Ontology Development:

Methodologies, Criteria, Evaluation

G. Falquet

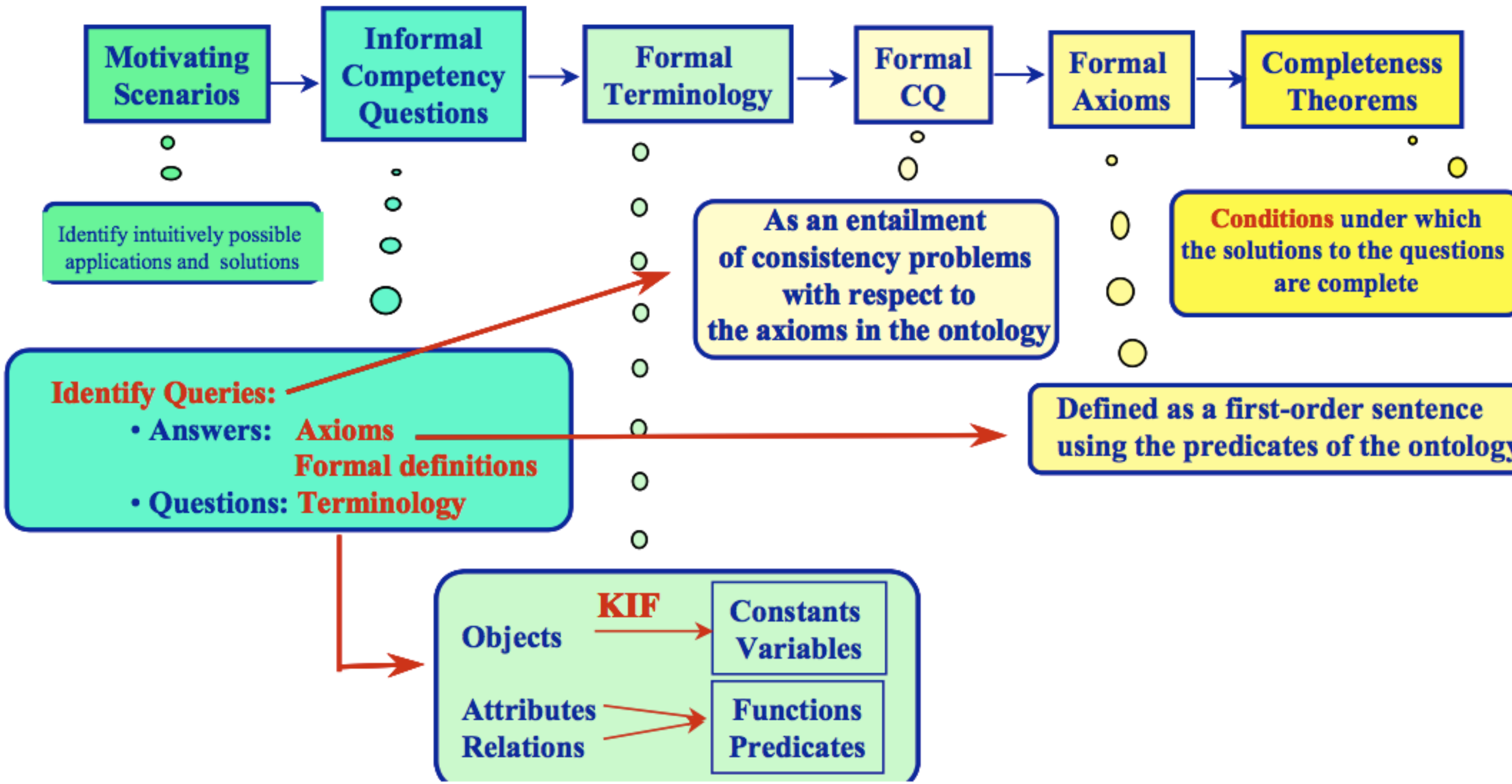
Uschold and King's method

1. *Identify purpose*. It is important to be clear why the ontology is being built and what its intended uses are.
2. *Building the ontology*
   1. *capture*
   2. *coding*
   3. *integration*
3. *Evaluation*, "... with respect to a frame of reference ... The frame of reference may be requirements specifications, competency questions, and/or the real world".
4. *Documentation* recommends that guidelines be established for documenting ontologies, possibly differing according to the type and purpose of the ontology.

