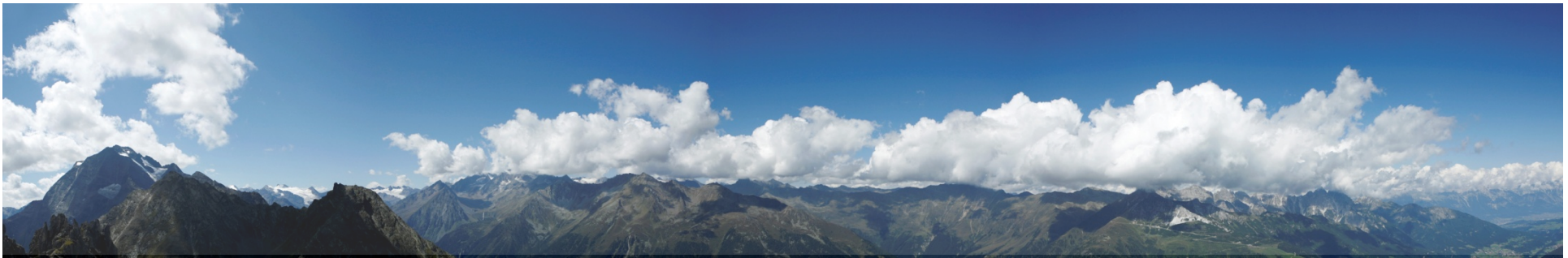



Semantic Web

Ontologies and the Semantic Web



Where are we?

#	Title
1	Introduction
2	Semantic Web Architecture
3	Resource Description Framework (RDF)
4	Web of data
5	Generating Semantic Annotations
6	Storage and Querying
7	Web Ontology Language (OWL)
8	Rule Interchange Format (RIF)
9	Reasoning on the Web
 10	Ontologies
11	Social Semantic Web
12	Semantic Web Services
13	Tools
14	Applications

1. Motivation
2. Technical solution and illustrations
 1. Foundations
 1. Definitions, features, classifications, applications, modelling principles
 2. Ontology engineering
 1. Methodologies and methods and tools to build, manage and use ontologies
 3. How to build an ontology (Ontology 101)
3. Extensions
4. Large example
5. Summary
6. References

A large part of this lecture is based on the book
Gomez-Perez et al., Ontological Engineering, Springer, 2004.

MOTIVATION

- Ontologies provide a shared understanding of a domain.
- They provide background knowledge to systems to automatize certain tasks.
- By the process of annotation, knowledge can be linked to ontologies.
 - Example: “Angelina Jolie” (Text) linked to concept Actress
 - In our ontology we also know that an actress always is female and a person.
- Ontologies allow the creation of annotations → machine-readable and machine-understandable content.
- If machines can understand content, they can also perform more meaningful and intelligent queries.
 - Distinction of Jaguar the animal and the car.
 - Combination of information that is distributed on the Web.

An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base.

A. Bernaras; I. Laresgoiti; J. Corra. *Building and Reusing Ontologies for Electrical Network Applications ECAI96. 12th European conference on Artificial Intelligence.* Ed. John Wiley & Sons, Ltd. 298-302

- Ontologies are the backbone of the Semantic Web.
- They provide the knowledge that is required for semantic applications of all kinds.
- They are essential to semantic annotation (see previous lecture).

Now we look at a potential application of ontologies in the Ontobroker system for searching and querying the Web. This example illustrates why ontologies are the „backbone of the Semantic Web“.

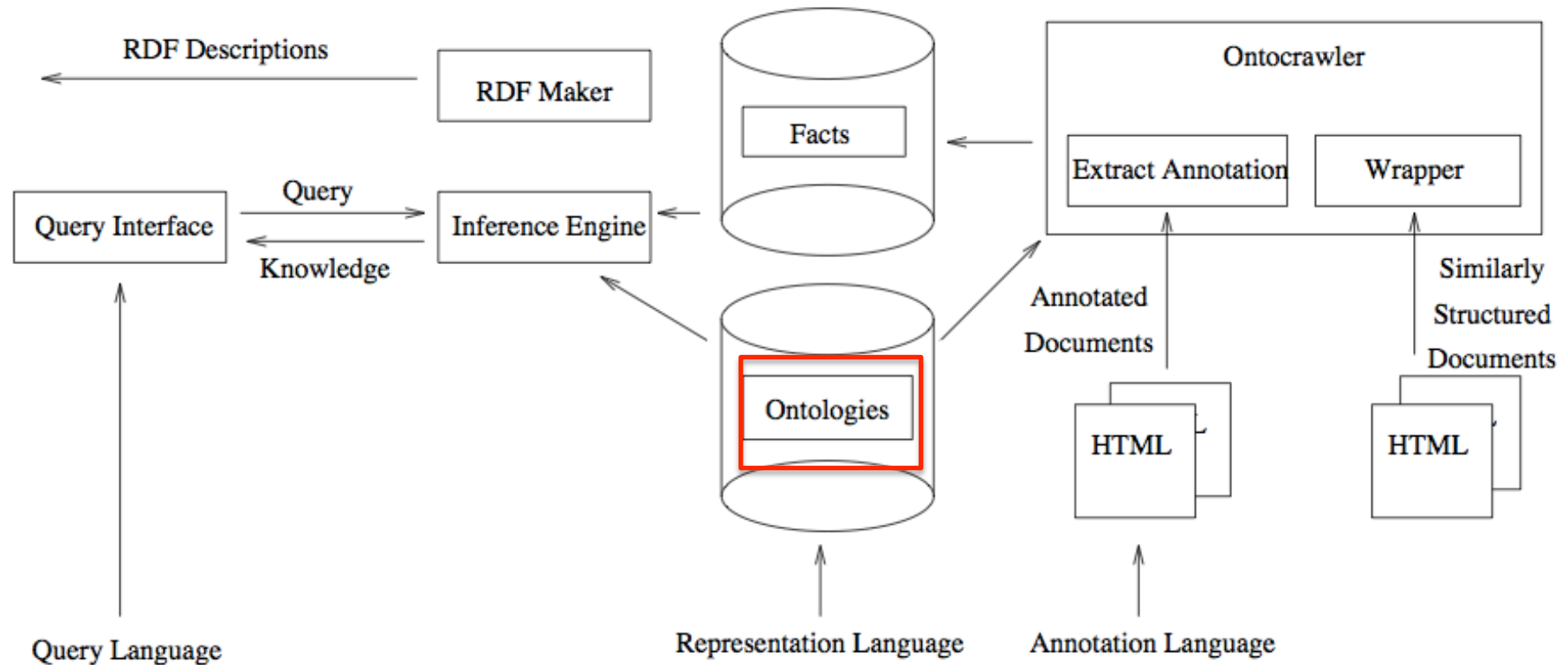
The Semantic Web Killer app: search and query the Web

- The Web is a constantly growing network of distributed resources
 - More than 1 trillion unique URLs
 - More than 100 billion pages
 - More than 200 million web sites
 - Check most updated data on:
http://news.netcraft.com/archives/web_server_survey.html
- User needs to be able to efficiently search resources/content over the Web
 - When I Google “Milan” do I find info on the city or the soccer team?
- User needs to be able to perform query over largely distributed resources
 - When is the next performance of the rock band “U2”, where it will be located, what are the best way to reach the location, what are the attractions nearby...

- Despite its enormous size, the Web's support for query answering and automated inference is very limited.
- Ontobroker is a system which uses formal ontologies to extract, reason, and generate metadata in the WWW.
- The system provides a means for semantic based query handling even if the information is spread over several sources.

S. Decker, M. Erdmann, D. Fensel, and R. Studer:
Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Database Semantics, Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999.

Ontobroker: overview



Graphic from S. Decker, M. Erdmann, D. Fensel, and R. Studer:
Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Database Semantics, Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999.

The heart of the system are **ontologies**. They are used in several components of the system. The components are the following:

- The **Ontocrawler** extracts formal knowledge from HTML pages. This is done in two different ways:
 - For large collections of web pages with a similar structure a wrapper generates formal descriptions of the content of the pages in relation to a certain ontology.
 - Often the effort for constructing specialized wrappers is too high: in this case an annotation language is used for enabling providers to enrich web documents with ontological information in an integrated, maintenance-friendly manner.
- The **inference engine** exploits the formal semantics of the representation language and enables well defined automatic reasoning.
- The **RDF-Maker** exploits the inference engine and generates an RDF representation of information inferable from the ontology and the facts with respect to a given web resource.
- The **query interface** enables the interactive formulation of queries while browsing the ontology and selecting the terms constituting the query.

- Ontocrawler searches through a fragment of the WWW that makes use of one of the ontologies and collects these knowledge fragments.
- It implements a wrapper that translates annotated web documents into facts formulated in the representation language.
- The knowledge *provider* has to use the annotation language.
- Each provider of an ontologically annotated knowledge portion has to do an index of his annotated documents and he has to use the annotation language and an ontology of the Ontobroker to annotate these documents.
- Ontobroker uses a web crawler to collect HTMLa (a format that integrates annotations directly into the HTML code) pages from the web, extracts their annotations, and parses them into the internal format of the system.
- RDF-A pages and RDF repositories can be included directly.
- HTML and XML data sources require processing provided by wrappers to derive RDF data.

