Inteligencia Artificial
Representación del conocimiento

Primavera 2007

profesor: Luigi Ceccaroni

Historia

- Las categorías son los bloques de construcción primarios para cualquier esquema de representación del conocimiento.
- Las **redes semánticas** proporcionan:
  - una ayuda gráfica para visualizar una base de conocimiento
  - algoritmos eficientes para inferir propiedades de un objeto en base a su pertenencia a una categoría.
- Charles S. Peirce (1909) desarrolló los **grafos existenciales** como el primer formalismo de redes semánticas usando lógica moderna.

Historia

- Ross Quillian (1961) inició el trabajo con las redes semánticas dentro del campo de la IA.
- Un artículo influyente de Marvin Minsky (1975) presentó una versión de las redes semánticas: los **marcos**.
- Un marco era una representación de un objeto o categoría o concepto, con atributos y relaciones con otros objetos o categorías o conceptos.
- El artículo fue criticado por ser un conjunto de ideas recicladas desarrolladas en el campo de la programación orientada a objetos, como la herencia y el uso de valores por defecto.

Historia

- En los años ‘90, en el campo de la IA, se adoptó el termino **ontología** para los esquemas de representación del conocimiento basados en redes semánticas o marcos.
- Una ontología es una especificación formal y explícita de una conceptualización compartida, que puede ser leída por un ordenador (Gruber, 1993; Borst, 1997; Studer et al., 1998; Ceccaroni, 2001).

Redes semánticas

- Una red semántica es un grafo donde:
  - los nodos representan conceptos
  - los arcos (dirigidos) representan relaciones entre conceptos
- Mecanismos de razonamiento específicos permiten responder a preguntas sobre la representación:
  - ¿Están relacionados dos conceptos?
  - ¿Que relaciona dos conceptos?
  - ¿Cuál es el concepto mas cercano que relaciona dos conceptos?

Marcos

- A los marcos se asocia normalmente una parte procedimental.
- Las relaciones y atributos, y no solo las categorías, tienen una estructura que permite describir su semántica.
- Ejemplo:
  - Arteria
    - superclases: Vaso sanguíneo
    - pared: Muscular
    - forma:

Marcos

- Un marco está generalmente dividido en:
  - una parte **declarativa** (atributos o *slots*)
  - una **procedimental** (métodos o *demons*)
- La parte procedimental permite obtener más información o hacer cálculos sobre sus características o las relaciones que pueda tener con otros marcos.
- La descripción de los atributos también está estructurada: un atributo puede tener propiedades (*facets*).
- En el caso más general se pueden tener taxonomías de atributos.

Marcos

- Las **relaciones** poseen una descripción formal que establece su semántica y su funcionamiento.
- Dividimos las relaciones en dos simples clases:
  - **taxonómicas**:
    - enlace ES-UN (subclase/clase)
    - enlace INSTANCIA-DE (instancia/clase)
  - **no taxonómicas**

Marcos

- Los **atributos** poseen un conjunto de propiedades que permiten establecer su semántica:
  - dominio
  - rango
  - cardinalidad
  - valor por defecto
  - métodos
  - ...
- Permiten definir procedimientos de manera que se realicen cálculos bajo ciertos eventos (a través de los métodos).

Marcos

- Los **métodos** pueden ser:
  - *if-needed* (se activan al consultar el atributo);
  - *if-added* (se activan al asignar valor al atributo);
  - *if-removed* (se activan al borrar el valor del atributo);
  - *if- modified* (se activan al modificar el valor del atributo).
- Se puede declarar como el mecanismo de herencia afecta a los atributos.

Marcos: atributos

- Etiqueta
  - nombre:
  - valor:
  - **dominio**: lista de marcos donde puede aparecer
  - **rango** (tipo de valores que admite): lista, clase…
  - cardinalidad máxima:
  - **cardinalidad mínima** (si es ≥ 1, el atributo es obligatorio):
  - valor-por-defecto (a usar si no hay valor):
  - función para calcular valor:
  - métodos (funciones con activación condicionada):
  - condiciones de herencia (atributo + valor): sí/no (por defecto: relaciones taxonómicas = sí; otras = no)

- Para acceder al valor de un atributo se usa la sintaxis:

  <nombre marco>.<nombre atributo> (valor o lista de valores)

Marcos: ejemplo de atributo

- owl:edad
  - nombre: Edad
  - valor:
  - **dominio** (lista de marcos donde puede aparecer): clase Persona
  - **rango** (tipo de valores que admite): entero [0..140]; joven/viejo
  - cardinalidad máxima: 1
  - **cardinalidad mínima** (si es ≥ 1, el atributo es obligatorio): 0
  - valor-por-defecto (a usar si no hay valor):
  - función para calcular valor:
  - métodos (funciones con activación condicionada):
  - condiciones de herencia (atributo + valor): sí/no (por defecto: relaciones taxonómicas = sí; otras = no)
- Para acceder al valor del atributo se usa la sintaxis:

  Persona.edad

Marcos: métodos

- Son acciones o funciones que permiten obtener información sobre el mismo marco u otros marcos.
- Los métodos pueden invocarse desde marcos abstractos (clases) o marcos concretos (instancias).
- Pueden ser heredables (se permite invocarlos en los descendientes) o no heredables (exclusivos del marco donde están definidos).
- A veces, pueden ser invocados con parámetros.