U&K – Ontology capture

- Identification of the key concepts and relationships in the domain of interest, that is, scoping.
  - center on the concepts as such, rather than the words representing them.
- Production of precise unambiguous text definitions for concepts and relationships. Identification of terms to refer to such concepts and relationships.

Grüninger and Fox's methodology



G&F. 1. Capture of motivating scenarios

- Story problems or examples which are not adequately addressed by existing ontologies.
- A motivating scenario also provides a set of intuitively possible solutions to the scenario problems.
- These solutions provide an informal intended semantics for the objects and relations that will later be included in the ontology.

G&F. 2. Formulation of informal competency questions

- Based on the scenarios obtained in the preceding step
- An ontology must be able to
  - represent these questions using its terminology
  - characterize the answers to these questions using the axioms and definitions.
- competency questions are stratified (composition and decomposition)
- serve as constraints rather than determining a particular design
- used to evaluate the ontological commitments that have been made to see whether the ontology meets the requirements.

Ontology 101 (Noy, McGuiness)

- Step 1. Determine the domain and scope of the ontology
  - Define Competency questions
- Step 2. Consider reusing existing ontologies
- Step 3. Enumerate important terms in the ontology
- Step 4. Define the classes and the class hierarchy
- Step 5. Define the properties of classes—slots
- Step 6. Define the facets of the slots (axioms on properties)
- Step 7. Create instances

O. 101: In Step 1. Determine the domain and scope of the ontology

Define **competency questions**

- User-oriented interrogatives that allow us to scope our ontology.
- Serve in the evaluation: Does the ontology contain enough information to answer these types of questions?

https://tishchungoora.medium.com/ontology-competency-questions-3d213eb08d33

- What characteristics belong to a ballpoint pen?
- Which types of components are common across all ballpoint pens?
- What are the common brands of ballpoint pens?
- Is a kind of ballpoint pen a specialisation of another?
- Can the definition of common types of ballpoint pens be used as a basis to define custom ballpoint pens?

https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

- Which wine characteristics should I consider when choosing a wine?
- Is Bordeaux a red or white wine?
- Does Cabernet Sauvignon go well with seafood?
- What is the best choice of wine for grilled meat?
- Which characteristics of a wine affect its appropriateness for a dish?
- Does a bouquet or body of a specific wine change with vintage year?
- What were good vintages for Napa Zinfandel?

Methontology [adapted to OWL2]

- Developed within the Ontology Engineering Group at the Polytechnic University of Madrid.
- It is rooted in the activities identified by the software development process proposed by the IEEE organization and other knowledge engineering methodologies.
- Proposal for the construction of ontologies by the Foundation for Intelligent Physical Agents (FIPA).
- Modelling primitives:
  - concept [class], instance [individual], attribute [datatype property], relation [object property], constant, class attribute, axiom [OWL2 axiom], rule [SWRL rule]

Steps

1. Build a glossary of terms
2. Build a concept taxonomy
3. Build a binary relationship diagram
4. Build a concept dictionary
5. Describe every relationship, attribute, …
6. Define Axioms
7. Define Rules

T1. Build a glossary of terms

- It includes all the relevant terms of the domain
  - concepts
  - instances
  - attributes
  - relationships between concepts,
  - etc.
- their descriptions in natural language,
- and their synonyms and acronyms.

Name	Synonym	Acronym	Description	Type	OWL Type
age of majority	--	--	The age of majority in Spain is 18	Constant	??
court	tribunal	--	Refers to the entity that represents a judicial court	Concept	Class
birthdate	--	--	Date of birth of a person	Instance attribute	Datatype property
defendant	accused	--	The person sued of accused in a court	Relation	Object property

Instance or Class?

