Ontology Development

A guide to creating the first ontology

Michael Tschuggnall

Content

1. Ontology definition and motivation
   - What is an ontology?
   - Why should an ontology be devoloped?
2. Ontology engineering methodologies
3. Ontology development
   - Basic concepts:
     classes, subclasses, slots, facets
   - 7 steps to create an ontology
4. Example
   - The Simpsons Ontology (by STI)

Ontology Definition

Ontology

- An ontology defines a common vocabulary for researchers who need to share information in a domain
- includes
  - machine-interpretable definitions of basic concepts
  - relations between them
- many definitions in the Artificial Intelligence literature, many of them contradict with each other

Why developing an ontology?

- to share common understanding of the structure of information among people or software agents
- to enable reuse of domain knowledge
- to make domain assumptions explicit
- to separate domain knowledge from the operational knowledge
- to analyze domain knowledge

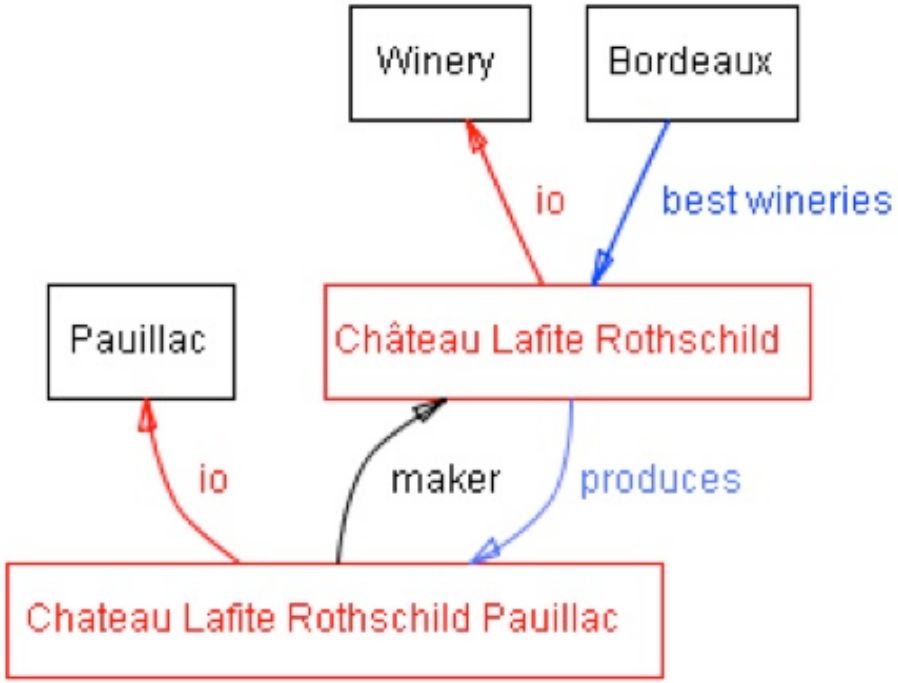
Ontology definition

- definition in our case:
- an **ontology** is
  - a formal explicit description of concepts in a domain (***classes***, sometimes called ***concepts***),
  - properties of each concept describing various features and attributes of the concept (***slots***, sometimes called ***roles*** or ***properties***), and
  - restrictions on slots (**facets,** sometimes called ***role restrictions***)
- ontology + set of individual instances = **knowledge base**

Classes and Slots

- **Classes** describe concepts in the domain
  - e.g.: *Wine*
- can have **subclasses** that represent concepts that are more specific than the superclass
  - e.g.: *Red Wine, White Wine, Rose Wine*
- **Slots** describe properties of classes and instances
  - e.g.: „*Château Lafite Rothschild Pauillac* has a full body and is produced by the *Château Lafite Rothschild* winery“
  - two slots:
    - *body* with value *full*
    - *maker* with value *Château Lafite Rothschild* winery

Example



- black: Classes
- red: instances
- io: instance-of
- direct links represent slots

Ontology Development

- no one single correct way to model a domain, but many alternatives
- depends on the application
- Ontology development is necessarily an iterative process
- Concepts in the ontology should be close to objects (physical or logical) and relationships in the domain of interest
  - nouns (objects)
  - verbs (relationships)