- Ejemplo de método *if-modified*:
  - Si *Deunan.edad* tenía valor 28 y se modifica a 32, se activa un método que cambia el valor del atributo *Deunan.ganas-de-casarse* de 1 a 5.

Marcos: relaciones

- Permiten conectar los marcos entre sí.
- Se define su semántica mediante un conjunto de propiedades:
  - dominio
  - rango
  - cardinalidad
  - inversa
  - transitividad
  - composición …
- Se pueden establecer **métodos** que tienen efecto ante ciertos eventos:
  - *if-added*: si se establece la relación entre instancias
  - *if-removed*: si se elimina la relación entre instancias
- Se puede establecer el comportamiento de la relación respecto al mecanismo de herencia (que atributos permite heredar).

Marcos: relaciones

- Etiqueta
  - nombre:
  - **dominio**: lista de marcos
  - **rango**: lista de marcos
  - **cardinalidad**: 1 o N
  - **inversa**: <nombre> (cardinalidad: 1 o N)
  - transitiva: sí/**no** (por defecto es *no*)
  - compuesta: **no** / descripción de la composición
  - métodos: {*if-added* / *if-removed*} <*nombre.acción*>
  - condiciones de herencia: lista de atributos (por defecto: lista vacía)

- Las acciones asociadas a los métodos no tienen parámetros.
- La expresión **<nombre marco>.<nombre relación>** devuelve el marco (si la cardinalidad es 1) o la lista de marcos (si es N) con los que está conectado el **<nombre marco>** a través de **<nombre relación>**.

Marcos: relaciones

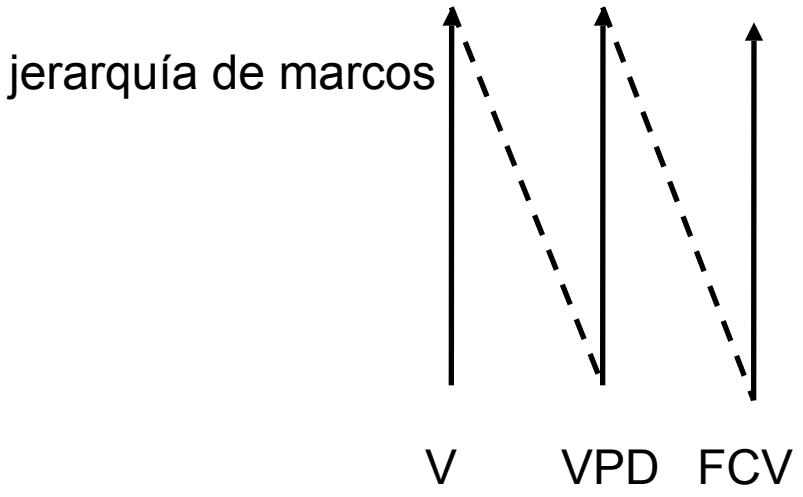
- Para consultar la cardinalidad se usa la función **card(<nombre marco>.<nombre relación>)** que devuelve un entero.
- Relaciones predefinidas:
  - enlace ES-UN (inversa: tiene-por-subclase)
    - Transitiva
  - enlace INSTANCIA-DE (inversa: tiene-por-instancia)
    - Se puede obtener por composición: INSTANCIA-DE ⊕ ES-UN
- Funciones bulianas que quedan definidas:
  - **atributo?(<marco>)**: *cierto* si <marco> posee este atributo.
  - **relación?(<marco>)**: *cierto* si <marco> está conectado con algún otro marco a través de la relación indicada por la función.
  - **relación?(<marco-o>,<marco-d>)**: *cierto* si existe una conexión entre <marco-o> y <marco-d> etiquetada con la relación indicada por la función.

Marcos: herencia

- La herencia permite obtener en un marco el valor o los valores de un atributo a través de otro marco con el que esta relacionado.
- En el caso de las **relaciones taxonómicas** (caso más común) la herencia (de atributos y valores) se da por defecto.
- En el **resto de las relaciones** se ha de establecer de manera explícita.
- Hay atributos no heredables. Ejemplo:
  - *tiene-por-instancia*
- Dado un marco es posible que la representación permita heredar un valor a través de múltiples relaciones (**herencia múltiple**): hay que establecer **criterios** (ejemplo: camino más corto).

