

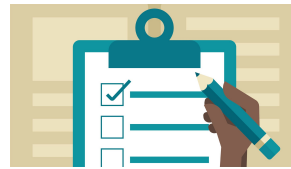
Building Ontologies

Semantic Relationships and
Competency Questions

IMT 530
Spring 2019

Learning Objectives

- Review types of relationships.
- Understand what makes ontologies unique and powerful.
- Learn how to build an ontology.
- Look at a bunch of different ontologies.
 - Consider design trade-offs
 - Learn tools



Example: Canopy Tree Library



<https://canopy.org/tree-info/canopy-tree-library/>

All Trees Found on Tree Walks Low Watering Needs Locally Native Trees Shade Providing OK Under Power Lines Thirsty Trees Not Recommended

The Mid-Peninsula low-lands were originally an oak-studded savanna punctuated by wooded riparian corridors. Over the years many species non-native to this area have been planted as cities and neighborhoods were developed. This list includes the majority of tree species found along the streets, in parks and gardens of Palo Alto. It is pretty typical of trees found in Northern California communities.

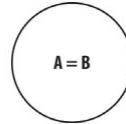
All Trees (384) Filter list

Common Name	Size
African Sumac	Height: <input type="text" value="any of: 8"/> <input type="checkbox"/> Small <input type="checkbox"/> Medium <input type="checkbox"/> Large <input type="checkbox"/> Extra large
Akebono Cherry	Spread: <input type="text" value="any of: 4"/> <input type="checkbox"/> Small <input type="checkbox"/> Medium <input type="checkbox"/> Large
Alba (White Redbud)	Trunk Clearance: <input type="text" value="any of: 8"/> <input type="checkbox"/> 2.5 ft radius <input type="checkbox"/> 5 ft radius <input type="checkbox"/> 7.5 ft radius
Aleppo Pine	Growth Rate: <input type="text" value="any of: 2"/> <input type="checkbox"/> Slow <input type="checkbox"/> Moderate <input type="checkbox"/> Fast
Allee Elm	Appearance
Almond	Ornamental Attributes: <input type="text" value="any of: 4"/> <input type="checkbox"/> Flowers <input type="checkbox"/> Fruits <input type="checkbox"/> Seeds <input type="checkbox"/> Fall color <input type="checkbox"/> Bark <input type="checkbox"/> Form
American Arborvitae	Litter Production: <input type="checkbox"/> Light <input type="checkbox"/> Medium <input type="checkbox"/> Heavy
American Chestnut	Type: <input type="text" value="any of: 2"/> <input type="checkbox"/> Evergreen <input type="checkbox"/> Deciduous
American Elm	Form: <input type="text" value="any of: 2"/> <input type="checkbox"/> Columnar <input type="checkbox"/> Conical <input type="checkbox"/> Palm <input type="checkbox"/> Pendulous <input type="checkbox"/> Rounded <input type="checkbox"/> Spreading <input type="checkbox"/> Vase <input type="checkbox"/> Upright
American Hornbeam	Shade Production: <input type="checkbox"/> Dense <input type="checkbox"/> Filtered <input type="checkbox"/> Little
American Linden	
Apricot	
Araucaria	
Arbutus Marina	

Let's Review: Types of Relationships

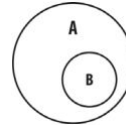
Equivalence (PT/ET)

- Synonyms, variants, etc.



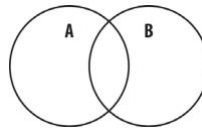
Hierarchical (BT/NT)

- Generic-specific
- Whole-part
- Instance



Associative (RT)

- See also, etc.
- **A bunch of more interesting semantic relationships**



We're gonna talk about the interesting semantic relationships today.

Note: We saw these pictures before. The pictures themselves are not in the ANSI/NISO standard, but the relationships are. The pictures are from the Information Architecture chapter that you read for this week.

Let's Review: Types of Relationships

Equivalence (PT/ET)

- Synonyms, variants, etc.