-
- The inference engine receives the query of a client and uses two information sources for deriving an answer.
 - It uses the ontology chosen by the clients and it uses the facts that were found by the Ontocrawler in the WWW.

- RDF-Maker generates RDF from the ontological facts that are stored in the knowledge base.
- In this process, the inference engine might generate additional facts, that are implicit before and makes them explicit.

- The user communicates with the inference engine using a query interface.
- Two different interfaces are available: a graphical one, that hides the syntax of the query language and enables a direct access to the ontology;
- There is also a conventional, text base interface, where a user can directly type in queries in the query language.

- Ontologies are the underlying component of the whole Ontobroker system.
- They provide the structure and underlying knowledge for all facts that are stored in the knowledge base.
- Ontologies are used when inferring additional knowledge by the inference engine.
- Ontologies are used for query formulation and answering.

Today's lecture will show what ontologies are and how they can be built.

TECHNICAL SOLUTION AND ILLUSTRATIONS

FOUNDATIONS

-
- ***An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary***

Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W.R.
Enabling Technology for Knowledge Sharing. AI Magazine. Winter 1991. 36-56

- ***An ontology is an explicit specification of a conceptualization***

Gruber, T. *A translation Approach to portable ontology specifications. Knowledge Acquisition. Vol. 5. 1993. 199-220*

- *An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base*

B. Swartout; R. Patil; k. Knight; T. Russ. *Toward Distributed Use of Large-Scale Ontologies* **Ontological Engineering**. AAAI-97 Spring Symposium Series. 1997. 138-148

- *An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base*

A. Bernaras; I. Laresgoiti; J. Correra. *Building and Reusing Ontologies for Electrical Network Applications* **ECAI96. 12th European conference on Artificial Intelligence**. Ed. John Wiley & Sons, Ltd. 298-302

What is an ontology (iii)?

“An ontology is a formal, explicit specification of a shared conceptualization”

formal
Machine-readable

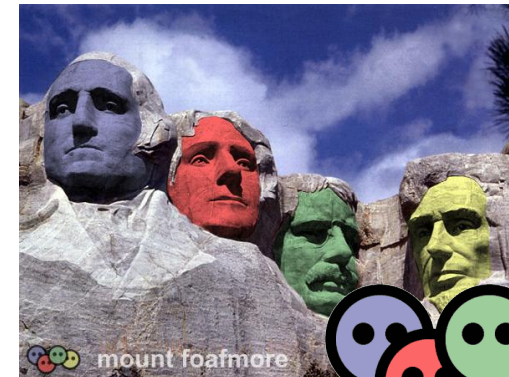
explicit specification of a
Concepts, properties
relations, functions,
constraints, axioms,
are explicitly defined

shared conceptualization
Consensual
Knowledge

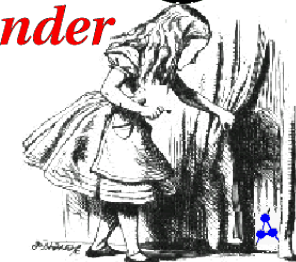
shared conceptualization
Abstract model and
simplified view of some
phenomenon in the world
that we want to represent

Studer, Benjamins, Fensel. Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*. 25 (1998) 161-197

Examples



Wonder



Web



OpenGALEN

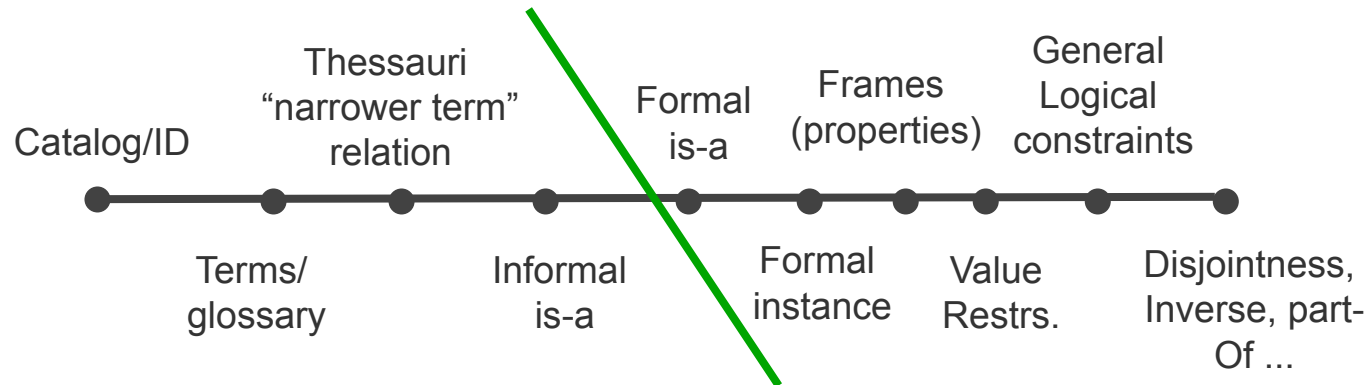


SUMO



-
- Modelled knowledge about a specific domain
 - Defines
 - A common vocabulary
 - The meaning of terms
 - How terms are interrelated
 - Consists of
 - Conceptualization and implementation
 - Contains
 - Ontological primitives

Classifications of ontologies



Lassila O, McGuinness D. The Role of Frame-Based Representation on the Semantic Web. Technical Report. Knowledge Systems Laboratory. Stanford University. KSL-01-02. 2001.

- Ontologies can be built using various languages with various degrees of formality
 - Natural language
 - UML
 - ER
 - OWL/RDFS
 - WSML
 - FOL
 - ...
- The formalism and the language limit the kind of knowledge that can be represented
- A domain model is not necessarily a formal ontology only because it is written in OWL

- Knowledge representation
 - Ontology models domain knowledge
- Semantic annotation
 - Ontology is used as a vocabulary, classification or indexing schema for a collection of items
- Semantic search
 - Ontology is used as a query vocabulary or for query rewriting purposes
- Configuration
 - Ontology defines correct configuration templates

- Clarity:
 - To communicate the intended meaning of defined terms
- Coherence:
 - To sanction inferences that are consistent with definitions
- Extendibility:
 - To anticipate the use of the shared vocabulary
- Minimal Encoding Bias:
 - To be independent of the symbolic level
- Minimal Ontological Commitments:
 - To make as few claims as possible about the world

Gruber, T., *Towards Principles for the Design of Ontologies*.
KSL-93-04. Knowledge Systems Laboratory. Stanford University. 1993

An ontology should communicate effectively the intended meaning of defined terms. Definitions should be objective. Definitions can be stated on formal axioms, and a complete definition (defined by necessary and sufficient conditions) is preferred over a partial definition (defined by only necessary or sufficient conditions)...

```
(define-class Travel (?travel)
  "A journey from place to place"
:axiom-def
  (and (Superclass-Of Travel Flight)
        (Subclass-Of Travel Thing)
        (Template-Facet-Value Cardinality
         arrivalDate Travel 1)
        (Template-Facet-Value Cardinality
         departureDate Travel 1)
        (Template-Facet-Value Maximum-Cardinality
         singleFare Travel 1))
:def
  (and (arrivalDate ?travel Date)
        (departureDate ?travel Date)
        (singleFare ?travel Number)
        (companyName ?travel String)))
```

No clarity →



```
(define-class Travel (?travel)
  "A journey from place to place"
  :axiom-def
  (and (Superclass-Of Travel Flight)
        (Subclass-Of Travel Thing)
        (Template-Facet-Value Cardinality
         arrivalDate Travel 1)
        (Template-Facet-Value Cardinality
         departureDate Travel 1)
        (Template-Facet-Value Maximum-Cardinality
         singleFare Travel 1))
  :iff-def
  (and (arrivalDate ?travel Date)
        (departureDate ?travel Date))
  :def
  (and (singleFare ?travel Number)
        (companyName ?travel String)))
```

Clarity

→ :iff-def

*An ontology should be coherent: that is, it should sanction inferences that are consistent with the definitions.[...]
If a sentence that can be inferred from the axioms contradicts a definition or example given informally, then the ontology is incoherent.*

```
(define-axiom No-Train-between-USA-and-Europe
  "It is not possible to travel by train between the USA and Europe"
:= (forall (?travel)
  (forall (?city1)
    (forall (?city2)
      (=> (and (Travel ?travel)
        (arrivalPlace ?travel ?city1)
        (departurePlace ?travel ?city2)
        (or (and (EuropeanLocation ?city1)
          (USALocation ?city2))
          (and (EuropeanLocation ?city2)
            (USALocation ?city1) )))
        (not (TrainTravel ?travel)))))))
(define-instance Madrid (EuropeanLocation))
(define-instance NewYork (USALocation))
```

*One should be able to define new terms
for special uses based on the existing vocabulary,
in a way that does not require the revision of the existing definitions.*

- Currency dimension
- Definition of currencies
- Relationship between currencies

```
(define-individual Euro (Unit-of-Measure)
  "An Euro is the currency on the European Union"
:= (* 0,96 USDollar)
:axiom-def
  (= (Quantity.dimension Euro) CurrencyDimension))
```

The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding.