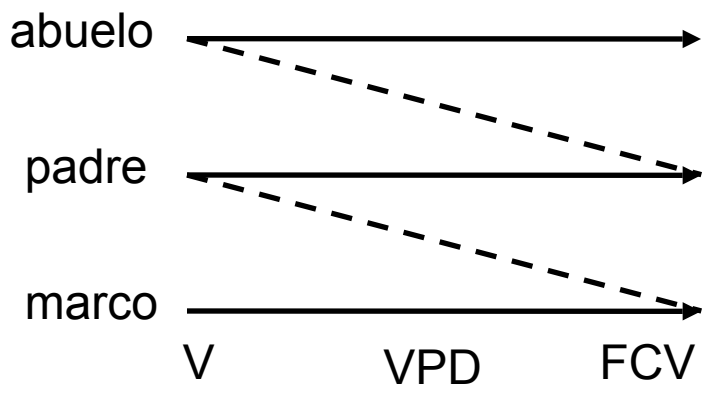
N-herencia

- Prioridad a los *facets* de los atributos
  1. Consultar el facet *valor* del atributo en el marco considerado
  2. Si no tiene valor, subir un nivel en la taxonomía i consultar *valor*
  3. Repetir la operación hasta encontrar valor en algún marco superior o bien llegar a la cabeza de la jerarquía
  4. En este último caso, volver al marco considerado y considerar *valor-por-defecto* del atributo y repetir la búsqueda hacia arriba
  5. Si tampoco hay éxito, consultar *función-para-calcular-valor* del atributo



Z-herencia

- Prioridad a los marcos más cercanos
  1. Consultar el facet *valor* del atributo en el marco considerado
  2. Si no tiene valor, consultar *valor-por-defecto*
  3. Si tampoco tiene, consultar *función-para-calcular-valor*
  4. Agotadas las posibilidades del marco considerado, se sube en la jerarquía y se hace lo mismo con el marco padre

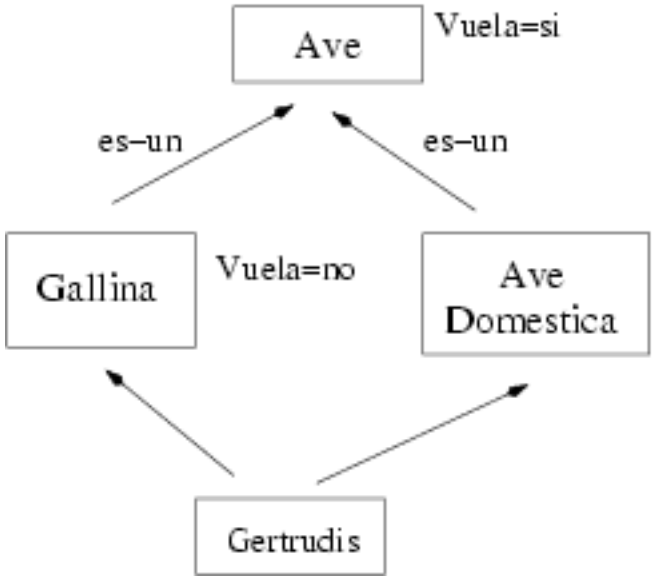


Herencia simple y múltiple

- Si la taxonomía es un **árbol**, la herencia es **simple**.
- La herencia es **múltiple** si:
  - la taxonomía es un **grafo** (dirigido acíclico)
  - **hay otras relaciones** (no taxonómicas) que permiten herencia
- **¡Puede haber conflicto de valores!**

  (obviamente, sólo si hay herencia de atributo **y valor**)

Marcos: herencia múltiple



- ¿Gertrudis vuela?
- Si considero el camino más corto: no.
- Pero, ¿si Gallina está sub-clasificado?
- El algoritmo de **distancia inferencial** permite establecer cual es el marco del que se ha de heredar.

Herencia: distancia inferencial

1. Buscar el conjunto de marcos que permiten heredar el valor del atributo → **Candidatos**
2. Eliminar de la lista **Candidatos** todo marco que sea padre de otro de la lista
3. Si el nuevo número de candidatos es:

   0 → No se puede heredar el atributo

   1 → Ese es el valor que se hereda

   N > 1 → Problema de herencia múltiple si la cardinalidad del atributo no es N

Ontologías

- Una ontología **en IA** se define como una:

formal explicit representation of a shared understanding of the important concepts in some domain of interest

- En filosofía el término ontología se refiere a un concepto más delicado, aunque relacionado:

the study of being as such

Formalidad

- Las ontologías son (deberían ser) formales: tienen que ser leíbles por los ordenadores.
- Nivel de formalidad:
  - de **altamente informal**: lenguaje natural
  - a **rigurosamente formal**: términos con semántica formal y axiomas
- En las ontologías “maduras”, las descripciones permitidas son sólo las consistentes con un conjunto de axiomas, que determinan su uso.

