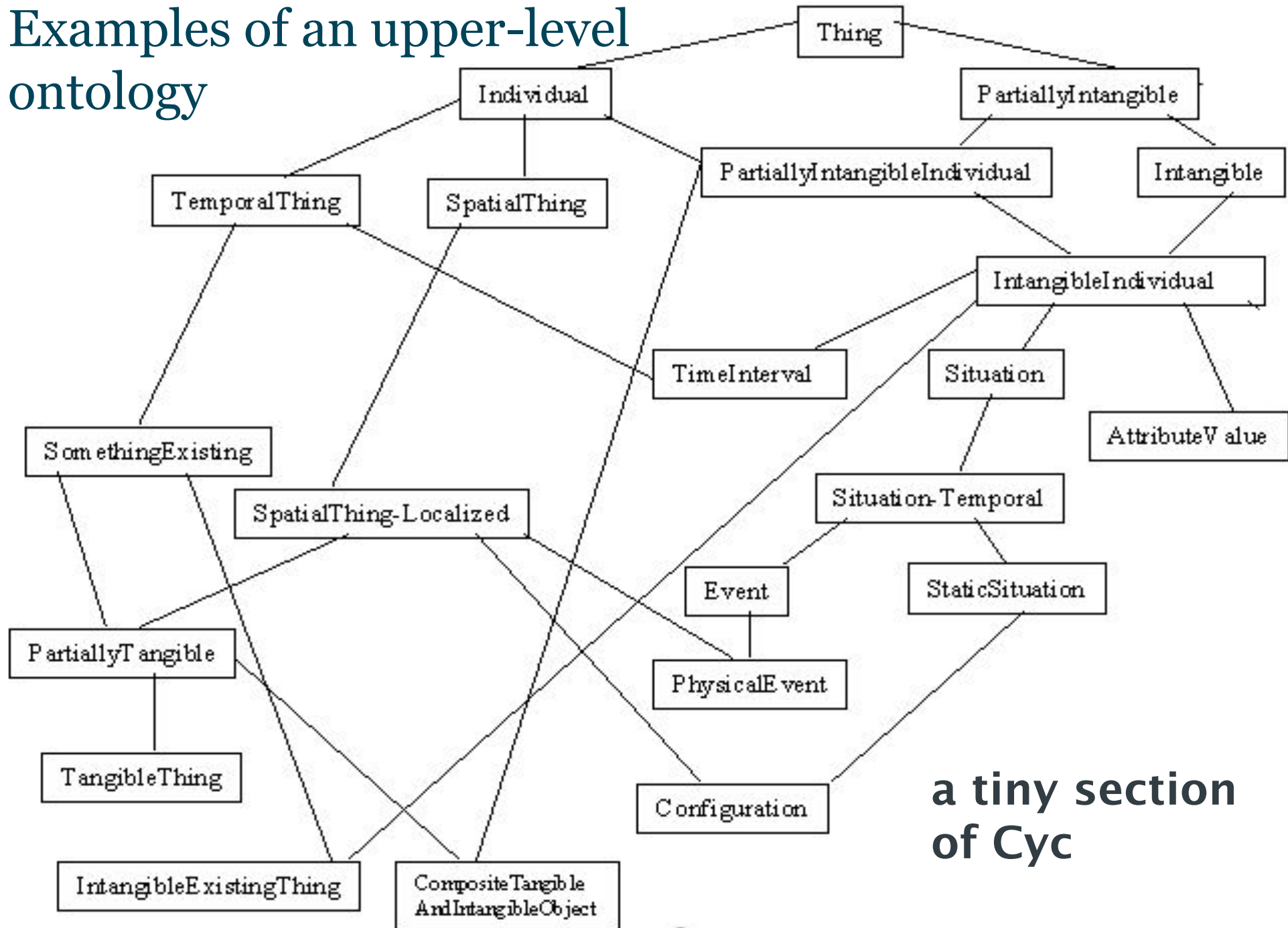
Section of the OWL ontology

General or upper-level ontologies

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

- Describe high-level, abstract, concepts
- Usually domain independent
 - Can be used as part of other ontologies
- Example ontologies:
 - Cyc: commonsense ontology
 - Hundreds of thousands of concepts
 - WordNet: English lexicon
 - Over 150K concepts
 - SUMO: Suggested Upper Merged Ontology
 - Around 10K concepts
- Tend to come in large sizes