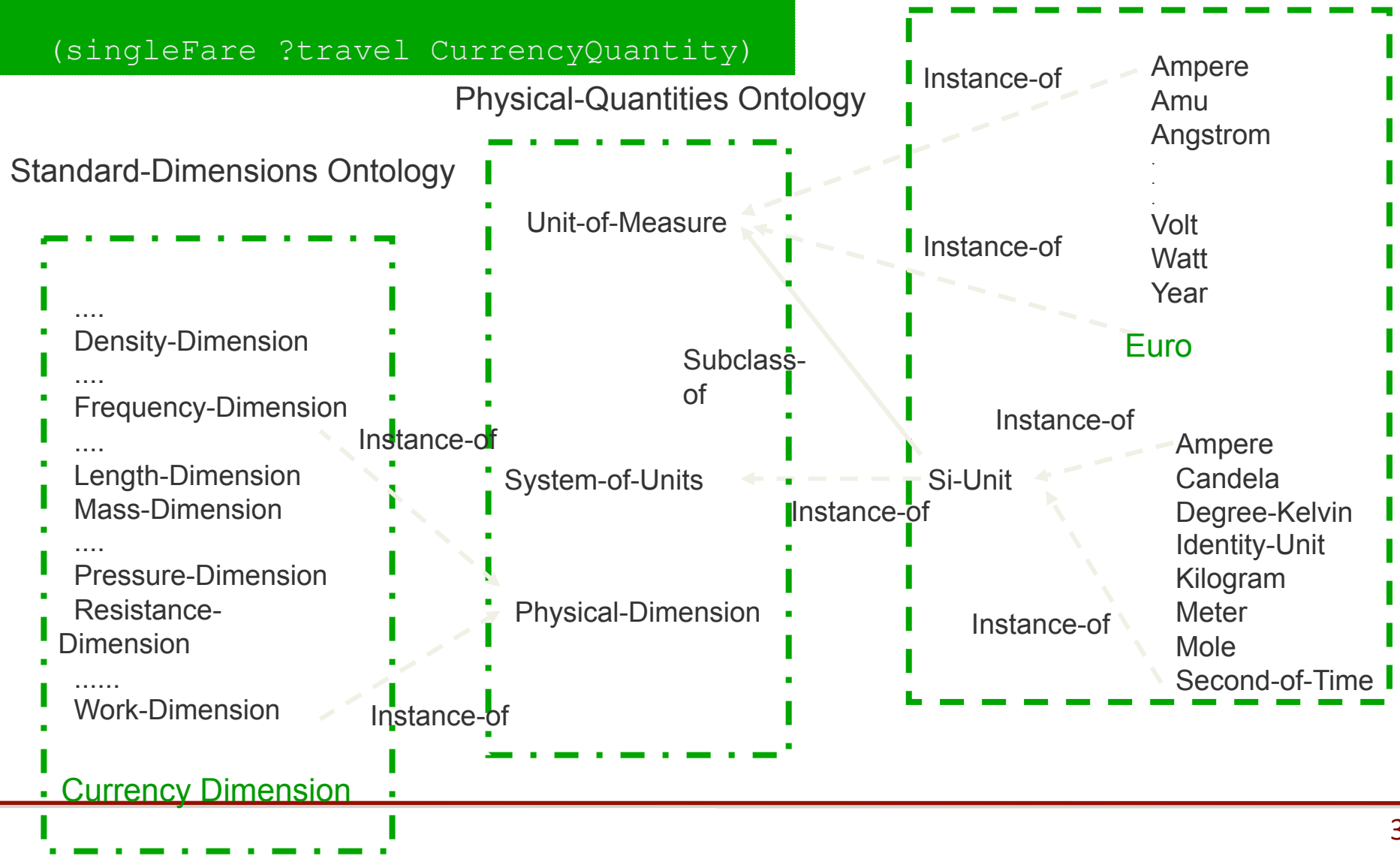
```
(define-class Travel (?travel)
  "A journey from place to place"
:axiom-def
  (and (Superclass-Of Travel Flight)
        (Subclass-Of Travel Thing)
        (Template-Facet-Value Cardinality
         arrivalDate Travel 1)
        (Template-Facet-Value Cardinality
         departureDate Travel 1)
        (Template-Facet-Value Maximum-Cardinality
         singleFare Travel 1))
:iff-def
  (and (arrivalDate ?travel Date)
        (departureDate ?travel Date))
:def
  (and (singleFare ?travel Number)
        (companyName ?travel String)))
```

No minimal encoding bias



Minimal Encoding Bias

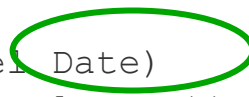
(singleFare ?travel Number)
 should be substituted by:
 (singleFare ?travel CurrencyQuantity)



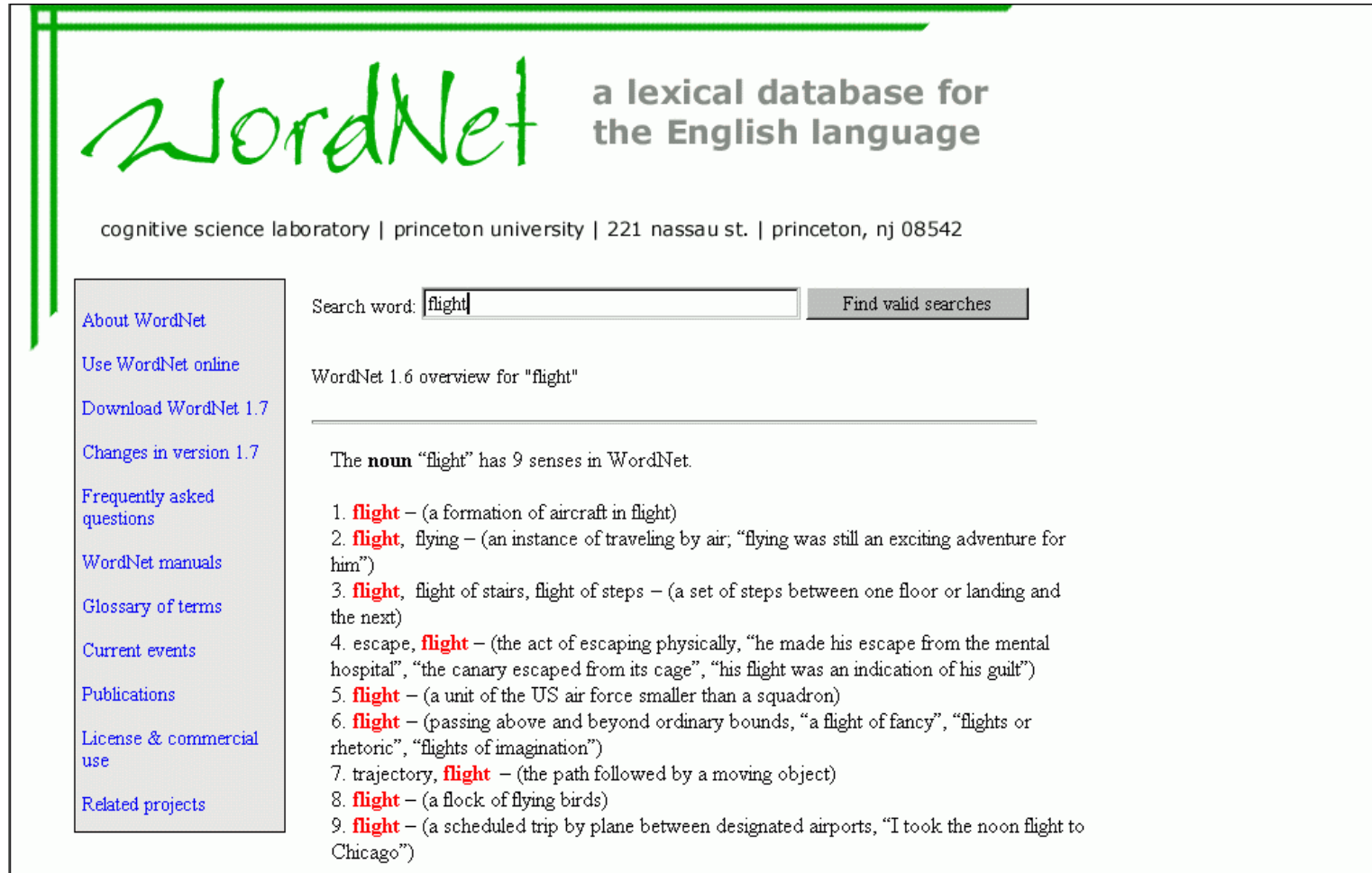
Since ontological commitment is based on the consistent use of the vocabulary, ontological commitment can be minimized by specifying the weakest theory and defining only those terms that are essential to the communication of knowledge consistent with the theory.

```
(define-class Travel (?travel)
  "A journey from place to place"
:axiom-def
  ( . . . . )
:iff-def
  (and (arrivalDate ?travel Date)
        (departureDate ?travel Date))
:def
  (and (singleFare ?travel Number)
        (companyName ?travel String)))
```

- Absolute/relative date?
- Format?



- Agreements to use the vocabulary in a coherent and consistent manner (Gruber)
 - An agent commits (conforms) to an ontology if it “acts” consistently with the definitions
- Connection between the ontology vocabulary and the meaning of the terms of such vocabulary



WordNet a lexical database for the English language

cognitive science laboratory | princeton university | 221 nassau st. | princeton, nj 08542

[About WordNet](#)
[Use WordNet online](#)
[Download WordNet 1.7](#)
[Changes in version 1.7](#)
[Frequently asked questions](#)
[WordNet manuals](#)
[Glossary of terms](#)
[Current events](#)
[Publications](#)
[License & commercial use](#)
[Related projects](#)

Search word:

WordNet 1.6 overview for "flight"

The **noun** "flight" has 9 senses in WordNet.