Especificación explícita

- Los **tipos** de los conceptos y las restricciones sobre su uso están (deberían estar) **definidos explícitamente**.
- Accesibilidad y transparencia:
  - documentación de los detalles técnicos

Comprensión compartida del conocimiento

- Las ontologías contienen (deberían contener) **conocimiento consensual**, aceptado por un grupo de personas lo más amplio posible.

Conceptualización

- Las ontologías son un modelo abstracto de algún dominio de algún mundo posible.
- Compromiso común: hacer el mínimo número de afirmaciones posible (sólo las necesarias) sobre el dominio que se está modelando, dejando a los reutilizadores de la ontología la libertad de especializarla e instanciarla tanto como haga falta.

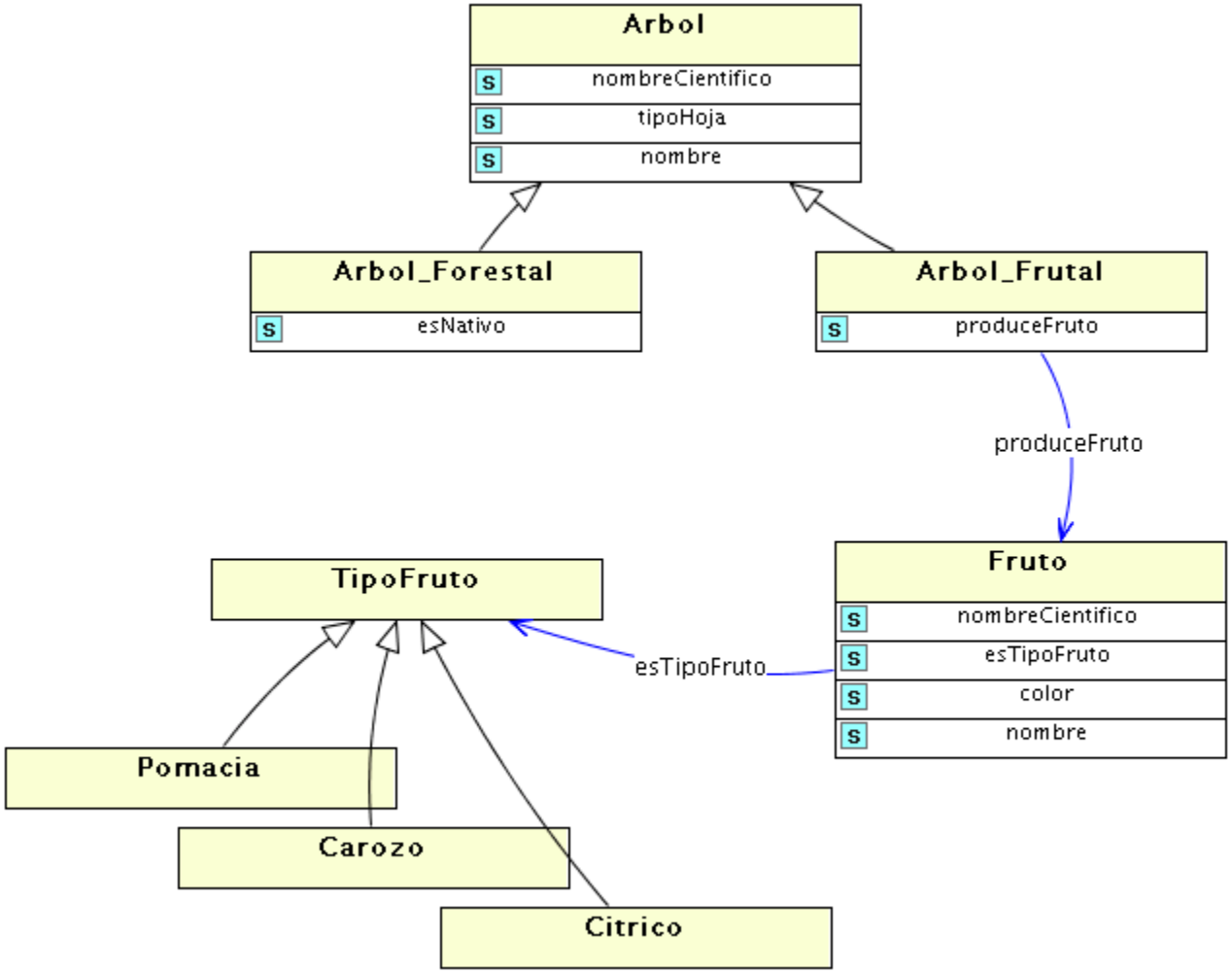
¿Por qué son útiles las ontologías?