Examples of an upper-level ontology



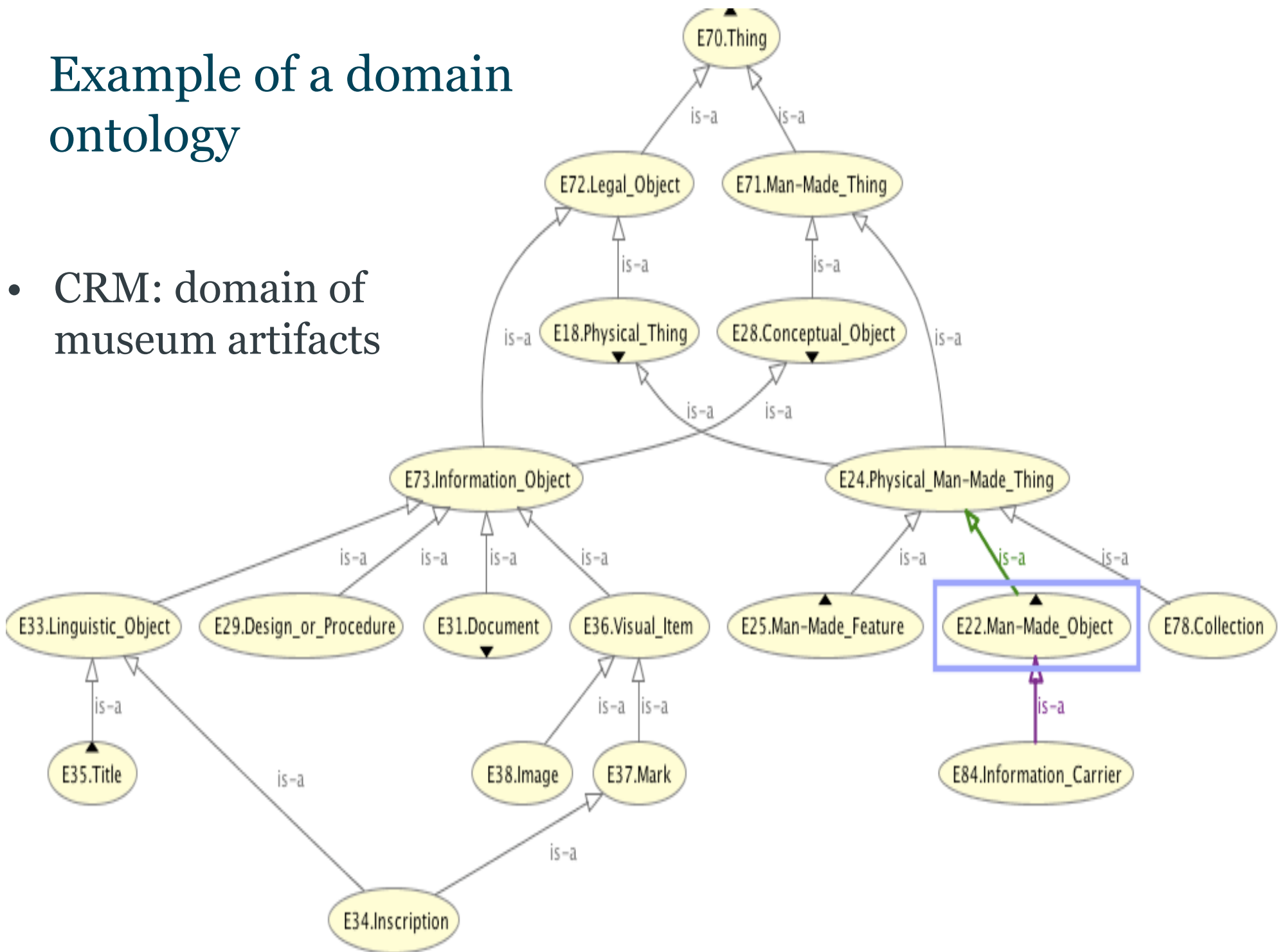
**a tiny section
of Cyc**

Domain ontologies

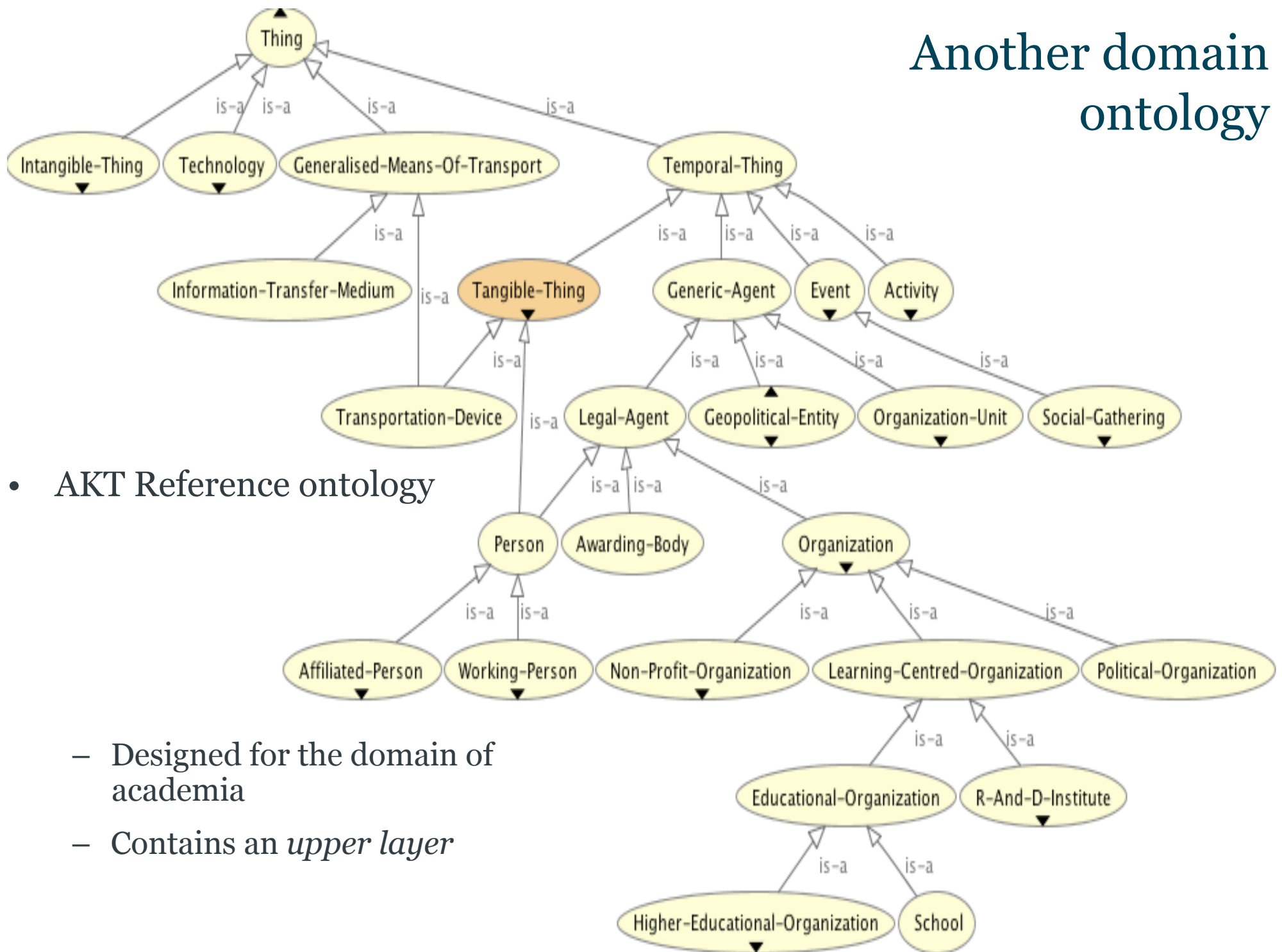
- Describe a particular domain extensively
- Domain dependent by definition
- Example ontologies:
 - GO: Gene Ontology
 - Roughly 25K concepts
 - CIDOC CRM: for cultural heritage
 - Roughly 100 concepts
 - FMA: Foundational Model of Anatomy
 - Around 75K concepts