Hierarchical (BT/NT)

- Generic-specific
- Whole-part
- Instance

Associative (RT)

- See also, etc.
- **A bunch of more interesting semantic relationships**

**This is what
we're focusing
on today.
Ontologies!**



Ontologies have all of these relationships, but what makes them different from synonym rings (equivalence), taxonomies (hierarchies), and thesauri (simple associations) are the semantic relationships. We're gonna focus on those interesting semantic relationships today.

Quick Side Trip: Polyhierarchy

Narrower term with multiple broader terms

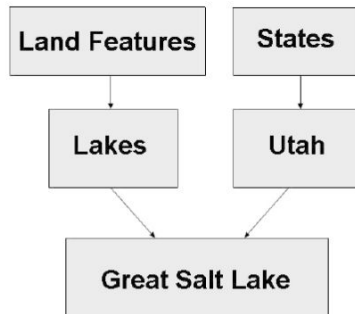


Figure 4.2 Polyhierarchy for the term *Great Salt Lake*

Quick sidetrack to discuss a type of hierarchical relationship. Note that semantic relationships are good at handling multiple relationships. Hierarchies struggle more because they have some assumptions that information objects or concepts “live” in one place. Think of the tradition of physical books being located in one place on the shelves.

Question:

- Let's say that you have a webpage with navigation on the left. Where does Great Salt Lake “live”? Under Lakes or under Utah? Both? What if you get there via search? What if you as the navigation designer choose Lakes as the main location, but also support browsing to GSL via Utah -- and then a user browses via Utah. Should the navigation show Utah as selected or Lakes?
- Similar experience for Dremel: Storage and Data Analysis.
- This is less of an issue when you allow users to select multiple facets.

What is an Ontology?

An ontology defines a **common vocabulary** for researchers who need to share information in a domain. Purposes:

- Share common understanding of the structure of information among people or software agents
- Enable reuse of domain knowledge
- Make domain assumptions explicit
- Separate domain knowledge from the operational knowledge
- Analyze domain knowledge

What is an Ontology?

“An ontology is a specification of a conceptualization.”

- Tom Gruber

What is an Ontology?

A formal explicit description of concepts in a domain of discourse.

- classes (sometimes called concepts)
- properties of each concept describing various features and attributes of the concept
- restrictions on properties (facets, sometimes called role restrictions)

So, How Is An Ontology Different From a Taxonomy?

UNIVERSITY of WASHINGTON



- Rich semantic relationships.
- Less emphasis on hierarchical relationships.
- Separation of domain knowledge (think hierarchies) and operational knowledge (think semantic relationships)
- Better support for inference (we'll see examples of this).
- Navigational: Think of the difference between a standard left nav and a faceted shoe-buying navigation.
- Search: Think of a few simple pieces of metadata to support full-text search vs. a rich set of connections and metadata.
- Ontologies require more work and are more powerful.

How to Build an Ontology

- There is not one true and correct way to build an ontology.
- Your approach depends on what you want to accomplish.
- The process is iterative.

That said, here are some steps from Noy and McGuinness.

1. Determine Domain and Scope

- 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?

This is straight from the reading.

Competency Questions

- Your design should be based on your users' goals. What information do they want?
- Does the ontology contain enough information to answer these types of questions?
- Example: What type of tree should I plant in my yard?

Unpack the tree question...what are some sub-questions there?

2. Consider Reusing Ontologies

Dublin Core

DBpedia

Wikidata

[Protege Ontology Library](#)

Lots more!



Note: Choose well-maintained ontologies.

Copying someone else's work is good here!

- Saves you time
- Consistency and interoperability
- Let domain experts be experts
- Be careful about using something that might change and fall out of date.

3. Enumerate Important Terms

- 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?

Corpus analysis is very important here.

Example from reading: important wine-related terms will include wine, grape, winery, location, a wine's color, body, flavor and sugar content; different types of food, such as fish and red meat; subtypes of wine such as white wine, and so on.

Be comprehensive now and weed out later.