- Modelado y uso compartido del conocimiento:
  - **comunicación** entre personas/agentes con diferentes necesidades y puntos de vistas debidos a contextos diferentes
    - marco unificado dentro de una **organización** para reducir la confusión conceptual y terminológica
  - **interoperabilidad** entre sistemas a través de traducciones entre diferentes paradigmas, lenguajes, herramientas informáticas y métodos de modelado
    - ontologías como ínter-lengua
    - integración de ontologías

¿Por qué son útiles las ontologías?

- Ingeniería de sistemas:
  - **re-usabilidad**
    - Ontologías altamente configurables y bibliotecas de ontologías facilitan la re-usabilidad entre diferentes sistemas de software.
  - **fiabilidad**
    - Las ontologías formales permiten la comprobación de consistencia, dando lugar a sistemas más fiables.

Ejemplo



Conceptos, organizados en taxonomías, enlazados por relaciones, en acuerdo con los axiomas.

Axiomas, posiblemente muy generales, p.e.: "*Árbol* no es *Árbol de navidad*".

Ingenieros de ontologías

- Los **ingenieros del conocimiento** construyen ontologías basadas en conocimiento **consensual** sobre un dominio
- Permiten a un grupo de agentes determinado (**usuarios** de la ontología) **compartir y reutilizar** ese conocimiento dentro del área de trabajo seleccionada (el dominio).

Metodologías

- Existen y están disponibles varias metodologías para la construcción de ontologías:
  - Mike Uschold
  - Michael Grüninger
  - Asunción Gómez-Pérez
  - John Sowa
- Enlace de referencia general:
  - http://www.lsi.upc.edu/~luigi/ontologies.htm