Example of a domain ontology

- CRM: domain of museum artifacts



Another domain ontology



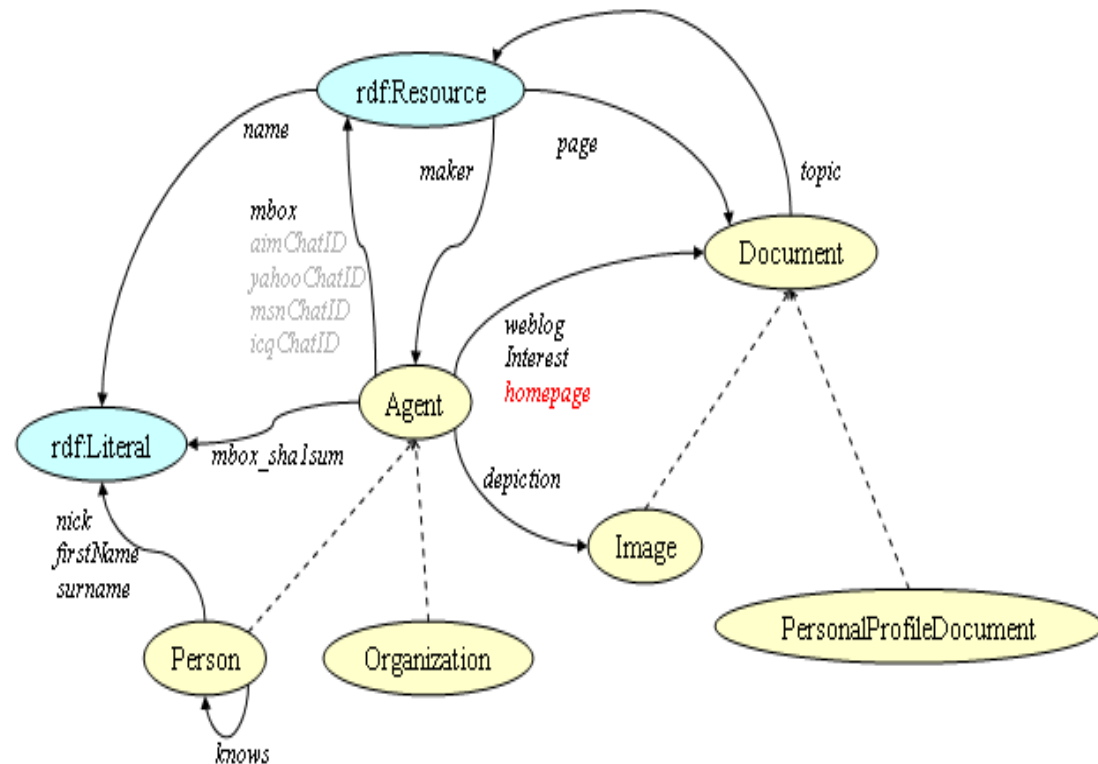
- AKT Reference ontology

- Designed for the domain of academia
- Contains an *upper layer*

Application ontologies

A Simple FOAF ontology (v0.9)

- Mainly designed to answer to the needs of an application
- Scaled and focused to fit the application domain requirements
- Example ontologies:
 - FOAF: Friend of a Friend ontology
 - ~ a dozen concepts
 - ESWCo6: for conference metadata
 - ~ 80 concepts, including FOAF



UNIVERSITY OF
Southampton

School of Electronics
and Computer Science

Ontology Building Methodologies

- No standard methodology for ontology construction
- There exists a number of methodologies and best practices
- The following life cycle stages are usually shared by the methodologies:
 - Specification - *scope and purpose*
 - Conceptualisation - *determining the concepts and relations*
 - Formalisation - *axioms, restrictions*
 - Implementation - *using some ontology editing tool*
 - Evaluation - *measure how well you did*
 - Documentation - *document what you did*

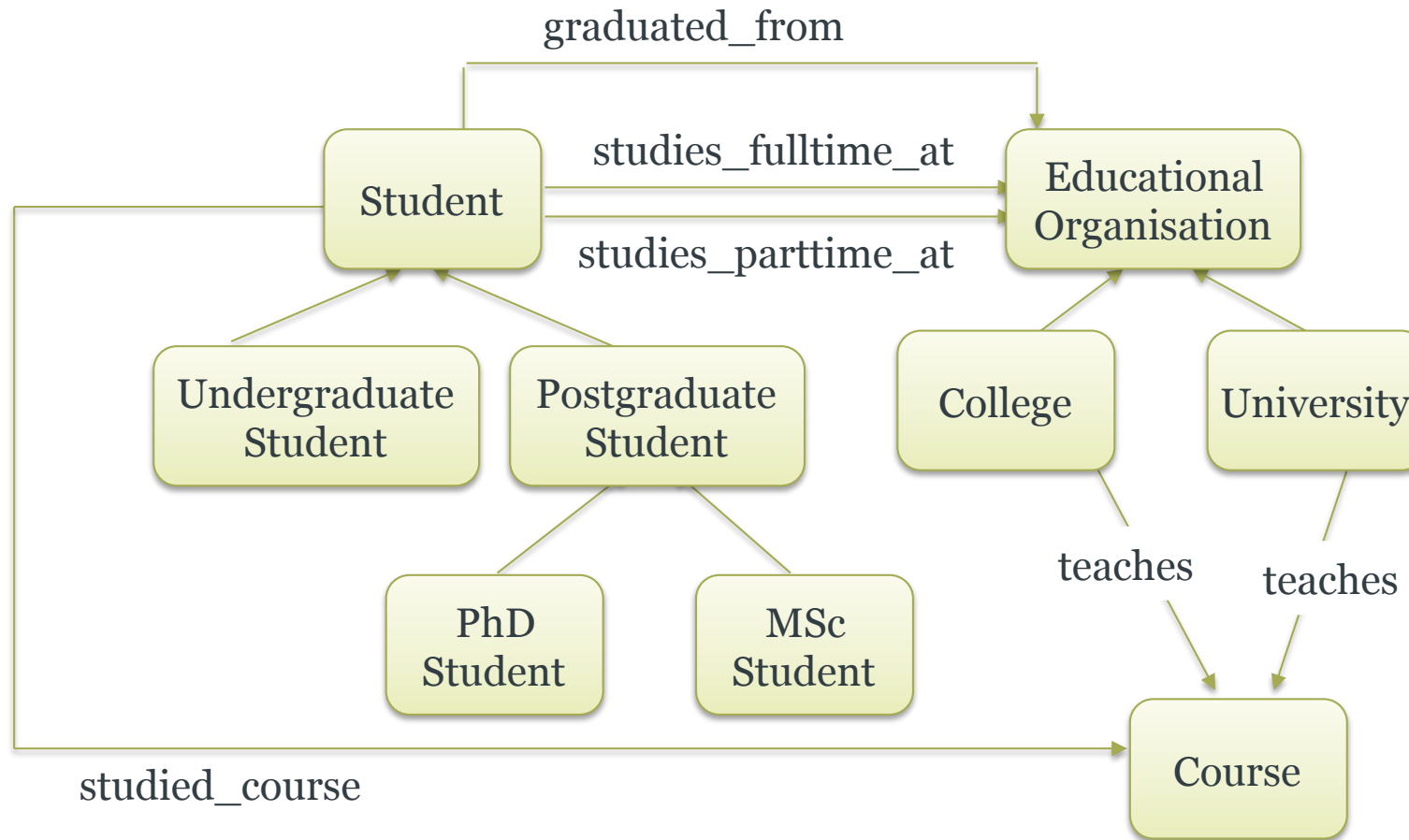
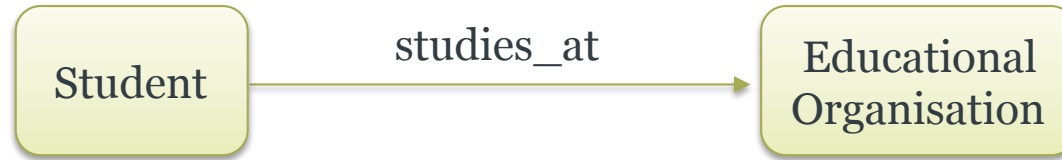
Ontology Development Life-Cycle

- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Specification

- Specifying the ontology's purpose and scope
 - Why building this ontology?
 - What will this ontology be used for?
 - What is the domain of interest?
 - *An ontology for car sales probably don't need to know much about types and prices of engine oil*
 - How much detail do you need?

Limit Your Scope



Competency Questions

- What are **some of** the questions you need the ontology to answer (competency questions)?
 - Make a list of such questions and use as a check list when designing the ontology
 - They help to define scope, level of detail, evaluation, etc.

- *The questions that you REALLY need*
 - *You probably don't need to worry about the questions that "perhaps someone might need to ask someday"*

- *The questions that CAN BE answered*
 - *i.e. you have/can get the necessary data to answer those questions*
 - *Permanent lack of some data may render parts of the ontology useless!*

Ontology Development Life-Cycle

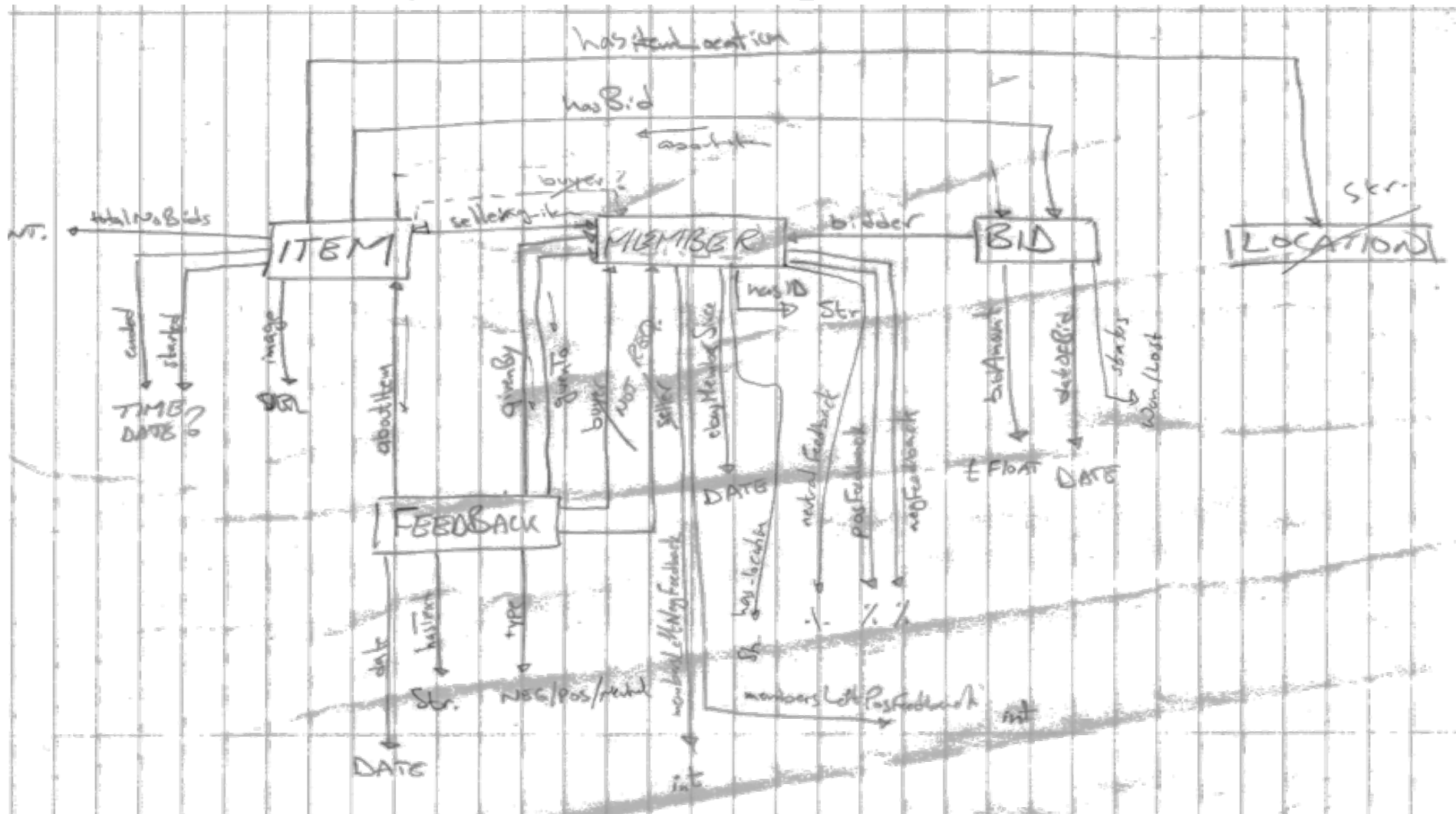
- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Conceptualisation

- Identifying the concepts to include in your ontology, and how they relate to each other
 - This will be dependent to some extent on your defined scope and competency questions
 - Set unambiguous names and descriptions for those classes and relationships (more on this in Documentation)
 - Reach agreement

Conceptualisation

- When designing an ontology, start with a drawing sheet or software (eg Visio, Mind Maps), or cards



Conceptualisation

- Ontologies are meant to be *reusable!*
 - *Technology for reusing ontologies is still limited*
- Always a good idea to check any existing models or ontologies
 - Check your database models or off-the-shelf ontologies
- Check existing ontologies
 - No need to reinvent the wheel, unless it is easier to do so!
 - Ontology search engines
 - Swoogle, Watson

ProtegeWiki: Protege Ontologies Library - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search

Address <http://protege.cim3.nyu.edu/>

Google

Protege

WikiHomePage | Recent Page Index

Please make insertions in order. Thank you!!