1. **flight** – (a formation of aircraft in flight)
2. **flight**, flying – (an instance of traveling by air, "flying was still an exciting adventure for him")
3. **flight**, flight of stairs, flight of steps – (a set of steps between one floor or landing and the next)
4. escape, **flight** – (the act of escaping physically, "he made his escape from the mental hospital", "the canary escaped from its cage", "his flight was an indication of his guilt")
5. **flight** – (a unit of the US air force smaller than a squadron)
6. **flight** – (passing above and beyond ordinary bounds, "a flight of fancy", "flights or rhetoric", "flights of imagination")
7. trajectory, **flight** – (the path followed by a moving object)
8. **flight** – (a flock of flying birds)
9. **flight** – (a scheduled trip by plane between designated airports, "I took the noon flight to Chicago")

- **The representation of disjoint and exhaustive knowledge**
 - If the set of subclasses of a concept are disjoint, we can define a disjoint decomposition. The decomposition is exhaustive if it defines the super-concept completely.
- **The standardization of names.**
 - To ease the understanding of the ontology the same naming conventions should be used to name related terms.

ONTOLOGY ENGINEERING

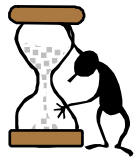
What is ontology engineering?

- *“the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies”*

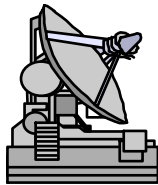
Gomez-Perez et al, 2004

Ontology engineering activities

Management



Scheduling



Control



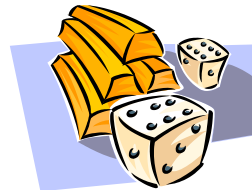
Quality assurance

Development oriented

Pre-development



Environment study



Feasibility study

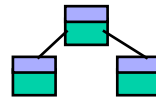
Development



Specification



Conceptualization

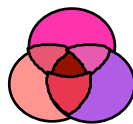


Formalization



Implementation

Post-development

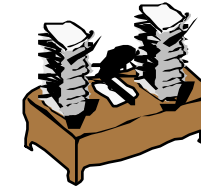


Maintenance



Use

Support



Knowledge acquisition



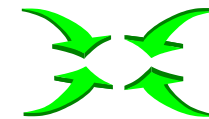
Evaluation



Integration



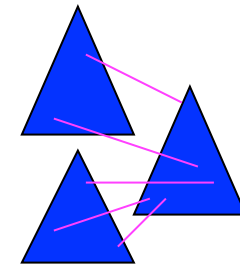
Documentation



Merging



Configuration management



Alignment

-
- Validated guidelines on how the ontology building process should be structured
 - Well-organized process instead of „ontology development driven by inspiration and intuition only“
 - Methodologies do not support all of the aforementioned activities
 - They implicitly assume a particular development paradigm for the ontology engineering process
 - Some of them also provide supporting methods and tools

-
- Two or more people interact and exchange knowledge in order to build a common, shared ontology in pursuit of a shared, collective, bounded goal.
 - *Interaction may be indirect but required.*
 - *Argumentation as a common interaction means.*
 - *Simple contributions not enough.*
 - *Bounded goal: beginning and end.*
 - *Collaborators may have individual goals.*

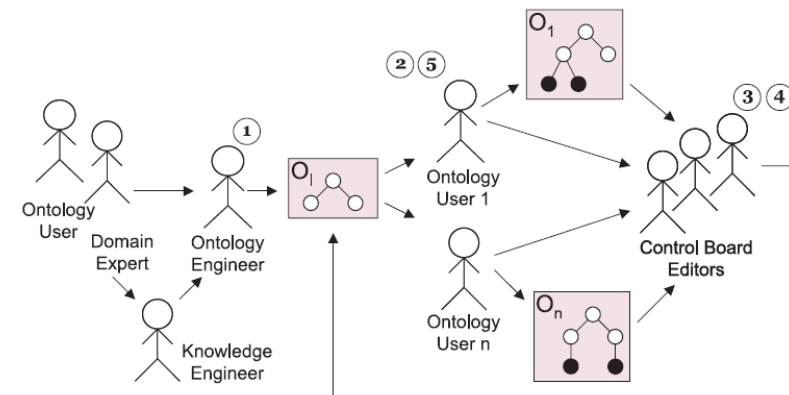
-
- Identify purpose
 - Build ontology
 - Capture
 - Coding
 - Integrating
 - Evaluation
 - Documentation

- Identify motivating scenarios
- Elaborate informal competency questions
- Specify terminology in FOL
 - Identify objects
 - Identify predicates
- Formal competency questions
- Specify axioms in FOL
- Specify completeness theorems

-
- Develop + Management Activities + Support in parallel
 - Develop:
 - Specification
 - Conceptualization
 - Formalization
 - Implementation
 - Maintenance
 - Focus on the conceptualization activity
 - Advantage: Integration of existing ontologies considered from early on
 - Conceptualization is evaluated early on, which prevents propagation of errors.

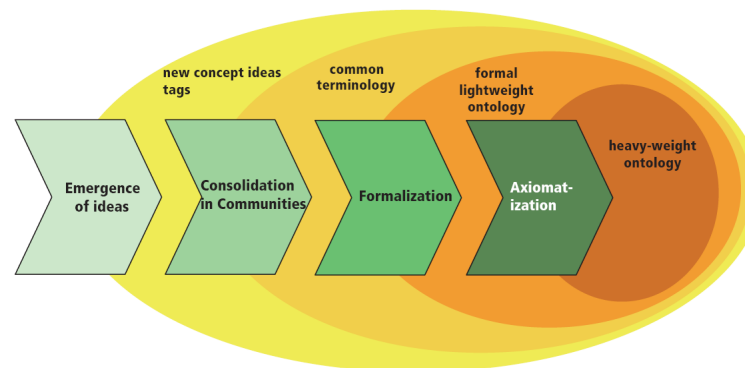
-
- Feasibility study
 - Kick-off: describe domain and goals, design guidelines (e.g. naming conventions), available sources
 - Refinement
 - Knowledge elicitation with domain experts
 - Formalization
 - Evaluation
 - Maintenance

1. **Build:** a small team builds an ontology which is not required to be complete.
2. **Local adaption:** users adapt it to their own needs in their local environments.
3. **Analysis:** the board selects which changes go to the next version of the ontology.
4. **Revision:** carried out regularly to avoid that the ontology diverges too far from the local adaptations.
5. **Local updates** may be performed if the users wish to align their local ontologies with the new version of the ontology.



Vrandecic et al., 2005

1. Emergence of ideas. In the first phase, new concept ideas are collected in an ad-hoc fashion. This is done using simple tags.
2. Consolidation in communities. The concept symbols generated in the first phase are re-used and adapted by the user community. The phase aims at extracting concepts from the available tags leading to a common terminology.
3. Formalization. This phase adds taxonomic and ad-hoc relations to the common terminology yielding lightweight but formal ontologies.
4. Axiomatization. The final step in the methodology addresses knowledge workers, i.e. ontology engineers, to add axiom leading to a heavy-weight ontology.



1. Preparation phase defines design criteria, determines boundary conditions and determines standards that can then be used for evaluation.
2. Anchoring produces a first ontology that helps for orientation of the participants. This ontology is refined in several steps.
3. Iterative improvement is an adaptation of the Delphi method, a technique for collecting views of several stakeholders. This process is repeated until consensus is reached.
4. Application is the actual usage of the ontology.

1. Requirements workflow
 - a) Determining the domain of interest and scope.
 - b) Defining the purpose involves a description of the application of the ontology or a motivating scenario.
 - c) Define storyboard and terminology.
 - d) Creating an application lexicon can be carried out in a semi-automatic fashion that can be supported by automatic tools that extract knowledge from documents.
 - e) Identifying the competency questions.
 - f) Use case identification.
2. The analysis workflow depicts the conceptual analysis of the domain in order to derive ontology requirements and ontological commitments.
 - a) Considering reuse of existing resources.
 - b) Modeling the application scenario using UML diagrams.
 - c) Building the glossary.
3. The design workflow refines results of previous steps and specifies relationships.
4. The implementation workflow leads to a serialized ontology in the preferred ontology representation formalism. The authors recommend OWL.
5. The test workflow validates the ontology by testing against the motivation scenario.

1. Specification.
 - a) identified the collaborators (team building), the knowledge workers
 - b) discuss requirements, produce specification documents, and
 - c) reach consensus on the scope and aim of the ontology.

 2. Conceptualization first takes place in personal spaces addressing the following tasks:
 - a) import of existing ontologies from ontology libraries,
 - b) consult generic top ontologies for better understanding,
 - c) improvising of ontologies, i.e. from-scratch-development, based on domain experts' views,
 - d) management, mapping and merging of various ontology versions,
 - e) comparison of available ontology versions,
 - f) enriching ontology concepts with more information and specification details.