Ontology Development

- ontology development includes
  - defining classes in the ontology
  - arranging the classes in a taxonomic (subclass–superclass) hierarchy
  - defining slots and describing allowed values for these slots
  - filling in the values for slots for instances
- after that a **knowledge base** can be created by
  - defining individual instances
  - filling the slots with specific values
  - adding restrictions to slots

Ontology Engineering Methodologies (1)

- describe methods/activities to construct an ontology
- Why formal methodologies?
  - development of consistent ontologies
  - efficient development of complex ontologies
  - distributed development of ontologies
- many different proposals, but no tools that support the methodologies directly

Ontology Engineering Methodologies (2)

- Many different methodologies distinguishing
  - ontology management activities
  - ontology development oriented activities
  - ontology support activities
- Some examples:
  - **Ontology Development 101**
  - METHONTOLOGY
  - OTK (Ontology engineering for knowledge management systems)
  - DILIGENT (Distributed, loosely controlled and evolving engineering of ontologies)
  - methodology by Ushold and King
  - methodology by Grüninger and Fox

Ontology Development 101

Ontology Development 101: 7 steps

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

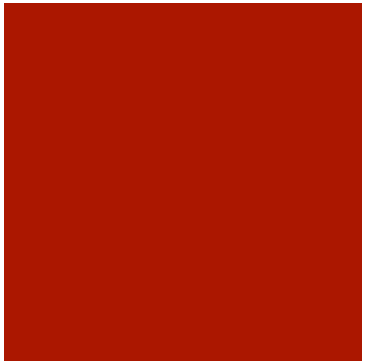
Step 1: Domain and Scope (1)

- determine the **domain** by answering the following questions:
  - What is the domain that the ontology will cover?
  - For what is the ontology going to be used?
  - For what types of questions the information in the ontology should provide answers?
  - Who will use and maintain the ontology?
- one way to determine the **scope** of the ontology is to use **competency questions**:
  - a list of sketched questions that the knowledge base based on the ontology should be able to answer

Step 1: Domain and Scope (2)

- **Competency questions** in the *Wine* domain could be:
  - 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?
  - ...

Step 2: Reusing existing ontologies



- Check if there are existing ontologies which can be extended for the particular domain
- Many ontologies already defined on the web:
  - Ontolingua ontology library (http://www.ksl.stanford.edu/software/ontolingua/)
  - the DAML ontology library (http://www.daml.org/ontologies/)
  - UNSPSC (www.unspsc.org)
  - RosettaNet (www.rosettanet.org),
  - DMOZ (www.dmoz.org)

Step 3: Enumerate important terms

- List important terms like:
  - What are the terms the ontology should talk about?
  - What properties do those terms have?
  - What can be said about the terms?
- List the terms without considering overlaps between concepts
- Example in the *Wine* domain:
  - wine, grape, winery, location of the winery, color, body, flavour, sugar content, ...