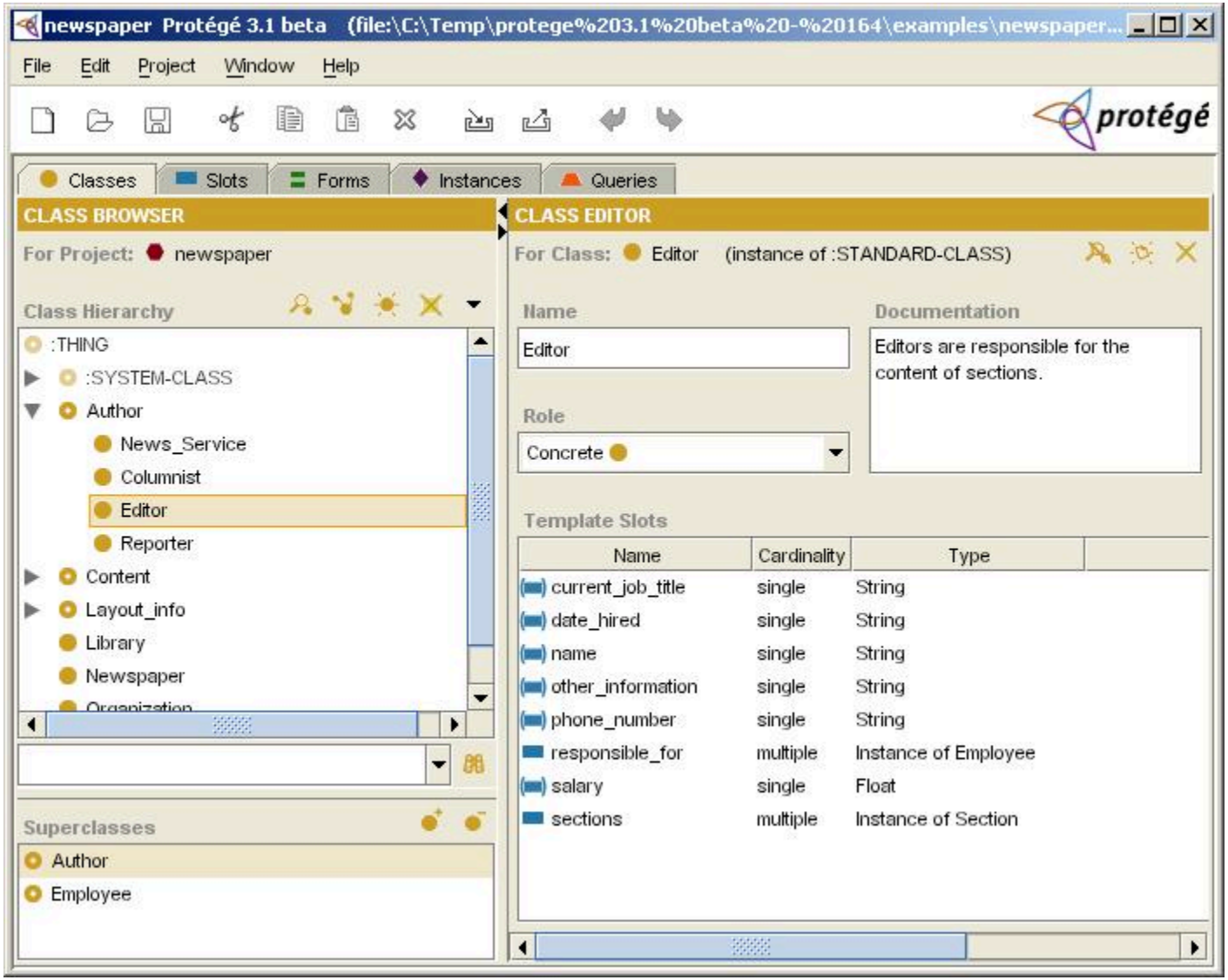
Construcción cooperativa

- Herramientas online:
  - **Ontolingua** Ontology Editor (*Stanford University*): http://www-ksl-svc.stanford.edu:5915/&service=frame-editor
  - **Ontosaurus** Web Browser (*University of Southern California*): http://www.isi.edu/isd/ontosaurus.html
  - **Tadzebao** y **WebOnto** (*The Open University*): http://riverside.open.ac.uk