 3. Exploitation. In this phase, the developed ontologies are pushed from the personal spaces to the shared space in order to reach a common understanding via structured conversation and criticism.
 - a) inspection of agreed or shared ontologies by collaborators,
 - b) comparisons of versions of one ontology in order to spot differences, and
 - c) publication of comments and feedback.
-

HOW TO BUILD AN ONTOLOGY

[Natalya F. Noy](#) and [Deborah L. McGuinness](#). "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report [KSL-01-05](#) and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

Step 1: Determine the domain and scope of the ontology

-
- What is the domain that the ontology will cover?
 - For what we are going to use the ontology?
 - For what types of questions the information in the ontology should provide answers?
 - Who will use and maintain the ontology?

-
- A set of queries which place demands on the underlying ontology.
 - Ontology must be able to represent the questions using its terminology and the answers based on the axioms
 - Ideally, in a staged manner, where consequent questions require the input from the preceding ones.
 - A rationale for each competency question should be given.

-
- Reuse ensures interoperability and reduces costs
 - Ontology libraries and tools for customization are required for this step
 - Sub-steps
 - Discover potential reuse candidates
 - Evaluate their usability
 - Customize ontologies to be reused
 - Integrate and merge to the target ontology

Step 3: Enumerate important terms in the ontology



- What are the terms we would like to talk about?
- What properties do those terms have?
- What would we like to say about those terms?

Step 4: Define classes and class hierarchy

- A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts.
- A bottom-up development process starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.
- Middle-out approach: define the more salient concepts first and then generalize and specialize them appropriately.

- From the list created in Step 3, select the terms that describe objects having independent existence rather than terms that describe these objects.
 - These terms will be classes in the ontology.
- Organize the classes into a hierarchical taxonomy by asking if by being an instance of one class, the object will necessarily (i.e., by definition) be an instance of some other class.
 - *If a class A is a superclass of class B, then every instance of B is also an instance of A.*
- Classes as unary predicates—questions that have one argument. For example, “Is this object a wine?”
 - Later: binary predicates (or slots)—questions that have two arguments. For example, “Is the flavor of this object strong?”
“What is the flavor of this object?”

-
- Interview: talk to subject matter experts.
 - Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
 - Observation and reflection.

 - Typical candidates for classes: **NOUNS**.
 - But: actors of use cases do not necessarily correspond to classes.

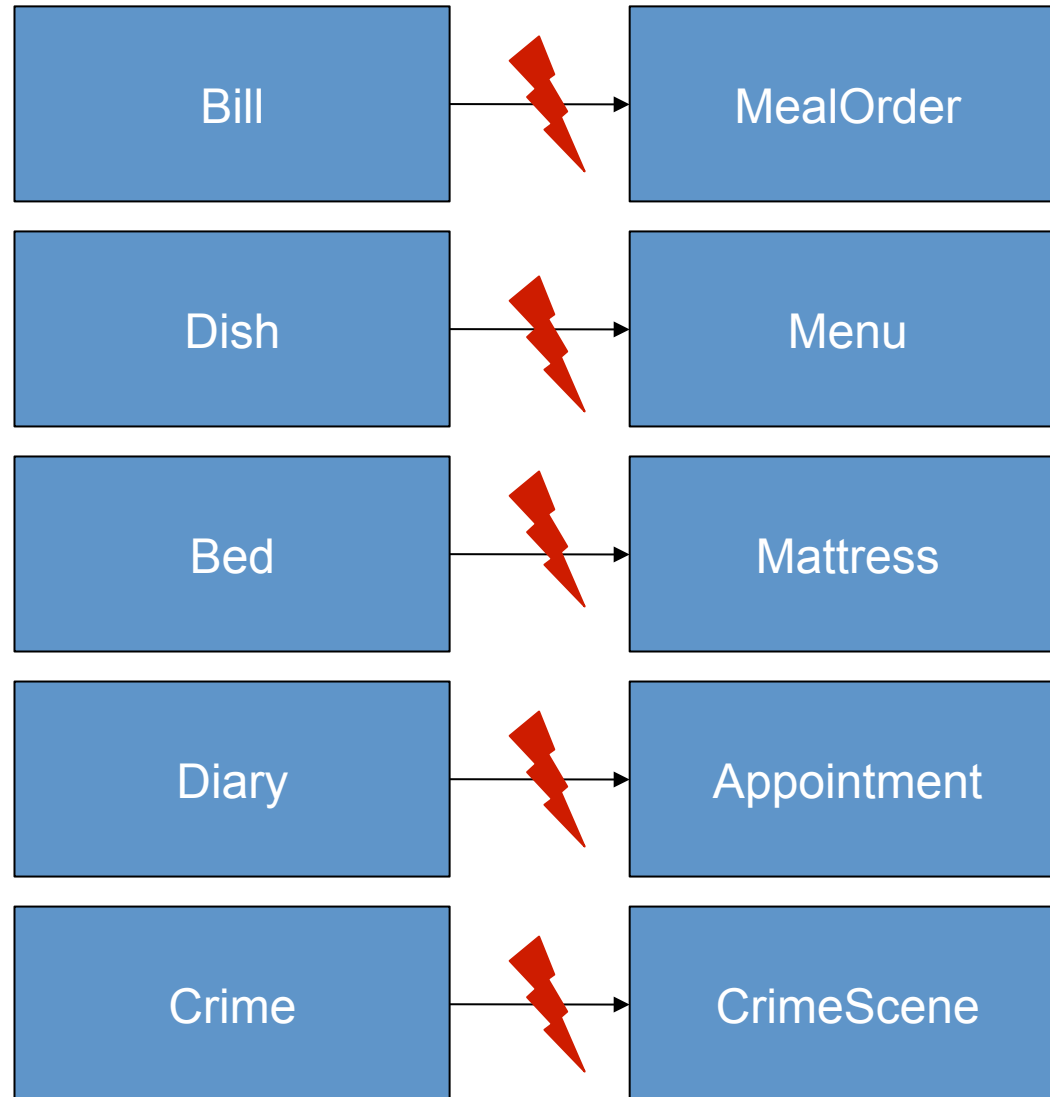
 - Other terms as well
 - Gerund: „My eyes glazing over...“ ~ withdrawal
 - Verbs: an association which starts to take on attributes and associations of its own turns into an entity: „Officer arrests suspect“.
 - Verbs: events: „Illness episode“.
 - Passive form: re-formulate in active form.
 - No pronouns.

- A class should represent one thing, all of that thing and nothing but that thing.
- You can prove cohesion by
 - Giving the class a representative name.
 - Noun (+ adjective, sometimes however also captured as attribute value).
 - Blackmail victim, robbery victim.
 - Blue car, red car.
 - **Cars** is not cohesive.
- Avoid ambiguous terms.
 - Manager, handler, processor, list, information, item, data...
- Instances vs classes of things.
- Identity ~ individuality: entities change values, but are still the same entity
 - Child/Adult: age

- Go out too far vs. going down too far.
- Investigate homonyms and synonyms.
 - Can medicine and drug be considered synonyms?
 - Do they have the same properties/characteristics/attributes/relationships?
 - Do they have a critical mass of commonalities?

- A subclass of a class represents a concept that is a “kind of” the concept that the superclass represents.
- Classes represent concepts in the domain and not the words that denote these concepts. Synonyms for the same concept do not represent different classes.
- All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality.
- If a class has only one direct subclass there may be a modeling problem or the ontology is not complete.
- If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary.
- Subclasses of a class usually (1) have additional properties that the superclass does not have, or (2) restrictions different from those of the superclass, or (3) participate in different relationships than the superclasses.
- If the concepts with different slot values become restrictions for different slots in other classes, then we should create a new class for the distinction. Otherwise, we represent the distinction in a slot value.