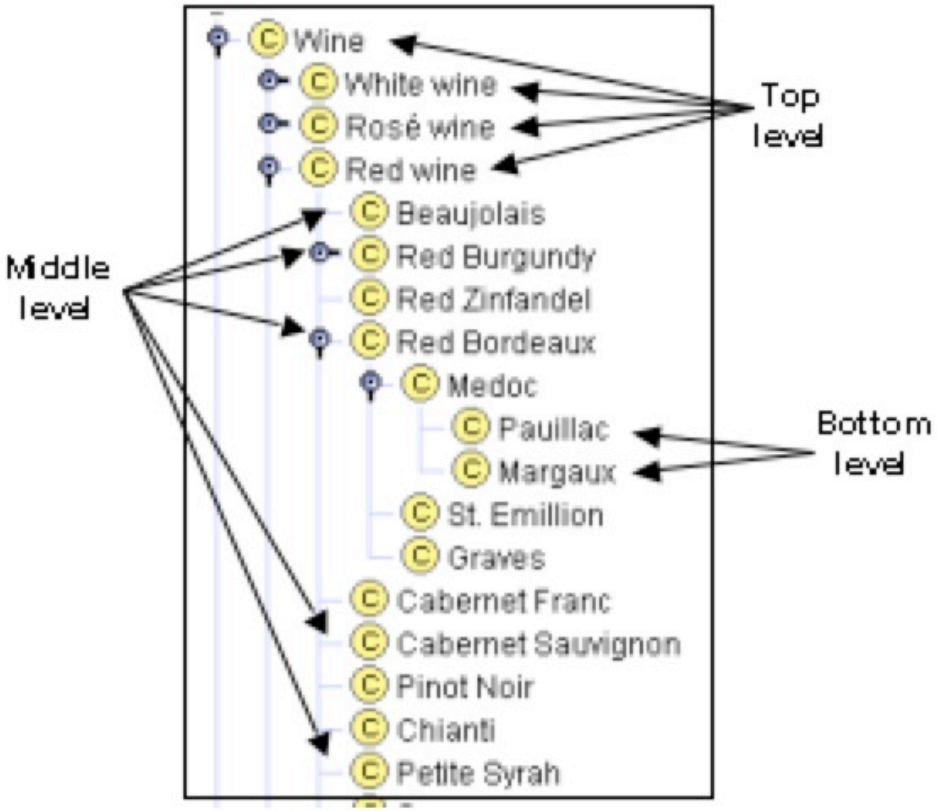
Step 4:
Define classes and hierarchy (1)

- 2 approaches:
  - top down
  - bottom up
- **Top-down** approach:
  - starts with the definition of the most general concepts
  - adds specialications („kind-of“) of those concepts subsequently
  - e.g:
    1. *Wine*, Food
    2. White Wine, *Red Wine*, Rosé Wine
    3. Red Burgundy, Cabernet Sauvignon, Syrah
    4. ...

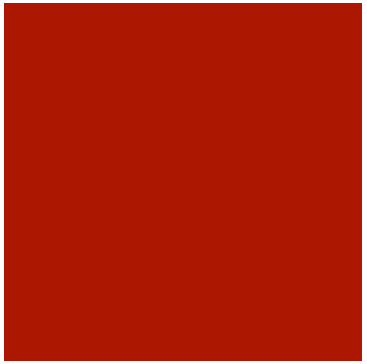
Step 4:
Define classes and hierarchy (2)

- **Bottom-up** approach:
  - starts with the definition of the most specific classes (leaves)
  - groups these classes subsequently into more general concepts
  - e.g.:
    1. *Pauillac*, *Margaux*, Chianti Classico
    2. *Medoc*
    3. *Bordeaux*
    4. Red Wine
- in practice: **combination** of top-down and bottom-up

Step 4:
Define classes and hierarchy (3)



Step 5: Define properties / slots



- To answer the questions defined in step 1, more information is needed
- Look at the terms from step 3: the ones which are left are likely to be properties of classes
- decide for each term to which class it belongs, then add it as a slot
- different kinds of properties:
  - intrinsic (e.g. *flavour* of a wine)
  - extrinsic (e.g. *name*, *area* of a wine)
  - relationships between different members of the class or other items
- subclasses **inherit** all slots from superclasses

Step 6:
Define facets of the slots (1)

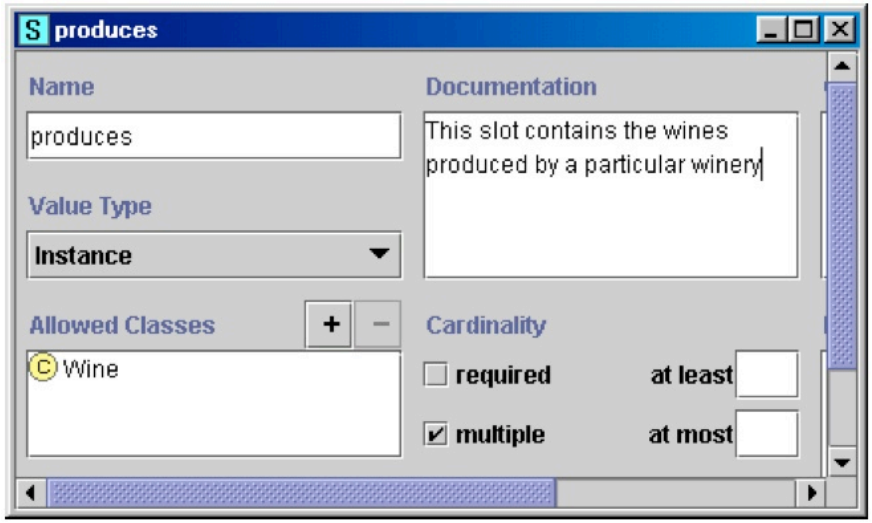
- a slot can have different facets describing:
  - value type
  - cardinality
  - allowed values (domain and range)
- common **value types** are:
  - String
  - Number
  - Boolean
  - Enumerated
    - specify a list of specific allowed values
  - Instance
    - relationship to another instance
    - allowed classes must be defined!