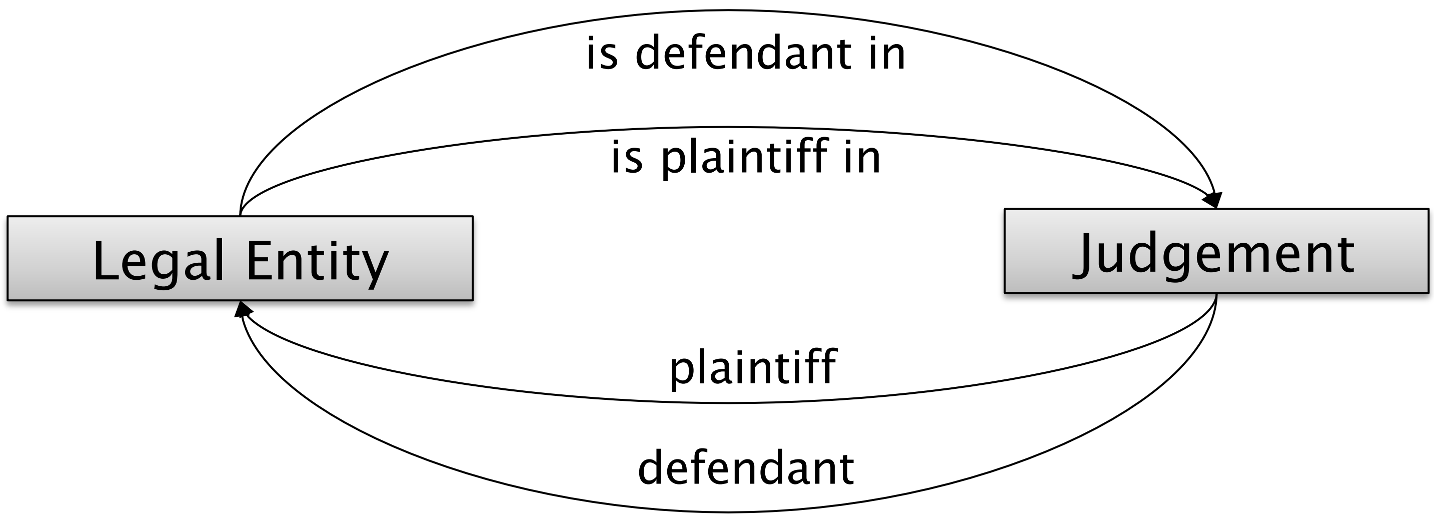
- Individual instances are the most specific concepts represented in a knowledge base.
- Depends on the ontology objectives/requirements
- Beware of the names!
  - Bordeaux subclass of France
    - YES: for wine classes (a Bordeaux is a French wine)
    - NO: for geographic entities

T2. Build a concept taxonomy

- Select the concepts from the glossary
- Arrange them in a subconcept hierarchy

T3. Build a binary relationship diagram

- Include relations between concept
- and their inverse



T4. Build a concept dictionary

- Specify the properties that describe each concept
  - concept [class] name
  - instances [individuals] (examples)
  - class attributes
  - instance attributes [datatype properties]
  - relationships [object properties]

Concept name [Classes]	Instances	Instance attributes [Datatype properties]	Relations [Object Properties]
court	Supreme court of NC Court of Appeals of NC Supreme court of the UK	number of members location jurisdiction	holds
company	--	name	--
trial	--	start date end date	plaintiff defendant held in
legal entity	--	--	is plaintiff in is defendant in

Describe every relationship

- name
- inverse
- origin (domain) concept
- destination (range) concept
- cardinality

Detailed description of

- attributes
  - type of values
  - cardinality
- constants
- instances

Axioms

- logical formulae that are always true (invariants)
  - act as constraints on instances
  - described in natural language and formally

