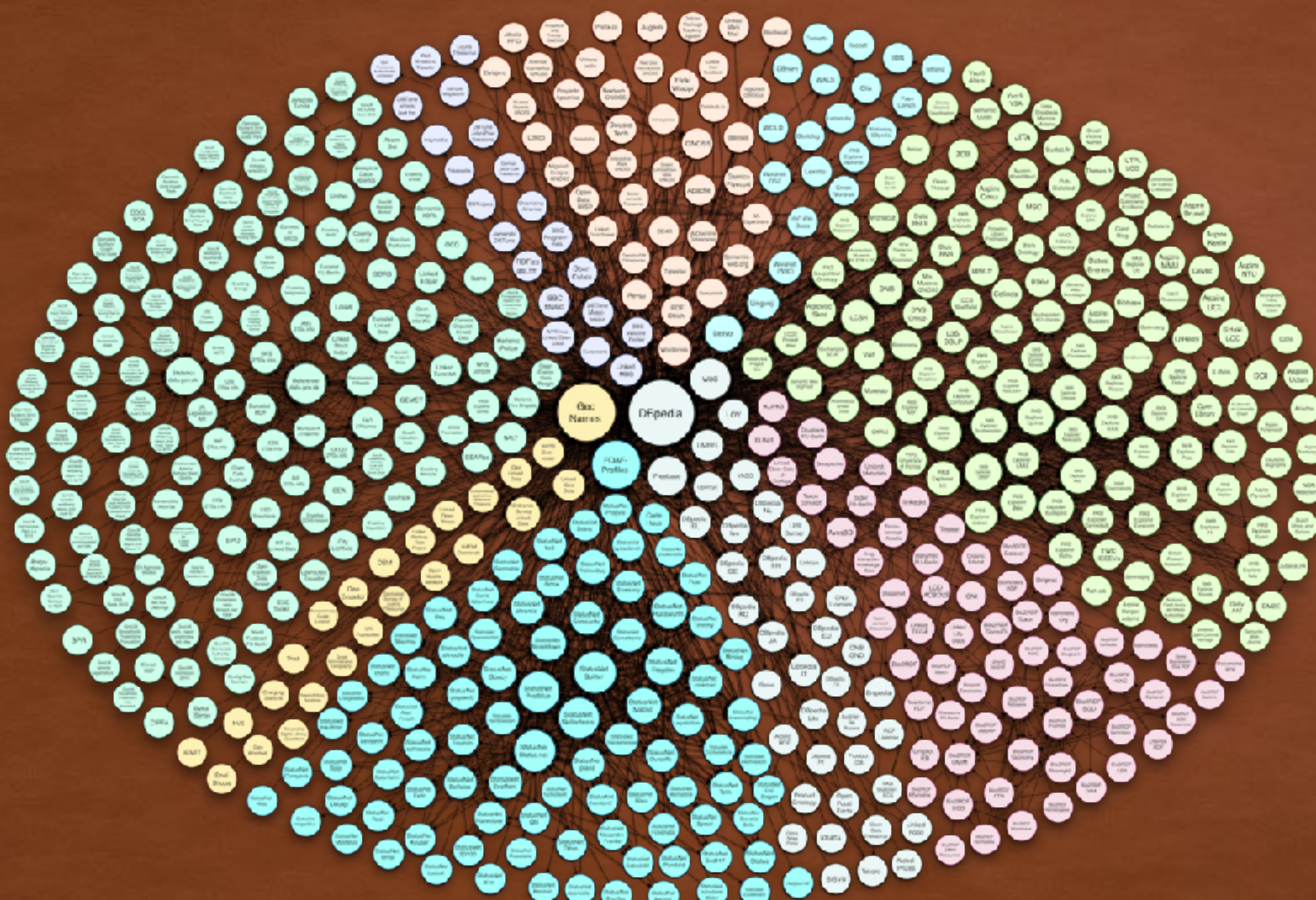