Examples



Step 5: Define attributes and relationships

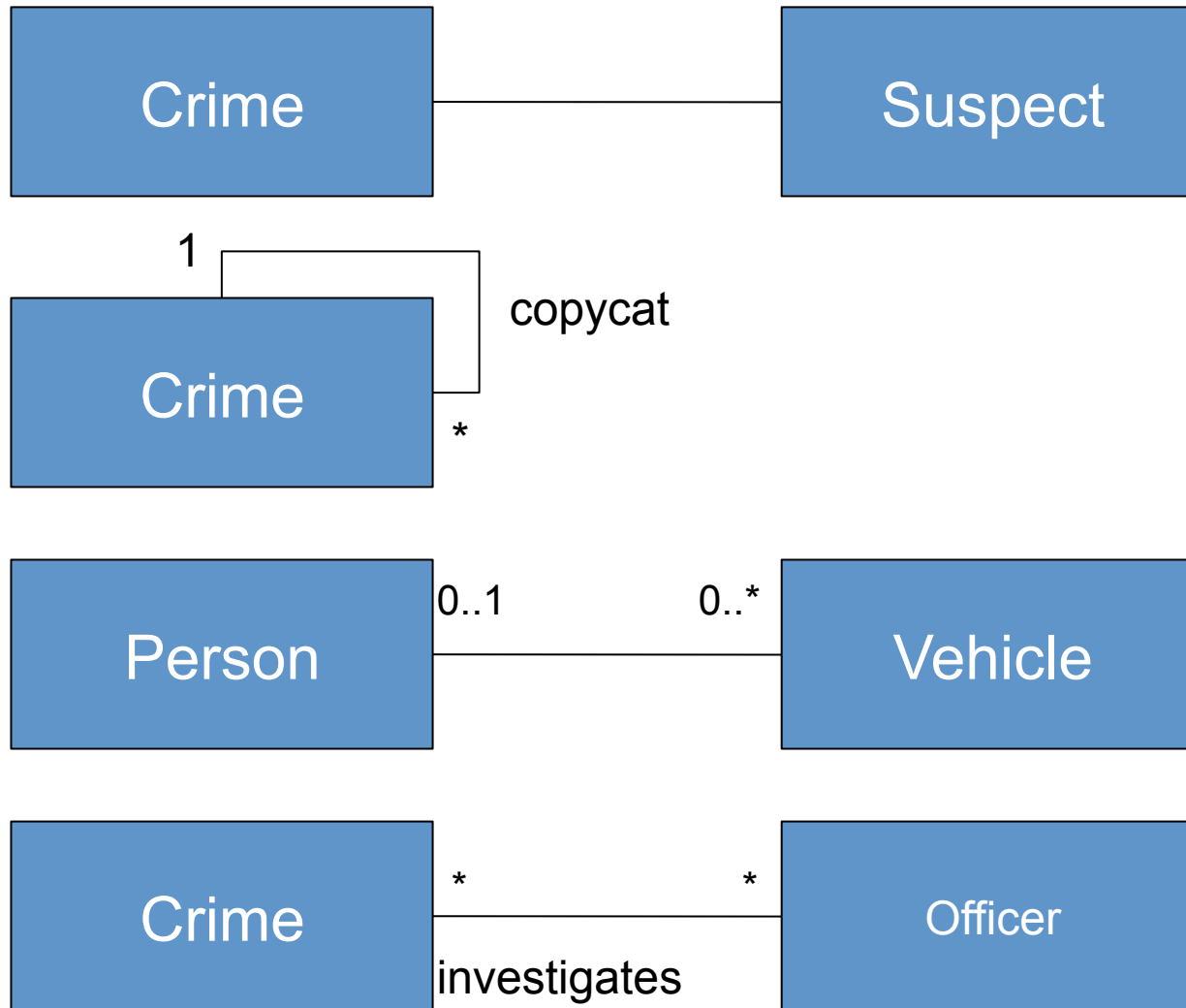
- Step 4 selected classes from the list of terms we created in Step 3.
 - Most of the remaining terms are likely to be properties of these classes.
 - For each property in the list, we must determine which class it describes.
- Types of properties
 - “intrinsic” properties
 - “extrinsic” properties
 - parts, if the object is structured (physical or abstract).
 - relationships to other individuals.
- Properties are inherited and should be attached to the most general class in the hierarchy.

- Two types of principal characteristics:
 - Measurable properties: attributes.
 - Inter-class connections: relationships.
 - Use relationships to capture something with an identity.
 - Arrest details as attribute of the suspect vs. arrest as an relationship.
 - Do we measure degrees of arrestedness or do we want to be able to distinguish between arrests?
 - Color of an image as attribute vs. class.
 - A „pointing finger“ rather than a „ruler“ indicates identity.

-
- Interview: talk to subject matter experts.
 - Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
 - Observation and reflection.
 - Nouns in „-ness“
 - Velocity-ness, job-ness, arrested-ness...
 - „How much, how many“ test.
 - If you evaluate this, then it is probably an attribute.
 - If you enumerate these, it is probably an entity.

-
- Are defined on sets of instances.
 - Properties: reflexivity, cardinality, functional, inverse-functional, discontinuous multiplicity, many-to-many, all values from, some values of, transitivity, symmetry etc.
 - Arity.

Examples

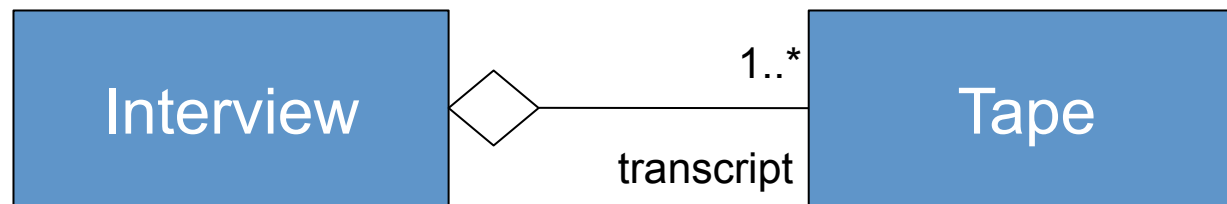


-
- Interview: talk to subject matter experts.
 - Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
 - Observation and reflection.
 - Verbs, verbal phrases and things that could have been verbs.
 - „*The butler murdered the duchess*“

- *Never draw more than three relationships between classes without going back and starting to label them.*
- The most common way to label relationships is with role names.
- Nouns, adjectives.
- Verbs: indication of time's passing.
 - Short-term, one-to-one associations should be named with present participles.
 - Longer-term, one-to-many associations should be named with past participles, or with the simple present third-person singular.



- Describes the relationship between wholes and parts.
- Existence dependence: Component is mandatory (or not) for the existence of the composiit.



- 1..* vs existence-dependence.

Step 6: Define the restrictions of the properties

- Refine the semantics of the properties
 - Cardinality
 - Domain and range
 - When defining a domain or a range for a slot, find the most general classes or class that can be respectively the domain or the range for the slots .
 - Do not define a domain and range that is overly general

Step 7: Create instances

- Define an individual instance of a class requires
 - choose a class
 - create an individual instance of that class
 - filling in the values of the properties

EXTENSIONS

-
- Hepp, M.; De Leenheer, P.; de Moor, A.; Sure, Y. (Eds.), **Ontology Management: Semantic Web, Semantic Web Services, and Business Applications**, Series: [Semantic Web and Beyond](#) , Vol. 7, 2008.
 - **Social and Collaborative Construction of Structured Knowledge, WWW 2007**, <http://www2007.org/workshop-W7.php>.
 - Neon project: <http://www.neon-project.org/web-content/>
 - INSEMTIVES: <http://www.insemtives.eu>
 - Vocamps: http://vocamp.org/wiki/Main_Page

LARGE EXAMPLE

-
- A platform offering services and infrastructure for:
 - (semi-) automatic semantic annotation and
 - ontology population
 - semantic indexing and retrieval of content
 - query and navigation over the formal knowledge
 - Based on Information Extraction technology

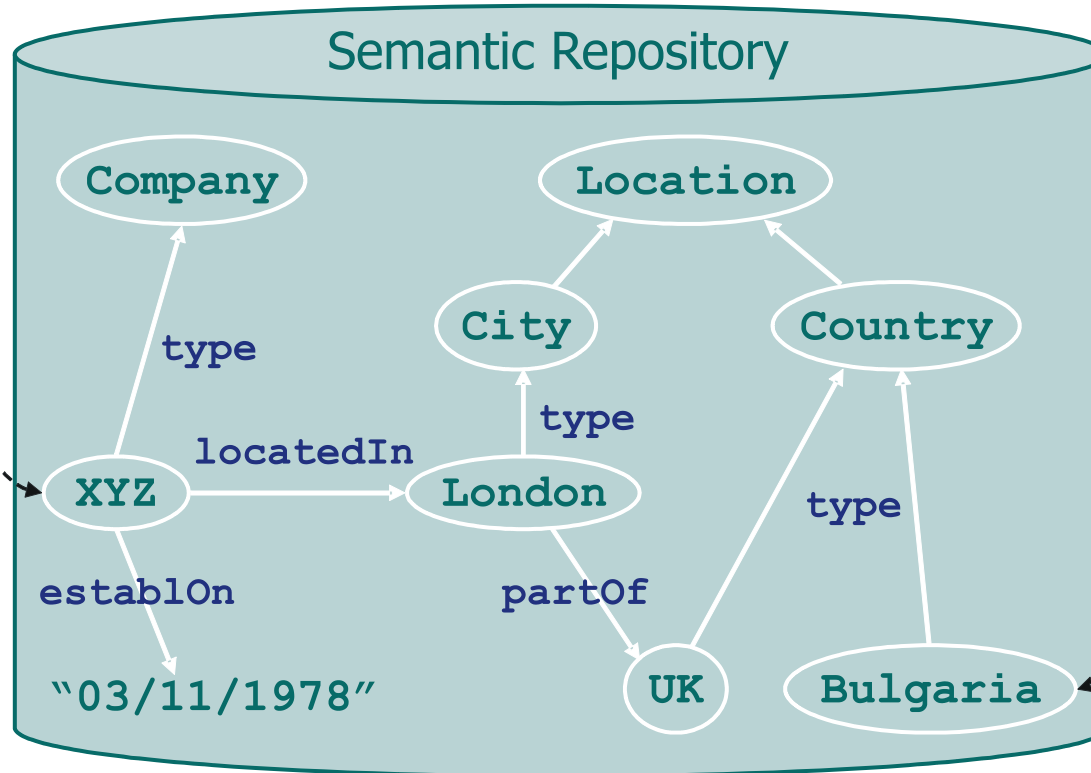
The KIM Platform includes:

- Ontologies (PROTON + KIMSO + KIMLO) and KIM World KB
- KIM Server – with a set of APIs for remote access and integration
- Front-ends: Web-UI and plug-in for Internet Explorer.

-
- Aim: to arm Semantic Web applications
 - by providing a metadata generation technology
 - in a standard, consistent, and scalable framework

What KIM does Semantic Annotation

XYZ announced profits in Q3, planning to build a \$120M plant in Bulgaria, and more and more and more and more text...