"The same person cannot be the defendant and the plaintiff in the same trial"

$$\neg\ (\exists x \exists y\ (\mathrm{person}(x) \wedge \mathrm{trial}(y) \wedge \mathrm{defendant}(x, y) \wedge \mathrm{plaintiff}(x, y)))$$

Rules

- to infer knowledge
  - if <condition> then <consequence or action>

"A trial where the defendant is a minor who is over 14 years old is held in a juvenile court"

IF

minor(?X), trial(?Z) , tribunal(?W) , age(?X, ?Y) , ?Y > 14 ,
defendant(?X, ?Z) , held_in(?Z, ?W)

THEN

youth_court(?W)

Gruber’s criteria

- Clarity
- Coherence
- Extendibility
- Minimal encoding bias
- Minimal ontological commitment

(Gruber) **1. Clarity:**

- effectively communicate the intended meaning of defined terms.
  - definitions should be *objective.*
  - the definition should be independent of social or computational context.
- formalism: When a definition can be stated in logical axioms, it should be.
- (where possible) complete definition (necessary **and** sufficient conditions) is preferred over a partial definition (only necessary or sufficient conditions).
- definitions should be documented with natural language.

(Gruber) **2. Coherence:**

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

(Gruber) **3. Extendibility:**

designed to anticipate the uses of the shared vocabulary.

- offer a conceptual foundation for a range of anticipated tasks
- the representation should be crafted so that one can extend and specialize the ontology *monotonically*.
  - possibility to define new terms for special uses without revision of the existing definitions.

(Gruber) 4. Minimal encoding bias

- encoding bias results when a representation choices are made purely for the convenience of notation or implementation.
- encoding bias should be minimized, because knowledge-sharing agents may be implemented in different representation systems and styles of representation.
  - e.g. DOLCE exists in FOL and in DL

(Gruber) 5. Minimal ontological commitment

About **ontological commitment**

- an agent commits to an ontology if its observable actions are consistent with the definitions in the ontology.
- The agents sharing a vocabulary need not share a knowledge base
  - each knows things the other does not
  - in DL: common TBox, different ABoxes

(Gruber) **5. Minimal ontological commitment:**

- require the minimal ontological commitment sufficient to support the intended knowledge sharing activities.
- make as few claims as possible about the world being modeled,
  - allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed.
- specify the weakest theory (allowing the most models) and define only those terms that are essential to the communication of knowledge consistent with that theory.

Ontology Evaluation

- **Errors in developing taxonomies**
  - Gómez-Pérez presents a set of possible errors that can be made by ontologists when building taxonomic knowledge into an ontology or by Knowledge Engineers when building KBs under a frame-based approach.
- **OntoClean**
  - OntoClean has been elaborated by the Ontology Group of the LADSEB-CNR in Padova (Italy). It is a method to clean taxonomies according to notions such as: *rigidity*, *identity* and *unity* [Gangemi et al.; 2001]:

Errors in developing taxonomies

- *Subclass partition with common instances*.
- *Subclass partition with common classes*
- *Exhaustive subclass partition with common instances*..
- *Exhaustive subclass partition with common classes*
- *Exhaustive subclass partition with external instances*
- *Subclass partition omission*
- *Exhaustive subclass partition omission.*
- *Redundancies of subclass-of relations*
- *Redundancies of instance-of relations*