- [Biological Process](#): A model of biological functions that is comprehensive and interpretable, (30H)
- [BioPAX](#): An Ontology of pathways, it is an exchange data resources.

File Edit View Favorites Tools Help

Back Forward Stop Home Search

Address <http://swoogle.umbc.edu/index.html>

Google swoogle

Swoogle

semantic web search 2006

list ontologies matching ontology search

<http://www.cyc.com/2004/>
PureRDFDocument, RDFXML, metadata, cached

<http://139.91.183.30:9090/>
[DEFINES] | SewerCost | Conference LandTrust | Granite | Highest [USES] | Drying | Sewer Cost | Sewer Land Trust | LandTrust
PureRDFDocument, RDFXML, metadata, cached

<http://www.esd.org.uk/standards/Igci/1.03/Igci-schema>

[DEFINES] | CalibrationServices | Student Loans | Student Loans | Prolysis |

show document's metadata

<http://www.aktors.org/ontology/portal> document

View it using ... [W3C RDF validator](#) [Pellet](#) [Hyperdaml](#)

Swoogle's Metadata

- [datePing](#): 2006-01-17
- [pingState](#): PingAliveModified
- [type](#): PureRDFDocument
- [parseState](#): ParseSuccess
- [rdfSyntax](#): RDFXML
- [dateLastmodified](#): 2004-12-10
- [dateCache](#): 2006-01-17
- [docCharset](#): ISO-8859-1
- [docTriples](#): 1,480.00
- [docLength](#): 89K
- [ontoRatio](#): 0.85

Cached Metadata

[sparql query] SELECT ?predicate, ?object WHERE { <
<http://www.aktors.org/ontology/portal> > ?predicate ?object }

NO	frequency	?predicate	?object
1	1	owl:versionInfo	2.0
2	1	rdfs:label	AKT Reference Ontology (Portal Ontology)

Ontology Development Life-Cycle

- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Formalisation

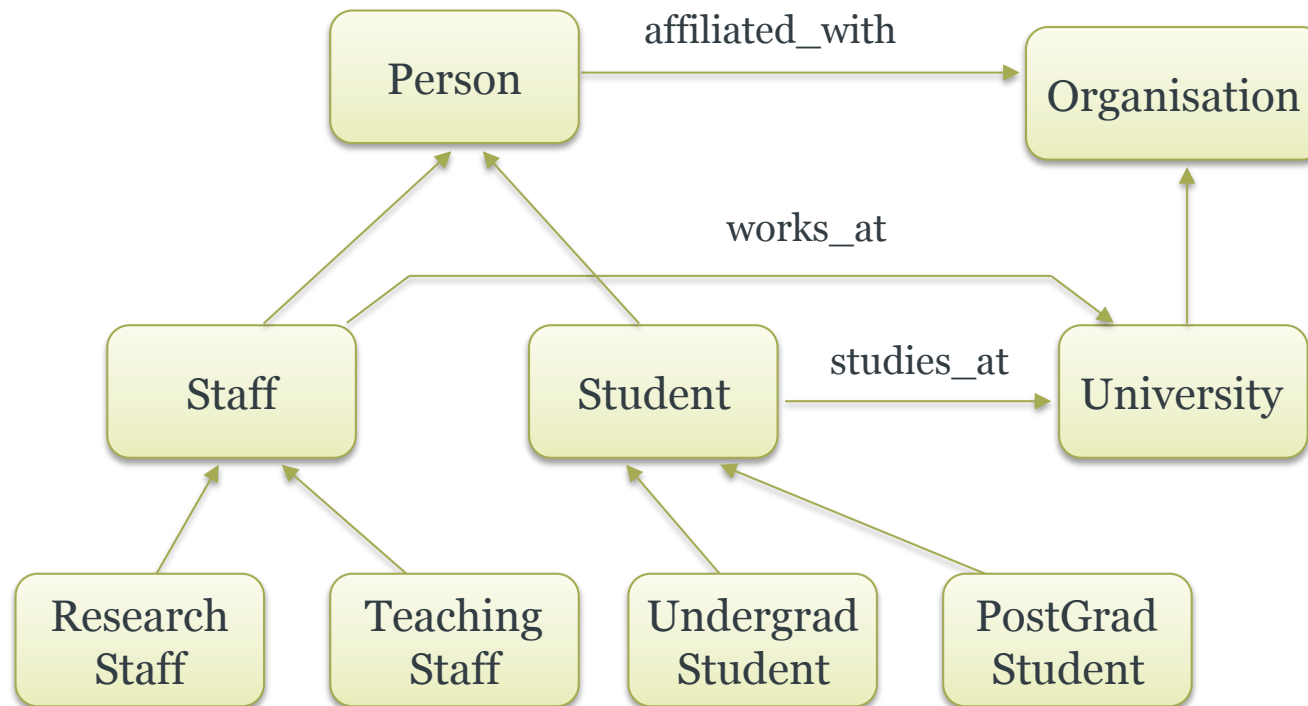
- Moving from a list of concepts to a formal model
- Define the hierarchy of concepts and relations
- Also note down any restrictions
 - E.g. NonProfitOrg isDisjoint from ProfitOrg
 - An email address is unique

Building the Class Hierarchy

- Several approaches have been suggested to construct the class hierarchy:
 - Top-down
 - Start with the most general classes and finish with the most detailed classes
 - Bottom-up
 - Start with the most detailed classes and finish with the most general ones
 - Middle-out
 - Start with the most obvious classes
 - Group as required
 - Then go upwards and downwards to the more general and more detailed classes respectively