Step 6: Define facets of the slots (2)

- the **slot cardinality** defines, how many instances a slot can have
- depending on the system the following cardinalities can be distinguished:
  - **single / multiple** cardinality (at most one / any number)
  - **minimum / maximum** (at least / at most)
- sometimes it may be useful to set the (maximum) cardinality to zero, indicating that this slot cannot have any values for a particular subclass

Step 6: Define facets of the slots (3)

- Allowed classes for slots with type *Instance* are called the **range** of a slot
- Example of the class *Winery* that has a slot *produces* of type *Instance,* where only instances of the class *Wine* are allowed:



Step 6: Define facets of the slots (4)

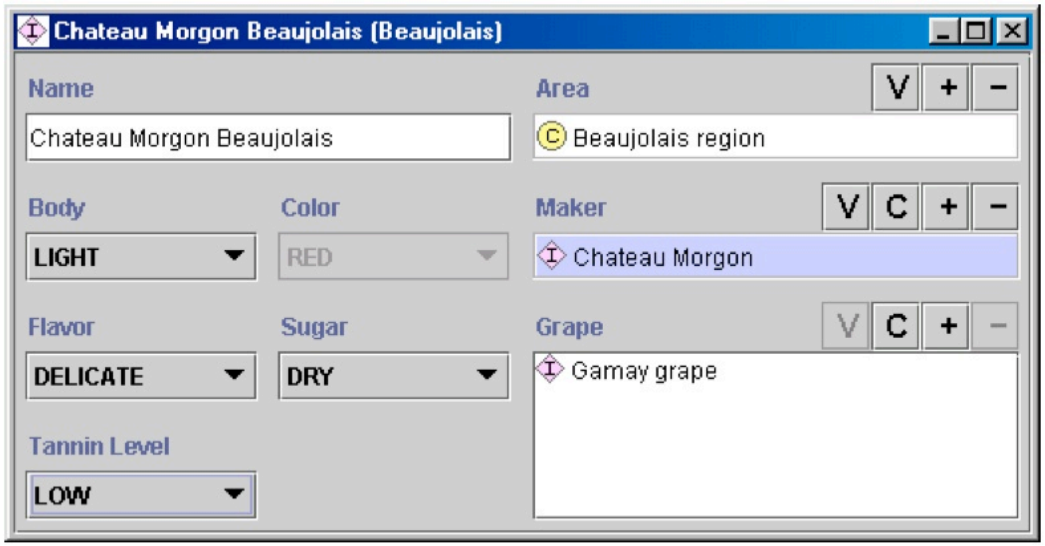
- The classes to which a slot is attached are called the **domain** of the slot
  - in the previous example: the class *Winery* is the domain of the slot *produces*

- some rules when defining the range and domain:
  1. Find the **most general classes** that can be the domain or range of a slot
  2. If a list of classes defining a range or a domain of a slot includes a class and its subclass, **remove the subclass**.
  3. If a list of classes defining a range or a domain of a slot contains all subclasses of a class A, but not the class A itself, the range should **contain only the class A** and **not the subclasses**.
  4. If a list of classes defining a range or a domain of a slot contains all but a few subclasses of a class A, consider if the class A would make **a more appropriate range definition**.