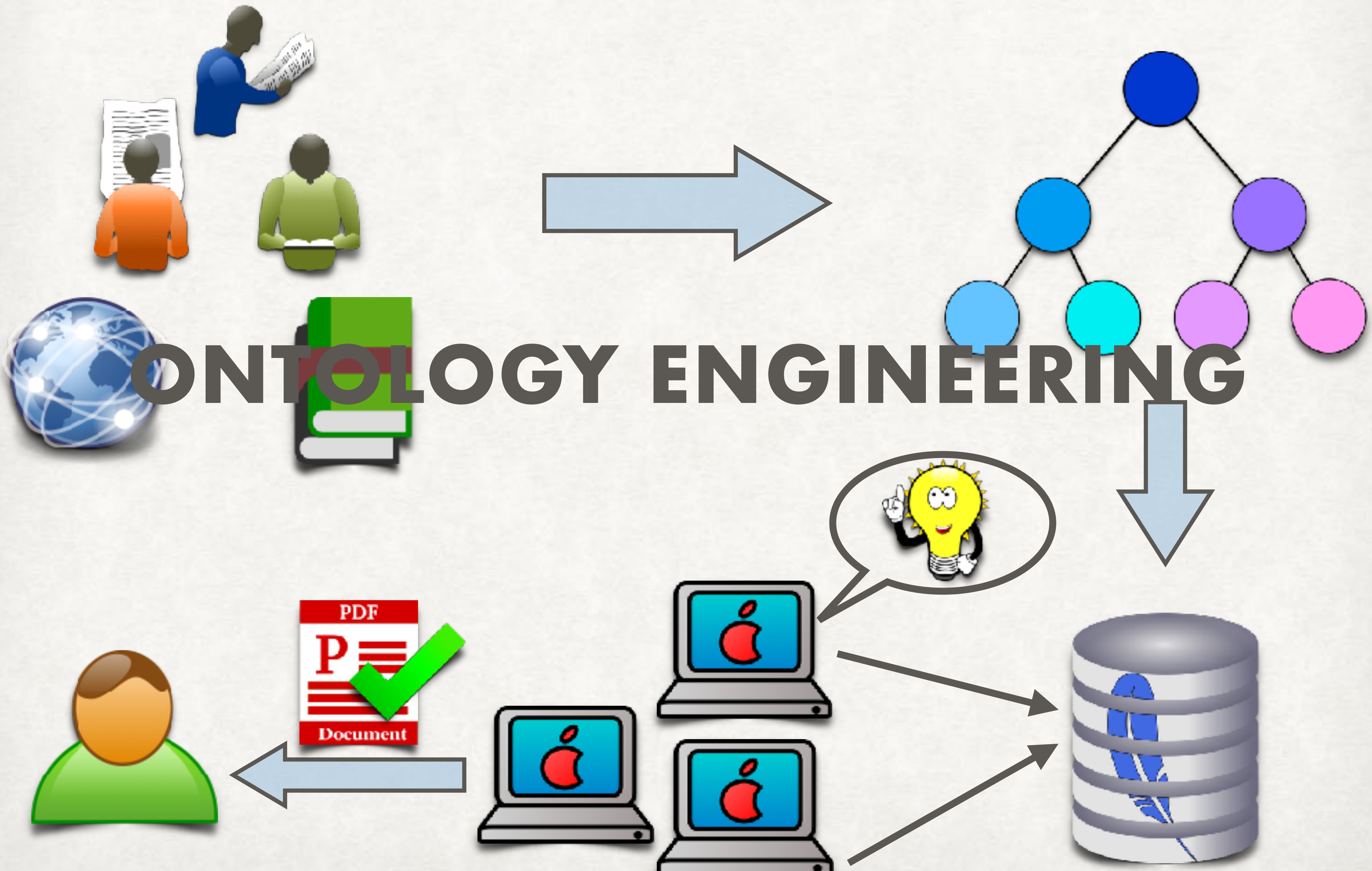
DEVELOPING KNOWLEDGE SYSTEMS