Slides by Ontotext

Rationale:

- The ontology is encoded in OWL Lite and RDF.
- provide common knowledge about world entities;
- KIM bets on scale and avoids heavy semantics;
minimum modeling of common-sense, almost no axioms;
- The ontology is encoded in OWL Lite and RDF.
- In addition, a number of rules (generative axioms) are defined, e.g.:
 $\langle X, \text{locatedIn}, Y \rangle$ and $\langle Y, \text{subRegionOf}, Z \rangle \Rightarrow$
 $\langle X, \text{locatedIn}, Z \rangle$
- Axioms of this sort are supported by OWLIM and they provide a consistent mechanism for “custom” extensions to the OWL or RDF(S) semantics with respect to a particular ontology

- **Name.** PROTON is an acronym for **Proto Ontology**
 - ex-names: BULO (basic upper-level ontology), GO (generic ontology);
 - “proto” – used in the sense of “primary”, “beginning”, “giving rise to”, vs. “first in time” or “oldest”;
 - connotations: positive, fundamental, elemental, “in favour of”, even romantic (like a science-fiction novel from the 60-ies) 😊
- **Intended usage.** A Basic Upper-Level Ontology like PROTON - **used for**:
 - ontology population
 - knowledge modelling and integration strategy of a KM environment;
 - generation of domain, application, and other ontologies.

- **Design principles:**
 1. domain-independence;
 2. light-weight logical definitions;
 3. Compliance with popular metadata standards;
 4. good coverage of concrete and/or named entities (i.e. people, organizations, numbers);
 5. no specific support for general concepts (such as “apple”, “love”, “walk”), however the design allows for such extensions

- PROTON defines about
250 classes and 100 properties
- Providing coverage of most of the upper-level concepts necessary for semantic annotation, indexing, and retrieval
- A **modular** architecture, allowing for great flexibility of usage and extension:
 - SYSTEM module - contains a few meta-level primitives (6 classes and 7 properties); introduces the notion of 'entity', which can have aliases;
 - TOP module - the highest, most general, conceptual level, consisting of about 20 classes;
 - UPPER module - over 200 general classes of entities, which often appear in multiple domains.

- The current version of the ontology is encoded in OWL Lite.
- A few custom entilement rules (axioms) are also defined for usage in tools that support them, for instance:

Premise:

`<xxx, protont:roleHolder, yyy>`

`<xxx, protont:roleIn, zzz>`

`<yyy, rdf:type, protont:Agent>`

Consequent:

`<yyy, protont:involvedIn, zzz>`

- Axioms of this sort are interpreted by OWLIM
- PROTON is portable to any OWL(Lite)-compliant tool.
- PROTON can be used without such axioms either.

A quasi-exhaustive coverage of the most popular entities in the world

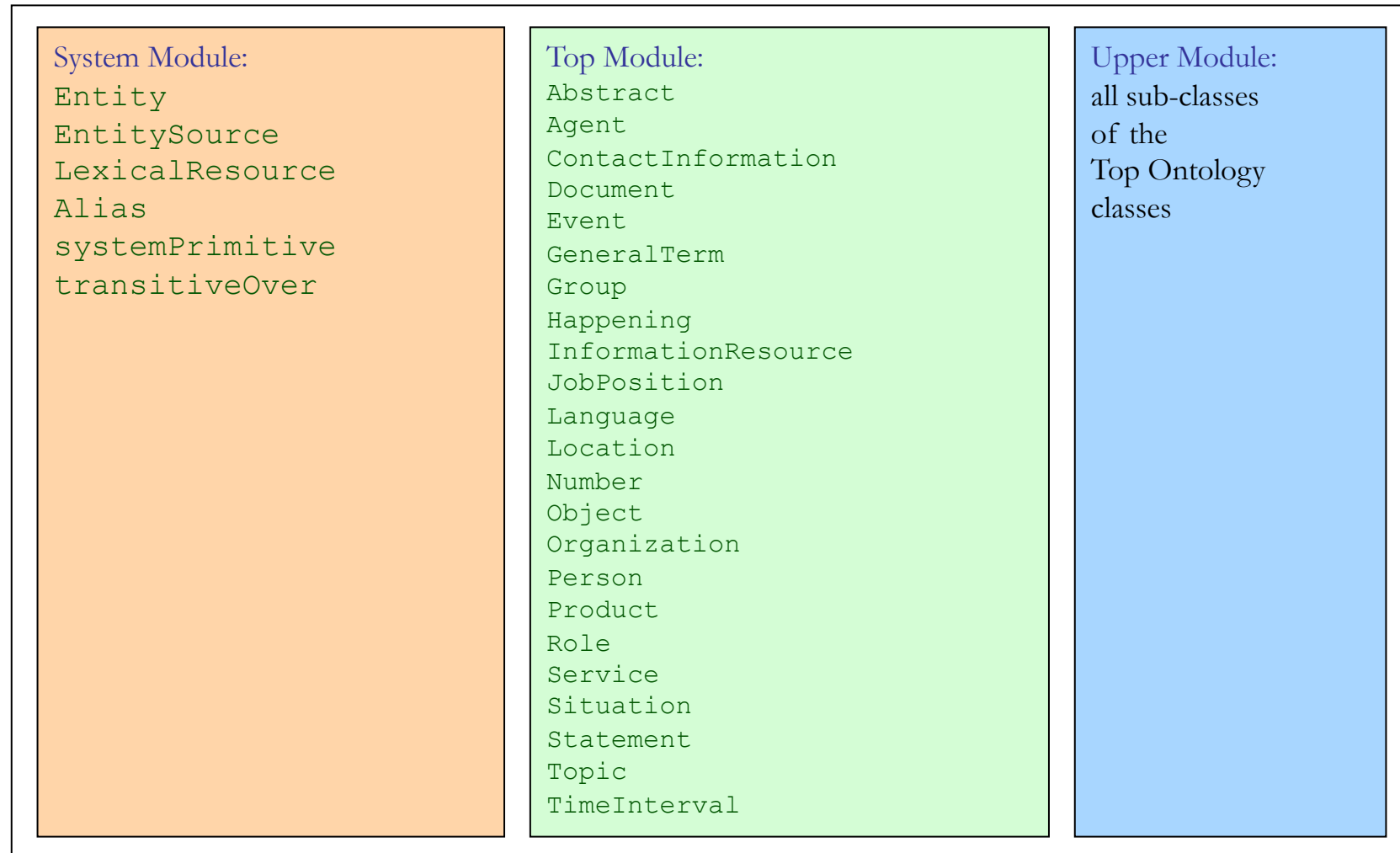
...

- What a person is expected to have heard about that is beyond the horizons of his country, profession, and hobbies.
- Entities of general importance ... like the ones that appear in the news ...

KIM “knows”:

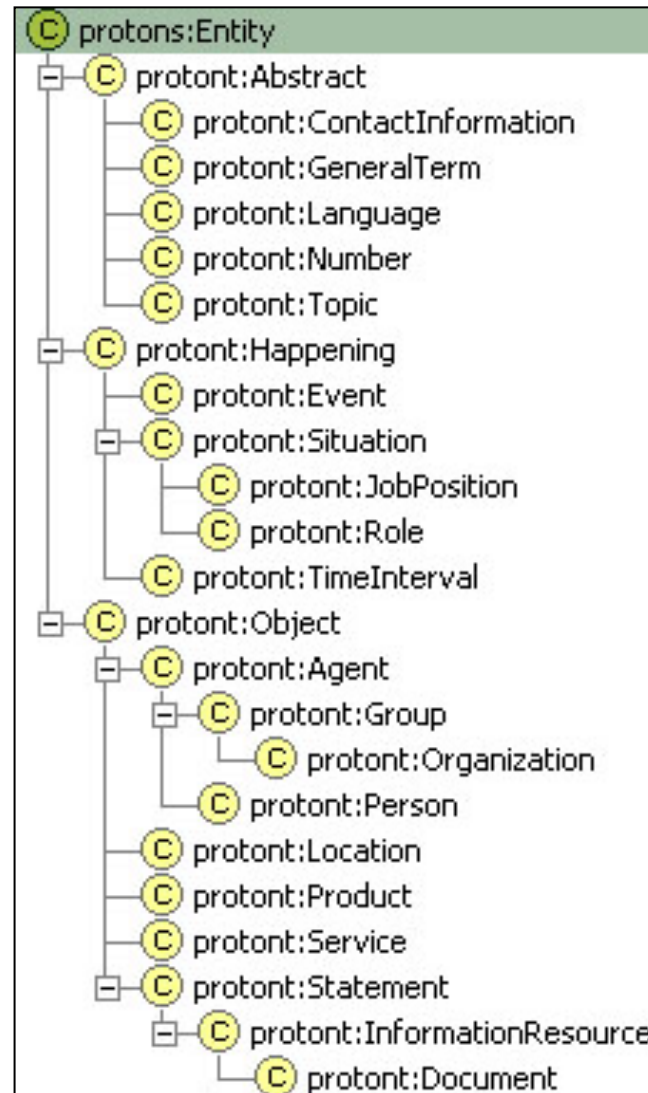
- Locations: mountains, cities, roads, etc.
- Organizations, all important sorts of: business, international, political, government, sport, academic...
- Specific people, etc.