Step 7: Create instances

- As the last step, instances can be created by
  1. choosing a class
  2. creating an individual instance of that class
  3. filling in the slot values

- Example in the *Wine* domain:



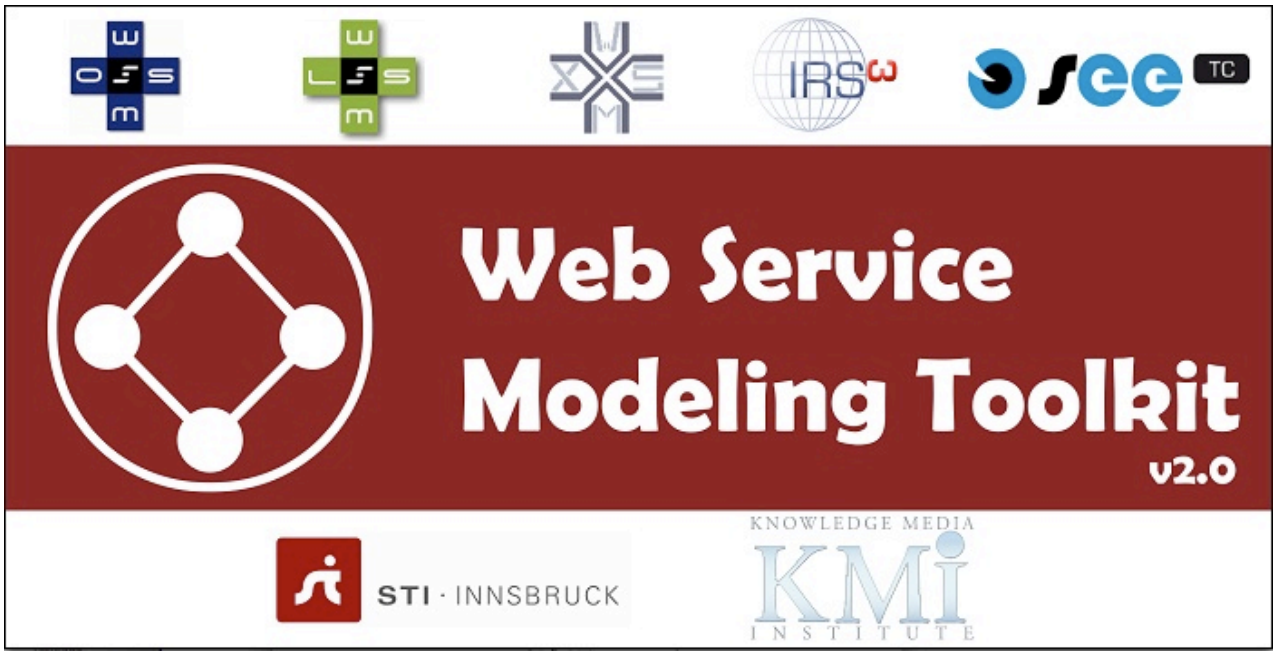
Ontology Tools

WSMO / WSML

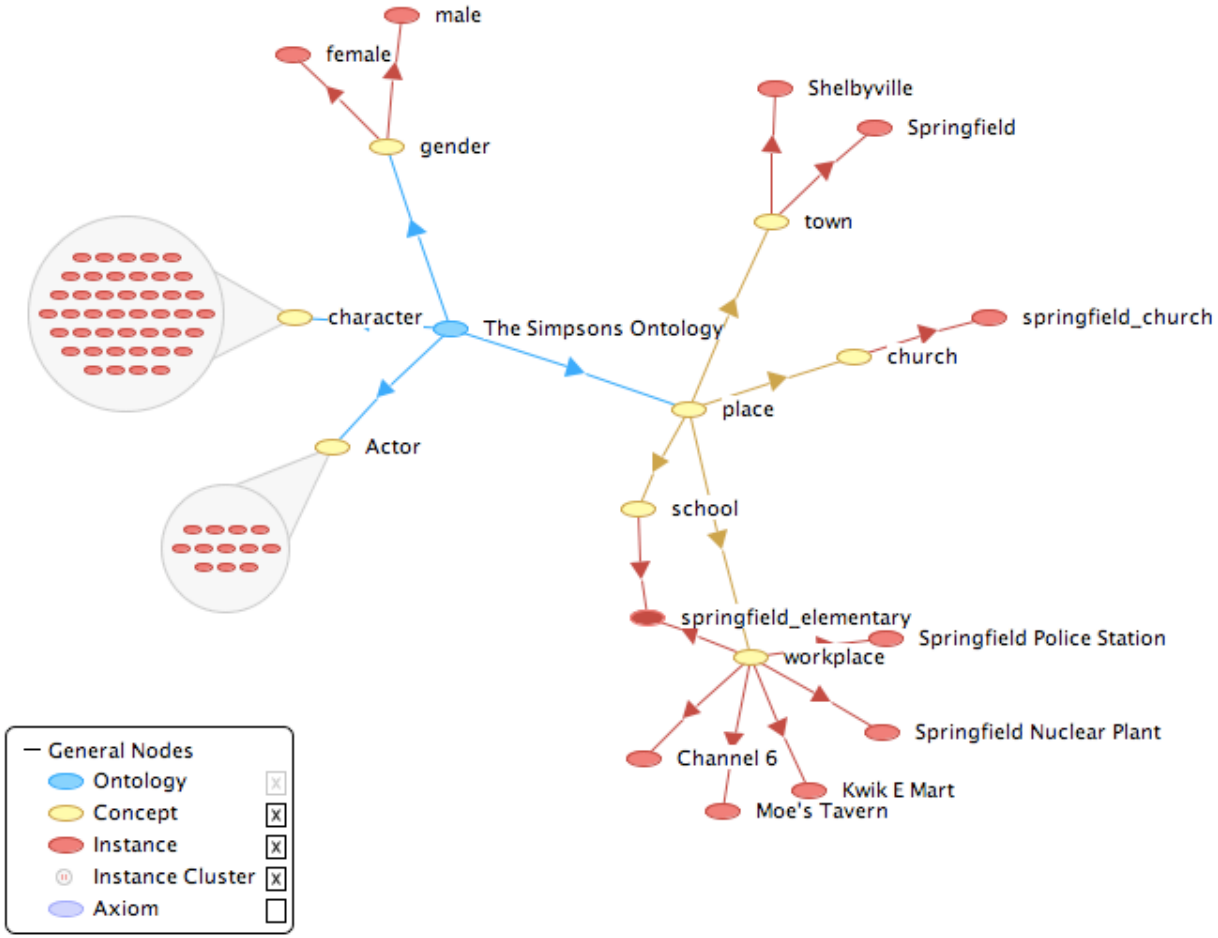
- **WSMO** = Web Services Modeling Ontology
- **WSML** = Web Services Modeling Language