ONTOLOGY ENGINEERING



JOHANNES KINZIG – CURRENT TOPICS – 02. FEB. 2017

WHAT ARE WE TALKING ABOUT ???



TOC

DEVELOPING KNOWLEDGE SYSTEMS

ENGINEERING PROCESS

SOFTWARE ENGINEERING – TECHNICAL ENGINEERING

- Software Engineering Process - advanced discipline
  - **safety critical / business critical** … applications
  - reliable nowadays - **evolving** from day do day
  - process models - project models - clear structure
    - Waterfall — V-Modell — V-Modell-XT — AUP/RUP — Scrum

1. System Requirements Analysis	5. Coding
2. Software Requirements Analysis	6. Testing & Integration
3. Analysis (Tools, Libraries, HW-Arch.)	7. Operations
4. Program Design	

ONTOLOGY ENGINEERING PURPOSE

"APPLICATIONS MAY RELYING ON ONTOLOGIES"

- #, @ Notation
- "you were mentioned by …."
- Picture tagging by location, hashtag, etc.
- Proprietary Solutions (knowledge systems):
  - Catalog / Shop systems
  - ERP systems / business logic systems
  - Laws — Case Classification and judgements
  - Medicine — Modelling Diseases and symptoms

PREREQUISITES

ONTOLOGY ENGINEERING

- Open Standards for data exchange
  - closed for modification open for extension — future standards

1. Modelling Formalism
   1. Technical storage: relational oo-DB vs. semantic approach
   2. Semantics: Decision about semantics — RDF vs. OWL vs. OWL2
2. Requirements specification

GATHERING KNOWLEDGE

"TEACHING THE KNOWLEDGE SYSTEM"

- Ontologies/knowledge systems need to be trained automatically
- Huge amount of data
- "Knowledge" needs to be taken from existing data sources
  - People - **to be structured** (manually or automatically)
  - Books - **unstructured** source
  - “Internet" - **semistructured** source
  - Databases - **structured** source

MODELLING ONTOLOGIES

PRINCIPLES AND METHODS

1. Logical Criteria	5. Quantification and Quantifiers
2. Structural and Formal Criteria	6. Part and Subclass Identity
3. Accuracy Criteria	7. Subclasses and equivalent classes
4. Disjointness	8. Translate loosely from natural languages

MODELLING ONTOLOGIES

1. LOGICAL CRITERIA

- Model has to be **consistent** — ("also against real world")
- **Consistency:** Correct mapping of real world and **model/formalism**
- **Inconsistency: != Consistency**
  - logical consequences —> deduction wrong
  - inconsistent/unsatisfiable class: if class interpreted as empty set
- **Coherency**: ontology does not contain unsatisfiable/inconsistent classes

$$Horse \sqsubseteq \neg Flies$$
$$FlyingHorse \equiv Horse \sqcap Flies$$
$$FlyingHorse(Pegasus)$$

REASONER WILL ALARM

MODEL TURNS INCONSISTENT WHEN ADDING INSTANCE

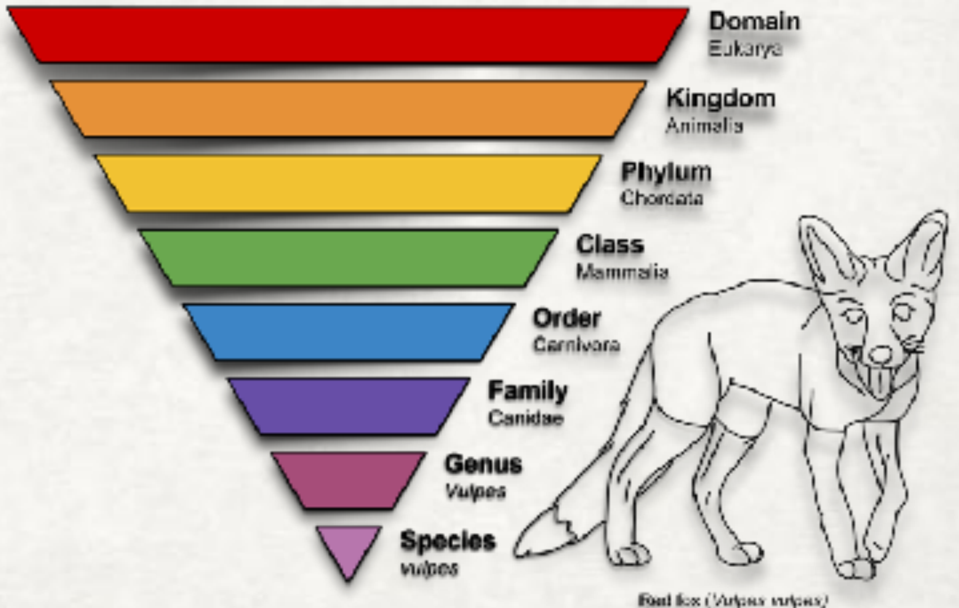
MODELLING ONTOLOGIES

2. STRUCTURAL AND FORMAL CRITERIA

TAXONOMIC – TAXON:
GROUP OF ONE OR MORE
POPULATIONS OF AN ORGANISM

- Taxonomic Cycles:
- Check for rigidity:
  - every member of a class cannot stop being a member without loosing existence

$$Architecture \sqsubseteq Faculty$$
$$Faculty \sqsubseteq University$$
$$University \sqsubseteq Building$$
$$Building \sqsubseteq Architecture$$