Anti-patterns

error-prone modeling decisions ⇒

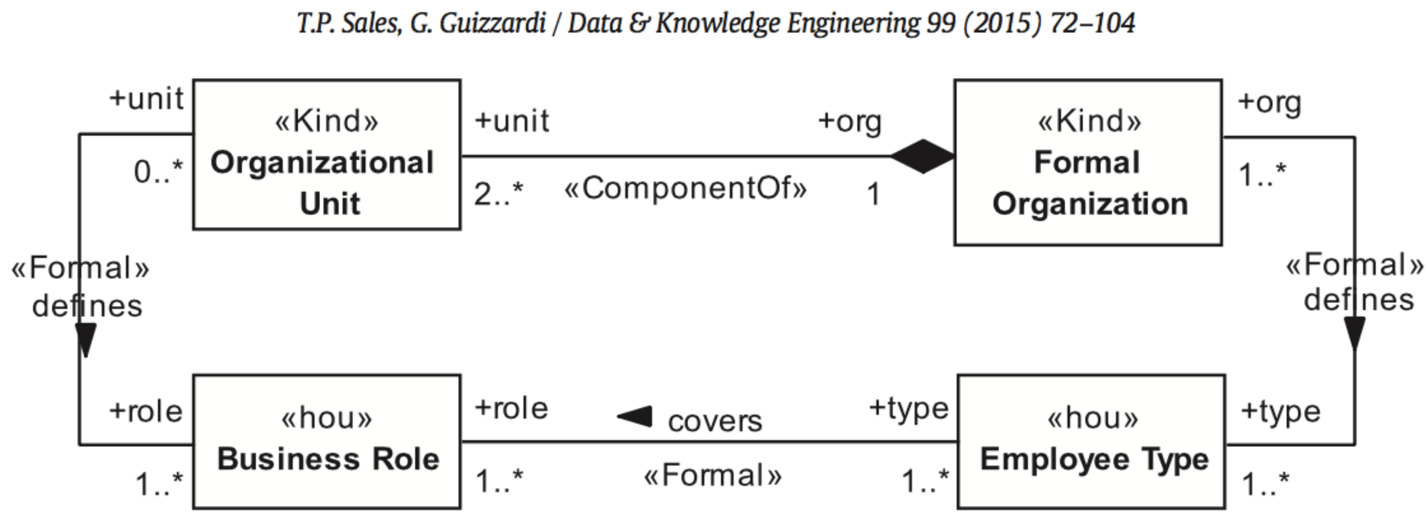
- creation of models that fail to exclude unintended model instances (representing unintended state of affairs)
- forbid intended ones (representing intended states of affairs).

Tiago Prince Sales , Giancarlo Guizzardi

Ontological anti-patterns: empirically uncovered error-prone structures in ontology-driven conceptual models

doi:10.1016/j.datak.2015.06.004

Example: AssCyc anti-pattern



- cycles allow for two very characteristic instantiations:
  - one in which there are cycles at the instance level,
  - and another one in which there is none.
- empirical studies strongly ⇒ only one should be allowed.

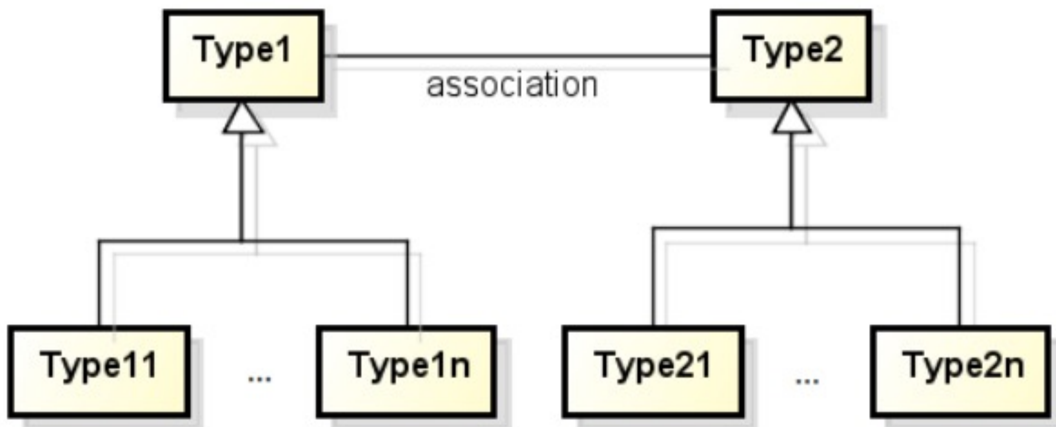
Refactoring

- first, to enforce the open cycle instantiation scenario at instance level through the specification of an OCL invariant;
- second, is an analogous solution to forbid instance level cycles;
- third, we set one of the associations as derived and its corresponding derivation OCL rule is specified.