  - formalizes WSMO
  - can be used to describe ontologies
- developed by STI Innsbruck
- tools, documentations and presentations downloadable from:
  - http://www.wsmo.org/
  - http://www.wsmo.org/wsml/

WSMT:
Web Service Modeling Toolkit

- developed by STI
- more details in the demo of Michael Rogger

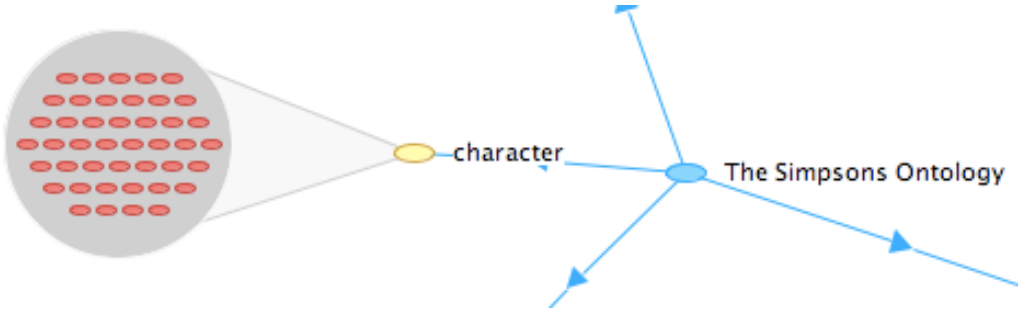


Example:
The Simpsons Ontology (1)