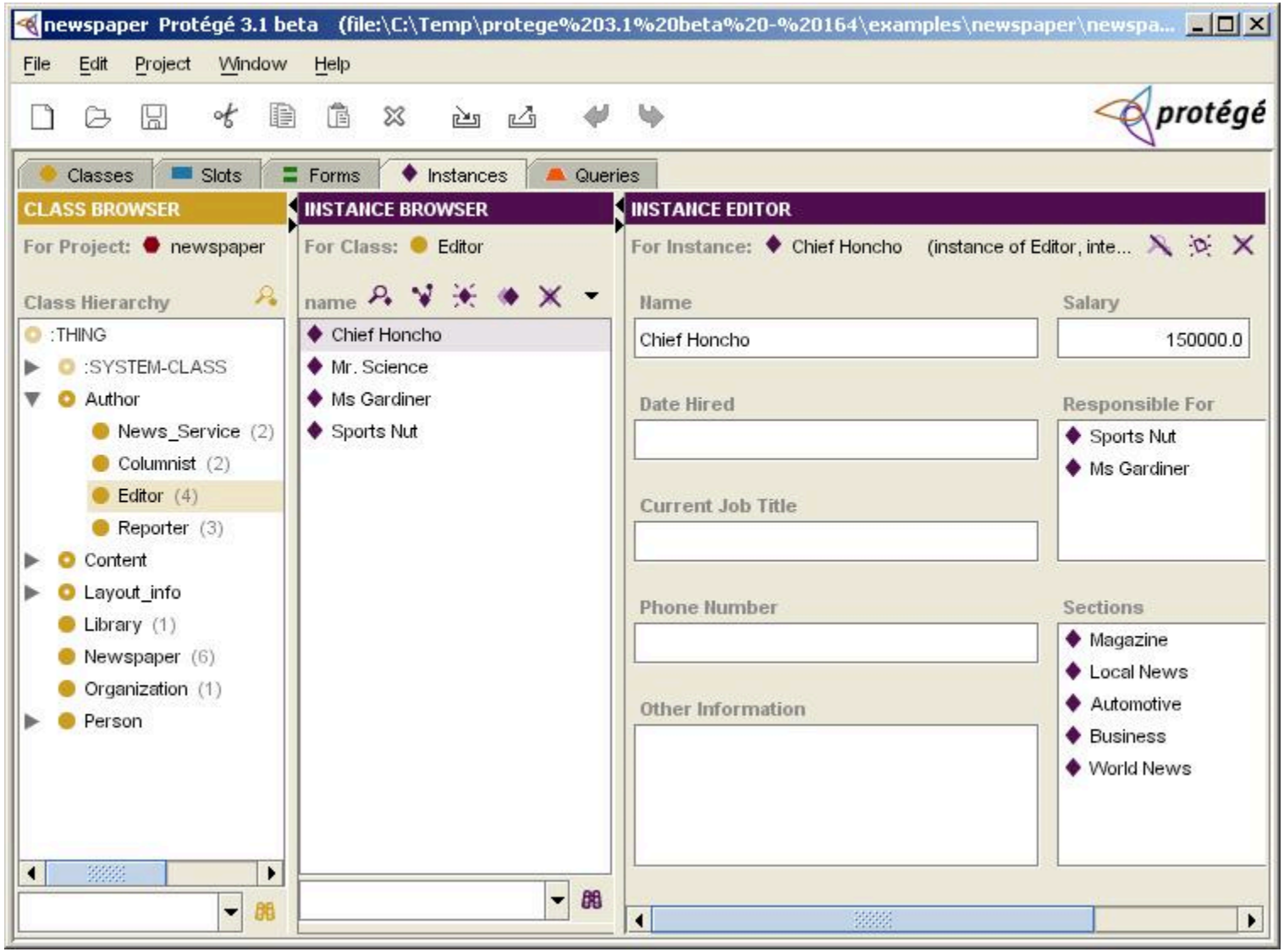
Construcción cooperativa

- Herramientas offline:
  - **OilEd** (*University of Manchester*): http://oiled.man.ac.uk/
  - **OntoStudio** (*Ontoprise*): http://www.ontoprise.de/content/e3/e43/index_eng.html
  - **Protégé** (*Stanford University*): http://protege.stanford.edu/
    - versión 3.2
  -

Protégé


newspaper Protégé 3.1 beta (file:\C:\Temp\protege%203.1%20beta%20-%20164\examples\newspaper...
File Edit Project Window Help
protégé
Classes Slots Forms Instances Queries
CLASS BROWSER
For Project: newspaper
Class Hierarchy
:THING
:SYSTEM-CLASS
Author
News_Service
Columnist
Editor
Reporter
Content
Layout_info
Library
Newspaper
Superclasses
Author
Employee
CLASS EDITOR
For Class: Editor (instance of :STANDARD-CLASS)
Name
Editor
Documentation
Editors are responsible for the content of sections.
Role
Concrete
Template Slots
Name Cardinality Type
current_job_title single String
date_hired single String
name single String
other_information single String
phone_number single String
responsible_for multiple Instance of Employee
salary single Float
sections multiple Instance of Section

Protégé


newspaper Protégé 3.1 beta (file:\C:\Temp\protege%203.1%20beta%20-%20164\examples\newspaper\newspa...
File Edit Project Window Help
protégé
Classes Slots Forms Instances Queries
CLASS BROWSER
For Project: newspaper
Class Hierarchy
:THING
:SYSTEM-CLASS
Author
News_Service (2)
Columnist (2)
Editor (4)
Reporter (3)
Content
Layout_info
Library (1)
Newspaper (6)
Organization (1)
Person
INSTANCE BROWSER
For Class: Editor
name
Chief Honcho
Mr. Science
Ms Gardiner
Sports Nut
INSTANCE EDITOR
For Instance: Chief Honcho (instance of Editor, inte...
Name
Chief Honcho
Salary
150000.0
Date Hired
Responsible For
Sports Nut
Ms Gardiner
Current Job Title
Phone Number
Sections
Magazine
Local News
Automotive
Business
World News
Other Information

Implementación

- **Uschold**: directamente en lenguajes específicos para ontologías (por ejemplo, Ontolingua, LOOM, KIF)
- **Gómez-Pérez**: a nivel de conocimiento del dominio, a través de un *shell* que traduce a Ontolingua
- **Protégé**: entorno visual con entrada y salida en OWL

Amplia gama de aplicaciones

- Gestión del conocimiento
- Generación de lenguaje natural
- Modelado de los procesos de empresa
- SBCs
- Navegadores de Internet
- Interoperabilidad entre sistemas
- Ingeniería de sistemas: especificación, fiabilidad, reutilización

Ejemplo: ontología para gestión de aguas residuales

- Methodology
  - Requirement determination:
    - Utility (necessity?)
    - Purposes
  - Knowledge acquisition
  - Conceptualization
  - Preliminary lists of terms
  - Development environment
  - Reuse
  - Formalization

Background

- Experts in various domains:
  - AI
  - chemical engineering
  - microbiology
  - computing
- Experts with different vocabularies
- Common terminology: not always
- No rules for term use, synonyms

Objetivos de la ontología

- To obtain a unified (but multilingual), complete and coherent terminology for the domain of wastewater
- To help in the diagnosis of problematic situations in wastewater
- To help in the management of wastewater treatment plants

Adquisición del conocimiento

- Reuse of existing, specific encoded knowledge
- Interviews with experts
- Texts analysis

Conceptualización

- Criteria followed:
  - easily understandable model
  - reflection of expert knowledge
  - easily extendable knowledge
  - easily integration with other existing ontologies
  - possibility to choose a subset of the knowledge and to use it in other ontologies or applications

Términos y jerarquía

- Actuator
- Body-Of-Water
- Descriptor
  - Descriptor-Off-Line
    - Descriptor-Qualitative
      - Appearance-Floc
      - Appearance-Surface-Clarifier
    - Descriptor-Quantitative
      - BOD
      - Chlorine Cod
  - Descriptor-On-Line
    - Water-Flow