The Architecture of PROTON (I)



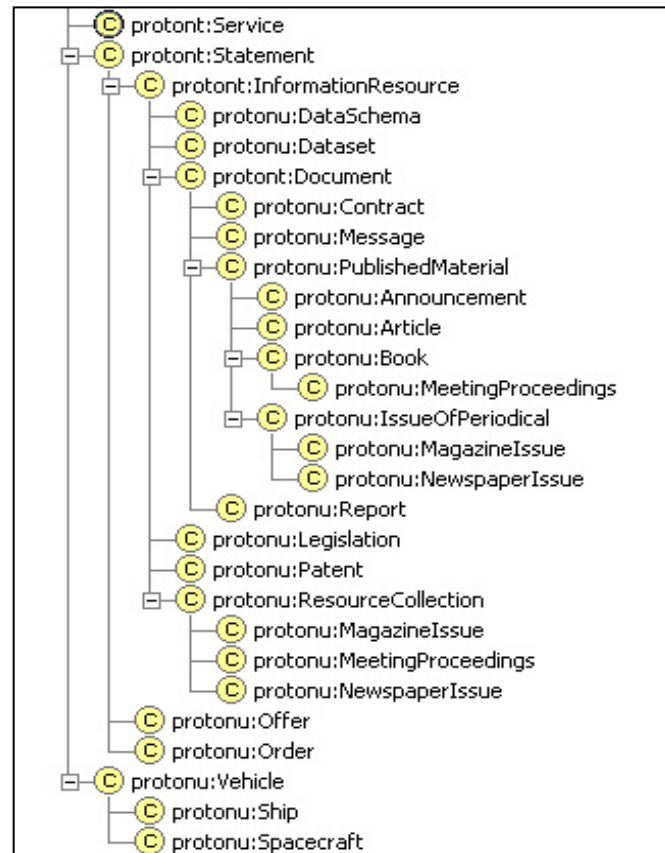
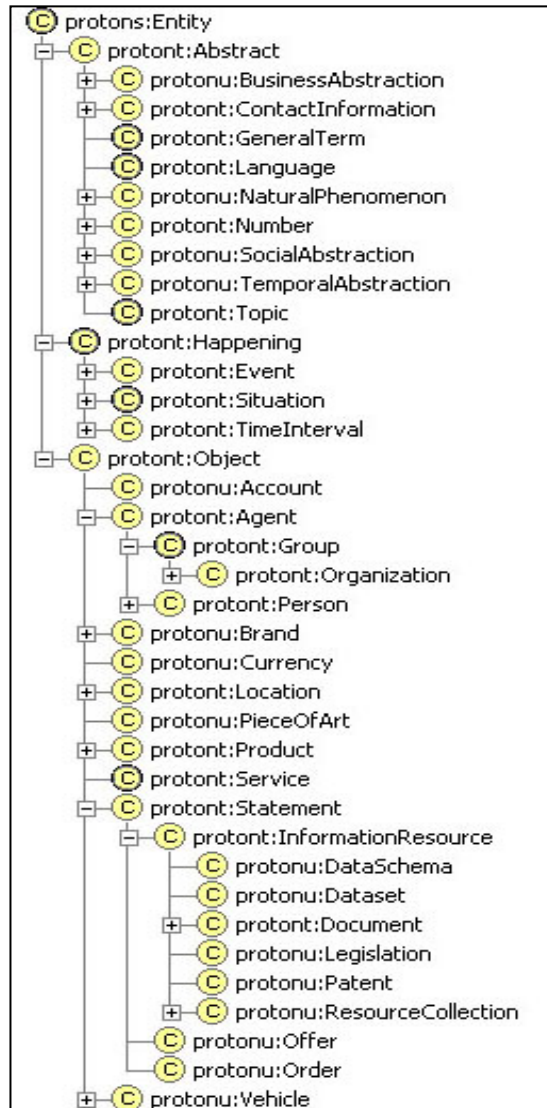
- 3 modules (layers)
- Diacritical modular architecture
- Allows for easy integration, extension, and remodelling
- System module
 - subservient, operative, may need to be hard-coded
 - a level that can be also classified as an “application ontology”
 - defines some notions and concepts of a technical nature that are substantial for the operation of any ontology-based software, such as semantic annotation and knowledge access tools
- Top module – most basic entity types
- Upper module
 - sub-classes of top classes
 - some branches include much more specialized entities
 - borders with some domain ontology characteristics

protont: Top Module In Brief (I)

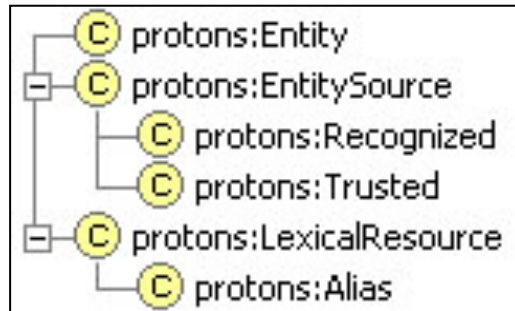


- The most essential ingredient in the PROTON pie
- The most general classes, required by the SEKT case studies
- Referred to via the "protont:" prefix
- Hierarchy - at the upper-most layer: basic philosophically-reasoned distinctions between entity types:
 - **protont:Object** – existing entities, as agents, locations, vehicles;
 - **protont:Happening** – events and situations;
 - **protont:Abstract** – abstractions that are neither objects nor happenings;
 - respective DOLCE equivalents: **Endurant, Perdurant, Abstract**.
- Further specialized by substantially real entity types of general importance: persons, locations, numbers, time, money, roles, employment (job) positions, commercial, government, and other organizations, etc.
- Characteristic properties (attributes and relations) for featured entity types

protonu: Upper Module In Brief (I)



- Represents the sub-class branches under Top module classes
- Defines much more specific classes, e.g.:
 - **protonu:Mountain**, as a specific type of **protont:Location**;
 - **protonu:ResourceCollection** as a sub-class of **protont:InformationResource**
- Over 200 general classes of entities
- Upper module classes often appear in multiple domains (e.g. various sorts of organizations, a comprehensive range of locations, etc.)
- Referred to via the "protonu:" prefix
- In the future – can be considered to be
 - “cleaned up” from entities that are too domain-specific;
 - split down further into sub-modules if needed (e.g. the **protont:Location** branch).



System module classes

- P protons:description
- P protons:generatedBy
- P protons:hasAlias
- P protons:hasMainAlias
- P protons:laconicDescription
- P owl:versionInfo
- P rdfs:comment
- P rdfs:label
- P protons:systemPrimitive
- P protons:transitiveOver

System module properties

- Provides a sort of high-level system- or meta-primitives, which are likely to be accepted and even hard-coded in particular tools that may use PROTON.
- The only component in PROTON that is not to be changed for the purposes of ontology extension.

- Referred to via the "protons:" prefix
- Classes:
 - **protons:Entity**
 - represents the root of the ontology
 - the super-class of the Top module (the other branches of System could be considered auxiliary or "system" ones)
 - **protons:EntitySource**
 - key role – limited support for recording of the provenance of the instance data;
 - its instances are used to separate the trusted (pre-populated) information in the KB, from the one that is extracted automatically;
 - such a distinction is indicated by the **protons:generatedBy** property of the specific entity;
 - each instance of **protons:Entity** could be linked to an instance of **protons:EntitySource** via the **protons:generatedBy** property;


SUMMARY

-
- *An ontology is a formal, explicit specification of a shared conceptualization.*
 - Ontologies should be community contracts, i.e. Reflect the view of the user community.
 - There are many different methodologies for building ontologies.
 - The choice depends on the application and nature of the ontology.
 - Varying degrees of expressivity.

- Mandatory reading:
 - [Natalya F. Noy](#) and [Deborah L. McGuinness](#). "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report [KSL-01-05](#) and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
 - Gomez-Perez et al., Ontological Engineering, Springer, 2004.
 - S. Decker, M. Erdmann, D. Fensel, and R. Studer: Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Database Semantics, Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999.

- Further reading:
 - S. Braun, A. Schmidt, A. Walter, G. Nagypal, and V. Zacharias. Ontology maturing: A collaborative web 2.0 approach to ontology engineering, Collaboration Workshop at WWW'07, May 8 2007.
 - C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. Communications of the ACM, 45(2):4247, 2002.
 - A.D. Nicola, R. Navigli, and M. Missiko. Building an eprocurement ontology with upon methodology. In Proceedings of the 15th e-Challenges Conferences, Lublijana, Slovenia, 2005.
 - K. Kotis and G.A. Vouros. Human-centered ontology engineering: The hcome methodology. Knowledge and Information Systems, 10(1):109131, jul 2005.
 - <http://www.ontotext.com/kim/>
 - <http://proton.semanticweb.org/>

- Wikipedia links:
 - <http://en.wikipedia.org/wiki/Ontology>
 - http://en.wikipedia.org/wiki/Ontology_engineering

#	Title
1	Introduction
2	Semantic Web Architecture
3	Resource Description Framework (RDF)
4	Web of data
5	Generating Semantic Annotations
6	Storage and Querying
7	Web Ontology Language (OWL)
8	Rule Interchange Format (RIF)
9	Reasoning on the Web
10	Ontologies
 11	Social Semantic Web
12	Semantic Web Services
13	Tools
14	Applications

Questions?