MODELLING ONTOLOGIES

3. ACCURACY CRITERIA

- Accuracy and granularity against "real-world-domain"
- Cannot be done automatically
- Double-Checking
- Modeling Samples for testing: **random sampling**

"The man saw her with the telescope"

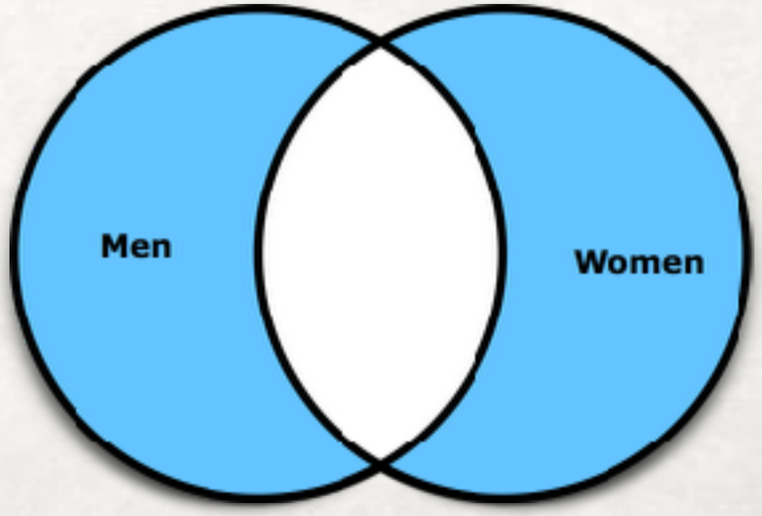
UNAMBIGUOUS DEFINITIONS !!!

MODELLING ONTOLOGIES

4. DISJOINTNESS

$$Woman \sqsubseteq Human, Human \sqsubseteq Man \sqcup Woman, Man \sqsubseteq Human$$
$$Woman(Anna), Man(Steve)$$

- Statement: ¬Man(Anna)
- no "logical reason" why "Anna" cannot be male and female
- **Real-life-logic**: "Anna" can definitely not be **a man and a woman**
- disjoint classes when necessary
  - People
  - when gender matters (e.g. animals)



MODELLING ONTOLOGIES

5. QUANTIFICATION AND QUANTIFIERS

- **existential quantifier** ($\exists$) >>> **universal quantifier** ($\forall$)

- **universal quantifier** ($\forall$) use when statements like
  - nothing but
  - only
  - exclusively

- Formalising: One wants to express “A car has wheels”
  - $\text{Car} \sqsubseteq \exists \text{has.Wheel}$ ✔
  - $\text{Car} \sqsubseteq \forall \text{has.Wheel}$ ✘
    - car has only wheels (if it has anything at all)

MODELLING ONTOLOGIES

6. PART AND SUBCLASS IDENTITY

$$Finger \sqsubseteq Hand, Hand \sqsubseteq Arm, Arm \sqsubseteq Body$$
$$Toe \sqsubseteq Foot, Foot \sqsubseteq Leg, Leg \sqsubseteq Body$$
$$Arm \sqcap Leg \sqsubseteq \bot$$
$$\text{(Arm and Leg are disjoint)}$$

- Modell Deduction: **Arm(myLeftThumb)**
  - thumb is not only a finger it is also a **hand** and an **arm** (according to model)
  - subclass relation **partOf** was used mistakenly
  - both subclasses share the property of "**belonging to something**" ???
- introducing a new role: **partOf**

$$Finger \sqsubseteq \exists partOf.Hand, Hand \sqsubseteq \exists partOf.Arm, Arm \sqsubseteq \exists partOf.Body$$
$$Toe \sqsubseteq \exists partOf.Foot, Foot \sqsubseteq \exists partOf.Leg, Leg \sqsubseteq \exists partOf.Body$$
$$Arm \sqcap Leg \sqsubseteq \bot$$