Middle-Out Approach

- Good for controlling scope and detail

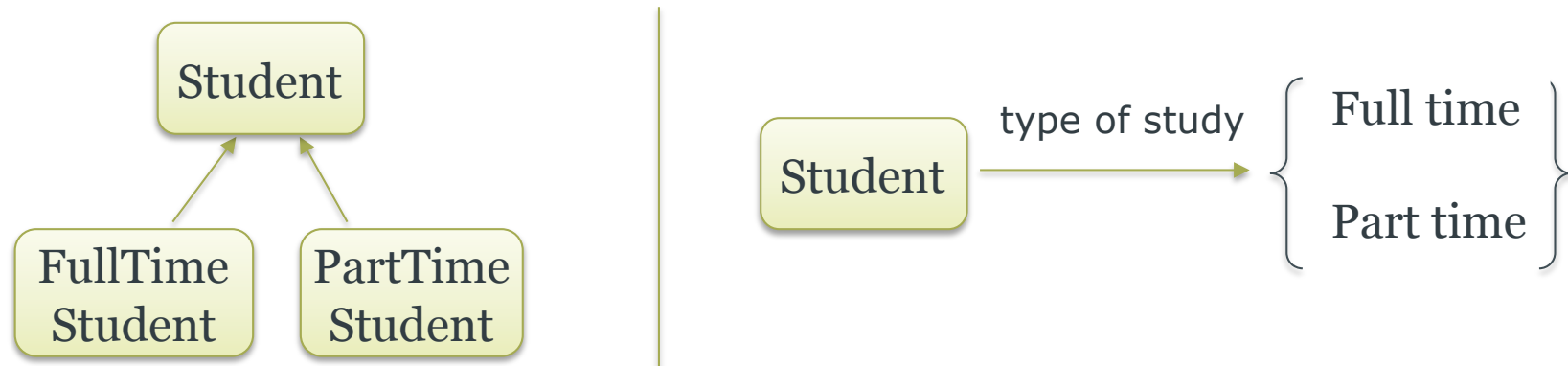


Naming Conventions

- Not rules, but conventions
- Avoid spaces and uncommon delimiters in class and relation names
 - E.g. use PetFood or Pet_Food instead of Pet Food or Pet*Food
- Capitalise each word in a class name
 - E.g. PetFood instead of Petfood or even petfood
- Names of relations usually start with a lowercase
 - E.g. pet_owner, petOwner
- Better use singular names
 - E.g. Pet, Person, Shop

Class or a Relation?

- Is it a class or a relation?



- Answer:
 - It depends!
 - If the subclass doesn't need any new relations (or restrictions), then consider making it a relation

Class or an Instance?

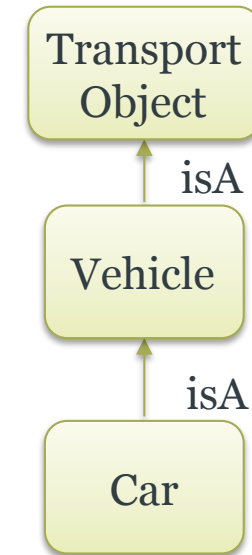
- Is it a class or an instance?



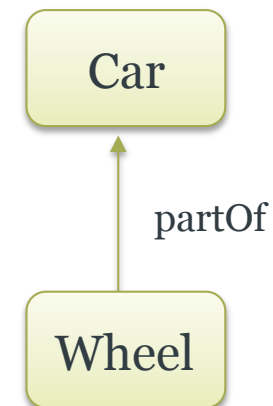
- Answer:
 - If it can have its own instances, then it should be a class
 - If it can have its own subclasses, then it should be a class

Transitivity of Class Hierarchy

- isA relation is always transitive
 - Car is a Transportation Object
 - Any instance of Car is also a TransportationObject