Example: The Simpsons Ontology (2)

- Character definition:

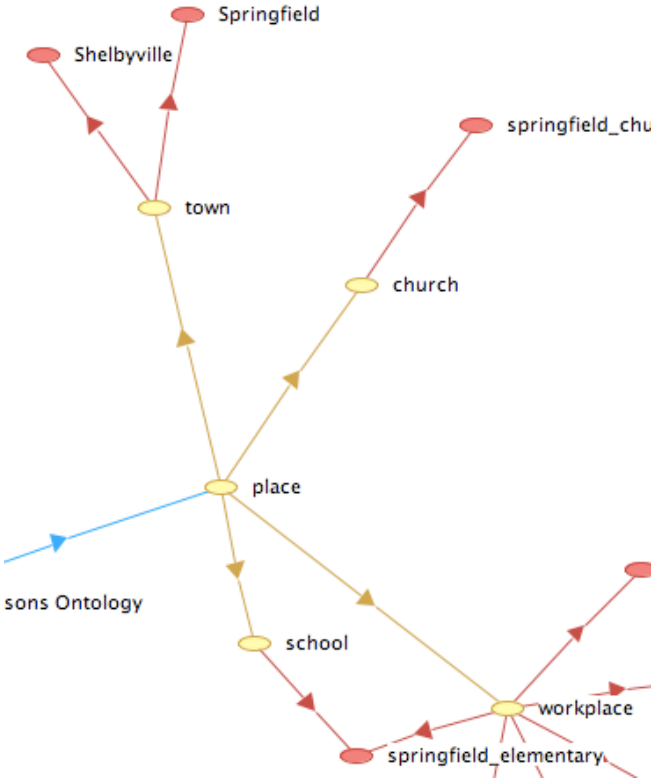



```
concept character
     hasName ofType _string
     hasGender ofType gender
     hasActor ofType actor
     hasSpouse ofType character
     hasChild ofType character
     hasParent ofType character
     hasSibling ofType character
     hasFriend ofType character
     hasNeighbour ofType character
     hasCatchPhrase ofType _string
     inLoveWith ofType character
     isCustomerOf ofType workplace
     hasWorkingPlace ofType place
     worshipsAt ofType church
     principleOf ofType school
     attends ofType school
     owns ofType workplace
     mayorOf ofType town
     policeChiefOf ofType town
     reverantOf ofType church
```

```
instance bart_simpson memberOf character
     nonFunctionalProperties
          dc#title hasValue "Bart Simpson"
     endNonFunctionalProperties
     hasName hasValue "Bart Simpson"
     hasGender hasValue male
     hasActor hasValue nancy_cartwright
     hasParent hasValue {homer_simpson, marge_simpson }
     hasSibling hasValue {lisa_simpson, maggie_simpson }
     hasFriend hasValue milhouse_van_houten
     hasNeighbour hasValue {ned_flanders, maude_flanders, todd_flanders }
     hasCatchPhrase hasValue {"I'm Bart Simpson, who the hell are you"}
     attends hasValue springfield_elementary
     worshipsAt hasValue springfield_church
```

Example: The Simpsons Ontology (3)

- Classes and subclasses:




```
concept place
     hasName ofType _string

concept town subConceptOf place
     hasMayor ofType character
     hasPoliceChief ofType character

concept workplace subConceptOf place
     hasOwner ofType character
     hasLocation ofType town

concept school subConceptOf place
     hasPrinciple ofType character
     hasLocation ofType town

concept church subConceptOf place
     hasReverant ofType character
     hasLocation ofType town
```

References

1. N. Noy, D. McGuinness: „*Ontology Development 101: A guide to creating your first ontology*"
2. M. Fernandez-Lopez; „*Overview of Methodologies for building Ontologies*“
3. WSMO: http://www.wsmo.org/
4. Protegé: http://protege.stanford.edu/
5. Further interesting links can be found at: http://www.aifb.uni-karlsruhe.de/WBS/cte/ontologyengineering/

End

Thank you for your attention!