Binary Relation Between Overlapping Types

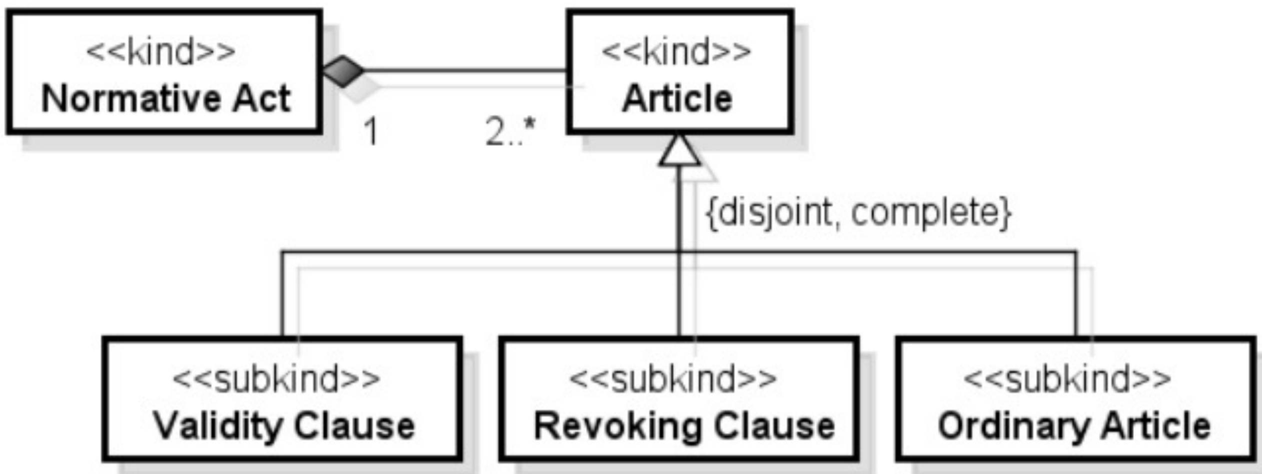
- when an association connects two types that constitute an overlapping set.
- it means that the same individual may eventually instantiate both ends of the relationship.

Example: Imprecise Abstraction Anti-Pattern



- the association is too generic
- good to simplify the model but ...
- does not reflect some cardinality constraints on specific subtypes

Example



- does not represent the fact that a Normative Act must have exactly one Validity Clause

OntoClean

Rigidity

a property is *rigid* (+R) if and only if it is necessarily essential to all its instances.

- person is rigid, since every person is essentially such,
- student is anti-rigid, since every student can possibly be a non-student a few years later.

Rule: anti-rigid properties cannot subsume rigid properties

Identity

A property *carries an identity criterion* (IC) (+I) if and only if all its instances can be (re)identified by means of a suitable "sameness" relation.

Example:

- two durations of the same length are the same duration.
- two intervals occurring at the same time are the same, but two intervals occurring at different times, are different, even if they have the same duration.
- $\Rightarrow$ a contradiction if all instances of time interval are also instances of time duration $\Rightarrow$ Interval is not a subclass of Duration

Dependency

- An individual x is constantly dependent on y if and only if, at any time, x cannot be present unless y is fully present, and y is not part of x.
- For example, a hole in a wall is constantly dependent on the wall.

Unity

- We can say that an individual is a *whole* if and only if it is made by a set of parts unified by a unifying relation *R*. All the instances of *P* are wholes under *R*. e.g. a Company

Anti-Unity

- A property carries *anti-unity* (~U) if all its instances can possibly be non-wholes (mere sums). Properties that refer to amounts of matter, like *gold*, *water*, etc., are good examples of anti-unity.
  - An instance of this class might be some amount of the material

Unity and Anti-unity are inherited in the class hierarchy