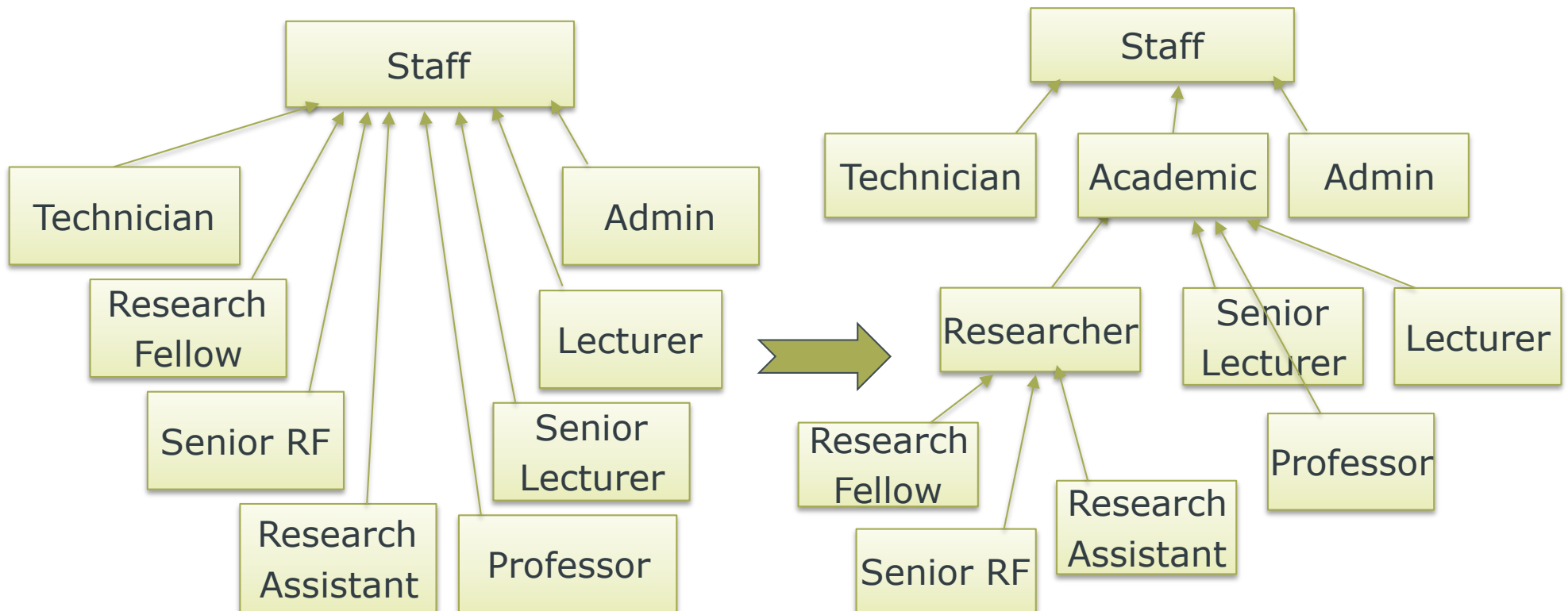


- isA is not the same as partOf



Tidy Your Hierarchy

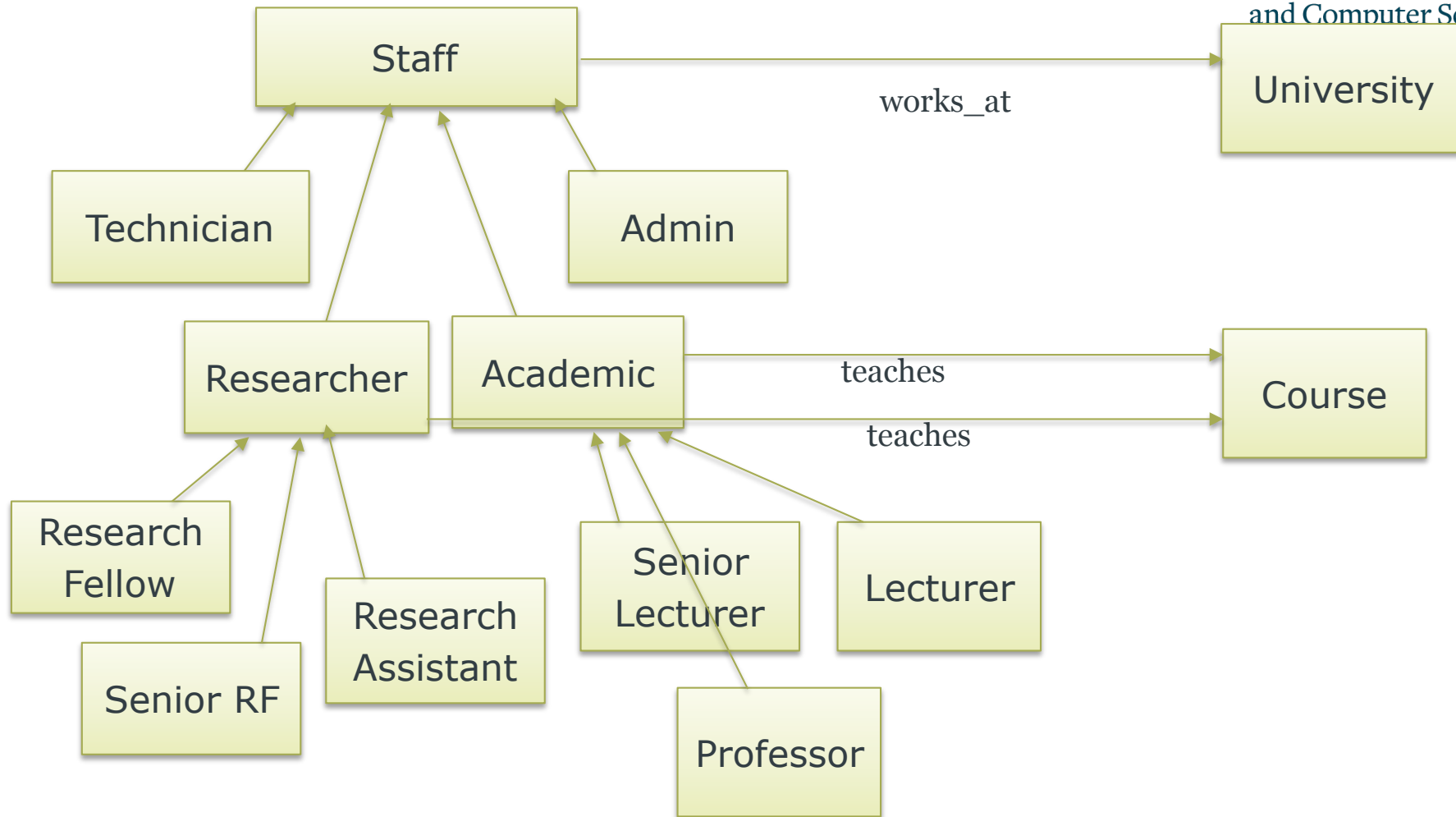
- Avoid isA clutter!
- Break down your hierarchy further if you have too many direct subclasses of a class



Where to Point my Relation?

- Relations should point to the most general class
 - But not too general
 - E.g relations pointing to Thing!
 - And not too specific
 - E.g. relations pointing to the bottom of the hierarchy

Where to Point my Relation?



Ontology Development Life-Cycle

- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Implementation

- Choose a language
 - e.g. RDFS, OWL Lite, OWL, DL
- Implement it with an ontology editor
 - e.g. Protégé, SWOOP, TopQuadrant
- Edit the class hierarchy
- Add relationships
- Add restrictions
- Select appropriate value types, cardinality, etc
- Apply a reasoner to check your ontology
 - e.g. Racer, Pellet, Fact++

Ontology Development Life-Cycle

- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Evaluation

- Implementing the ontology in an ontology editor helps to get the syntax correct
- You can also validate your OWL ontology online using:
 - The WonderWeb OWL validator
 - <http://www.mygrid.org.uk/OWL/Validator>
 - W3C RDF validator
 - <http://www.w3.org/RDF/Validator/>
- Check the ontology against your competency questions
 - Write the questions in SPARQL or in similar query languages
 - Can you get the answers you need?
 - Is it quick enough?
 - Perhaps you can add additional properties or restructure the ontology to increase efficiency

Ontology Development Life-Cycle

- Specification
- Conceptualisation
- Formalisation
- Implementation
- Evaluation
- Documentation

Documentation

- Documenting the design and implementation rational is crucial for future usability and understanding of the ontology
 - Rational, design options, assumptions, decisions, examples, etc.
- Skuce (1995) proposes a format for documenting ontological assumptions
 - Layer 1: Class and relation assumptions
 - Conceptual assumptions (C)
 - Terminological assumptions (T)
 - Definitional assumption (D)
 - Examples (E)

Documentation

The screenshot shows the Protege-2000 interface with the 'base:Publication' class selected. The left pane shows a class hierarchy starting from ':THING'. The right pane shows the class definition for 'base:Publication' with the following fields:

- Name:** base:Publication
- Role:** Abstract
- Documentation:** C: A published body of work. This differs from a document by nature of having been subject to some editorial process and/or sponsorship by an organisation (the publisher).
T: Publication
D: A published work, typically also edited.
E: The journal "Artificial Intelligence". A volume of LNCS. A book of conference proceedings.
- Template Slots:** base:URL

The 'Documentation' field is highlighted with a red border.