MODELLING ONTOLOGIES

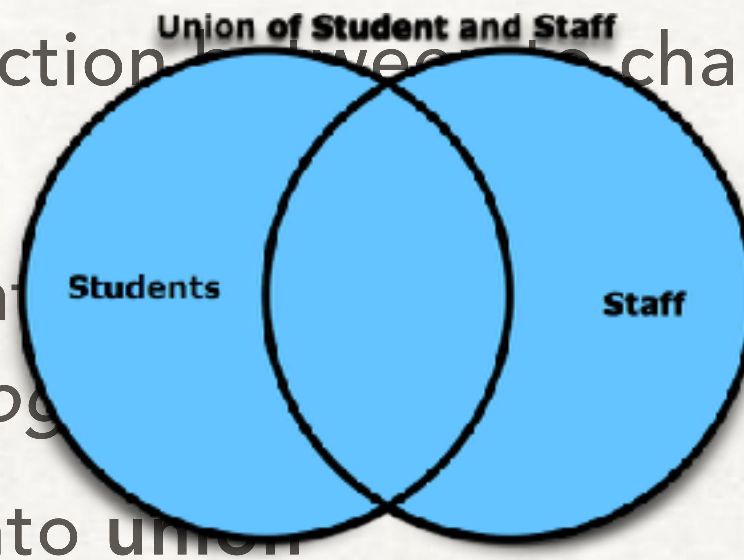
7. SUBCLASSES AND EQUIVALENT CLASSES

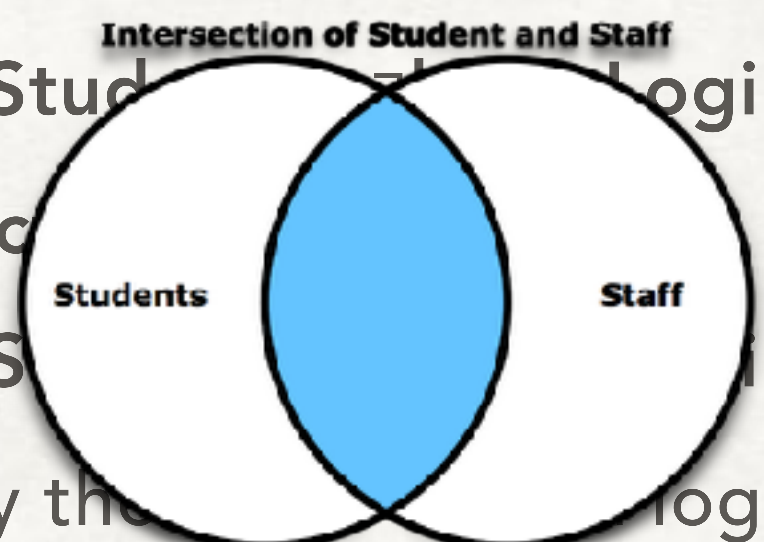
- Subclass or equivalent class?
- Subclassing: express characteristic about members of a class
- **LivingInWater** — a member **LivingInWater(fish)** is a fish
  - **necessary criterion** for being a fish — do not formulate **sufficient criterion** (plankton, coral)
- Equivalence statement: can be used iff a class description is **necessary** and **sufficient**
  - **Winner ≡ Player⊓∀hasCompleteCollection.Spades**
  - a player can only be a winner iff he is playing a game and is holding a whole collection of spades

MODELLING ONTOLOGIES

8. TRANSLATE LOOSELY FROM NATURAL LANGUAGES

- Misunderstanding when using ”and”
- not always an intersection … characteristics or properties
- Formalising statement … *d students of the university will get a log*…
  - ”and” translated into **u**…
    - **StaffMember ⊔ Stu**… **oginAccount**
  - **not** into an **intersec**…
    - **StaffMember ⊓ S**… **inAccount**
    - intersection: only th… login account who are **students and staff members**





DEMONSTRATION

NATURAL LANGUAGE TOOLKIT

NATURAL LANGUAGE PROCESSING FOR AUTOMATIC KNOWLEDGE GATHERING

python

http://www.nltk.org

https://upload.wikimedia.org/wikipedia/commons/f/f8/Python_logo_and_wordmark.svg

DEMONSTRATION: NLTK

NATURAL LANGUAGE TOOLKIT

- Natural language processing
  - Possibility to electronically **understand natural language**
  - —> **gathering knowledge**
  - Search Engines (Google, Yahoo, Qwant…) —> Question
  - Hard task: different types of languages (grammar, rules, etc.)
    - (- English, German - Japanese, Chinese -)
  - AI principles, machine learning techniques, huge amount of training data

CHALLENGES - NLP

DEMONSTRATION: NLTK

- Grammatical structure by language (Statements, Questions)
  - Special **grammar** for each language (comp. English and Chinese)
- Tenses: **Rules** apply, but **Exceptions** are common
  - Depends on language
- Language Recognition:
  - Regular Expressions (earlier days, nowadays)
  - AI methods - machine learning, neural networks (nowadays)

SOURCES

- See attached paper:
  - The ontology engineering process