Entorno de desarrollo

- Term hierarchy: **Ontolingua** because it was a quasi-standard
- Axioms: first in **Lisp**, then in **KIF** (associated to Ontolingua)
- Alternative: **Protégé** + **OWL**

Representaciones

- Classification tree of concepts: a taxonomy of concepts
- Glossary of terms: including concepts, individuals, attributes and axioms
- Various representations are predefined and available in the Ontolingua Ontology Server

Reutilización

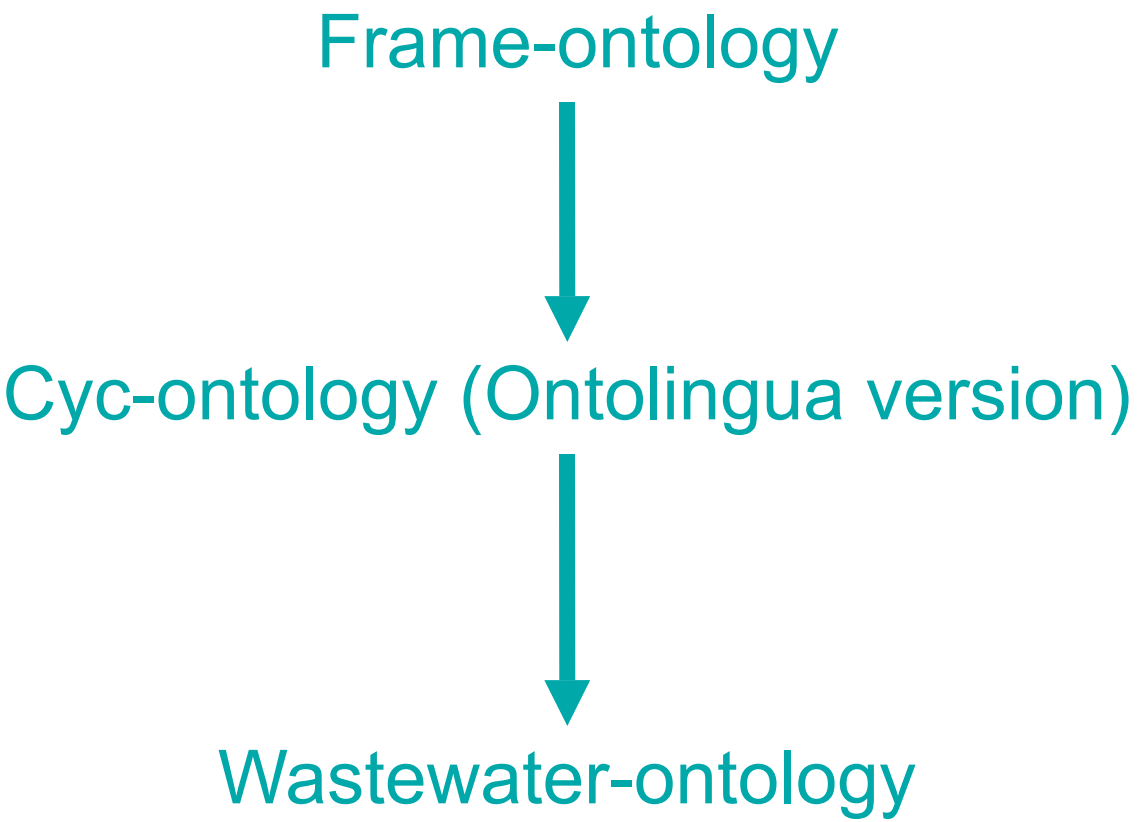
- The ontology is mainly built from scratch.

- Two other ontology are partially reused:
  - Frame ontology: it's Ontolingua meta-ontology
  - Cyc ontology (Lenat and Guha, 1990): manual encoding of millions of encyclopedic facts (Ontolingua version)

Re-engineering

- **Frame-ontology**: as is
- **Cyc**: little re-engineering to adapt the concepts to the specific domain
- No ontology **integration** (and consistency checking) needed.

Arquitectura de la información



Términos: detalles

- The ontology use the English language to describe the world at the highest level.
- Nevertheless the documentation of each term contains the translation of the term name to Spanish, Catalan and Italian.

- **Documentation**: *Class* ***Descriptor-Qualitative***
  - Synonyms:
    - Dato cualitativo (esp)
    - Dada qualitativa (cat)
    - Dato qualitativo (ita)

  Qualitative descriptors refer to microscopic observations (e.g.: activated-sludge microbes presence and diversity) and visual information (e.g.: presence of foam in the aeration tank or in the secondary clarifier, water odor, biomass color).

Formalización

- The Ontolingua Ontology Server automatically translates to Lisp.

- The translation has been checked and evaluated.

Conclusiones

- Success in unifying terminology and as a way of communication between AI researchers and chemical engineers.

- Evaluation as a tool of support to diagnosis.