Ontology Engineering Methodologies

- TOVE (Gruninger and Fox 1995)
 - Focus on competency questions
- ENTERPRISE (Uschold and King 1995)
 - Focus on conceptualisation
- METHONTOLOGY (Fernández and Gómez-Pérez 1997)
 - Inspired by IEEE Standard software development life cycle
 - Stresses the iterative process and evolution of prototypes

TOVE

- Capture motivating scenarios
 - E.g. what's the application
- Formulate competency questions and desired answers
 - What questions the ontology needs to answer to satisfy the scenarios
- Specify and formalise the terminology
- Formalise the competency questions
- Specify axioms and definitions
- Evaluate ontology against the competency questions

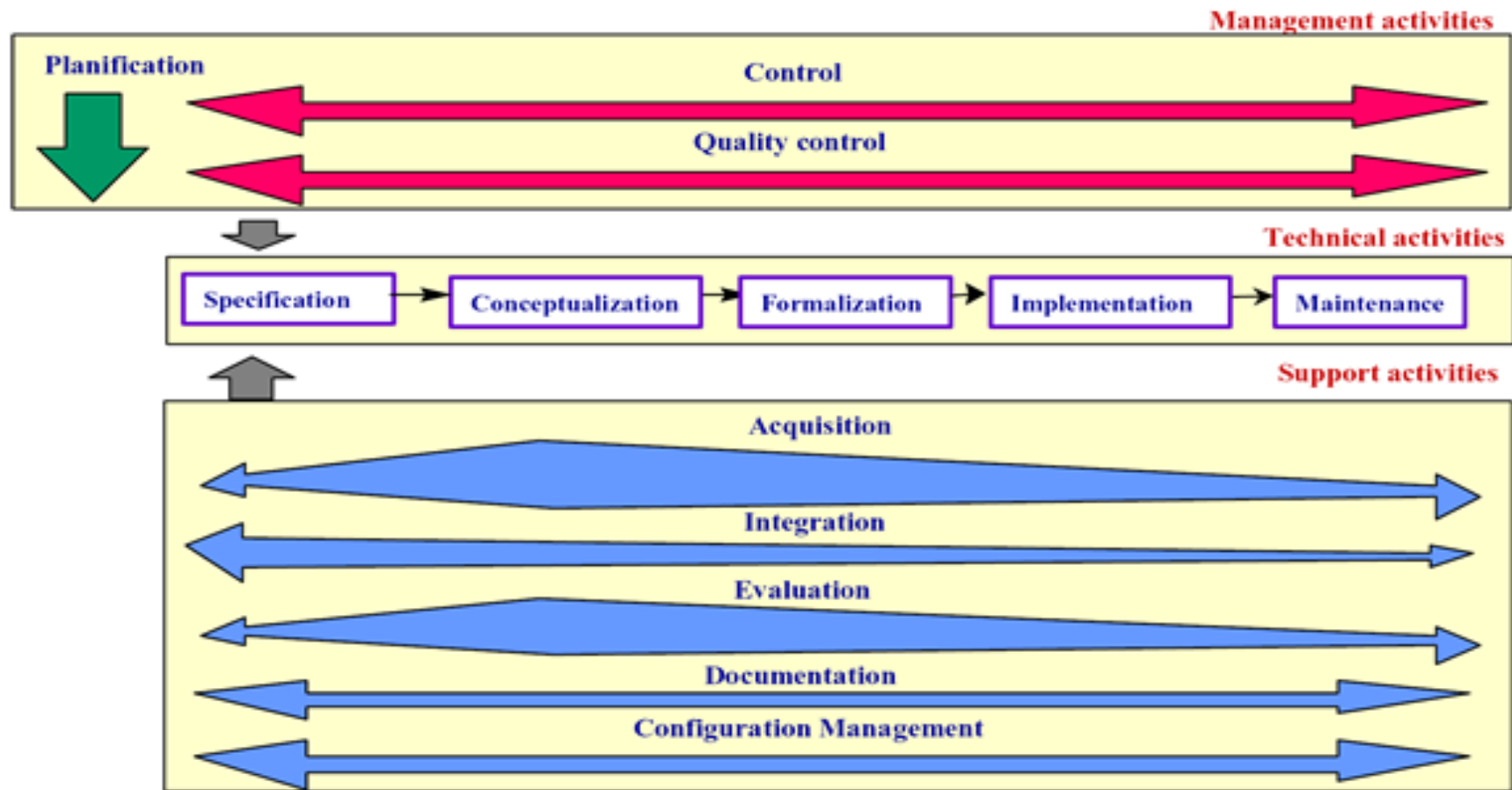
ENTERPRISE

- Identify purpose
 - Why build it, what for, who are the users
- Build the ontology
 - Ontology capture
 - Identify concepts, relations, labels using a middle-out approach
 - Ontology implementation
 - Integrate existing ontologies
- Evaluation
 - Check against the requirements, competency questions, real-world use
- Documentation

METHONTOLOGY

- Management activities
 - Scheduling, control, and quality assurance
- Development activities
 - Pre-development
 - Scenarios, feasibility study
 - Development
 - Specification, conceptualisation, formalisation, implementation
 - Post-development
 - Maintenance
- Support activities
 - Same time as development
 - Knowledge acquisition, evaluation, mapping and merging with other ontologies, documentation

METHONTOLOGY



Summary

- Ontology construction is an iterative process
 - You build the ontology, try to use it, fix errors, extend, use again, and so on.
- There is no single correct model for your domain
 - The same domain may be modelled in several ways
- Following best practices helps to build good ontologies
 - Well scoped, documented, structured
- Reuse existing ontologies if possible
 - Check your database models and existing ontologies
 - Reuse or learn from existing representations
 - Most ontology editing tools don't provide good support for reuse yet

Common Pitfalls

- Over scaling and complicating your ontology
 - Need to learn when to stop expanding the ontology
- Lack of documentation
 - For the design rational, vocabulary and structure decisions, intended use, etc.
- Redundancy
 - Increase chances of inconsistencies and maintenance cost
- Using ambiguous terminology
 - Others might misinterpret your ontology
 - Mapping to other ontologies will be more difficult