Not worried about relationships at this stage, just the terms or concepts that you care about.

4. Define Classes and Hierarchy

Sounds a lot like building a taxonomy, eh?

It is.

- Top-down: Identify big categories and the list out the narrower terms.
- Bottom-up: Identify terms and group them into categories.



Pluses and minuses of flat structure when defining classes and hierarchies.

5. Define Properties of Classes

- In PoolParty, we've defined these as attributed in custom schemes.
- Noy and McGuinness: Intrinsic, extrinsic, relationships
- Doane: Administrative, descriptive
- Examples: Growth Rate, Shade Tolerance

Note: I'm going to try to avoid using 'slots' terminology because I think it is confusing.

As we've seen, the difference between a class and a property can be unclear. Make a design choice based on your domain and your competency questions. If I want to choose a tree species based on things like growth rate and shade tolerance, tree or tree species is a good class and growth rate and shade tolerance are good properties. See section 4.5 of Noy and McGuinness for more discussion.

6. Define Facets

- Single or multiple values?
- Data type: string, num, bool, enum, instance
- Examples:
 - Tree height: small, medium, large, extra large
 - Shade tolerant: yes/no
 - Trunk clearance: 5 feet

Your choices inform what users can do with your ontology.

I don't particularly like the use of "facets" here. It is mostly about data type.

7. Create Instances

- Also known as individuals, these are the leaf nodes--you can't go any further.
- You'll define the properties for each individual.
- Example: [Bigleaf maple](#)

We'll look at individuals in Protege.

Example: Google Internal Docs

Google's internal software documentation
2005-2015:

- Published in HTML, Wiki, Sites, Docs, etc.
- About 15 writers for 15,000 engineers
- Very few standards or rules
- A few key repositories, not well connected
- Internal Bookmarks server (tagging)
- Poor internal search

UNIVERSITY of WASHINGTON

W

Also mention that I maintained the Google Developer Handbook, which had a hierarchical left nav of all of Google's software infrastructure. It was very hard to get it "right" and to represent the complex relationships between pieces of code with just broader term/narrower term relationships.

About search: Google search was better than internal search for a few reasons:

- Much, much bigger corpus.
- Also, more complete corpus.
- Website owners highly motivated to improve search ranking.
- Much more linking between docs than on intranet.
- Intranet lacked metadata and aggregation pages.

Goal: Help engineers find useful documentation.

Glossary

Community authored, globally visible definitions of about 20,000 terms.

- Support for equivalence, broader/narrower, and related terms
- Links to key documents and the code base

Establishes some authority control, with preferred terms and entry terms.

- Examples include old code names and official project names.

Supports some amount of hierarchy:

- Example: a team can give all of their sub-projects the same tag to collect all of them in one place.

Related terms to help engineers navigate to similar projects, integration points, etc.

Links become a hugely important part of navigation because search is bad and links are curated.

Bookmarks Server

- Help engineers remember important pages.
- Folksonomy: Supports tagging like de.licio.us.
- Data visible to all Googlers.
- Not *that* popular, but a key data set.
- Relationships not well defined. Isn't exactly a synonym ring, taxonomy, thesaurus, or ontology.

Siftosaurus: Browsing/Searching

Used tag data as a starting point.

Ontologists map formal definitions:

- cpp == c++ == c == c-lang
- cpp is preferred term
- cpp is a programming language
- bigtable is a storage system

Identified key facets for audience: programming language, expertise, tech type, doc type, etc.

Note that formal definitions had to be continuously updated to pull in all of the variant terms and typos -- and to add new technologies.

Siftosaurus: Browsing/Searching

Built search and browse UI:

- Click on Python + Noogler + Logs + Codelab
- Or search, with autosuggest via controlled vocabulary
- Displays exact matches and partial matches in ranked order. Click on links for docs.

Also partnered with teams to build scoped navigation.

Working with teams was interesting:

- We got increased adoption and teams were eager to tag/bookmark their content to get it to show up in their navigation pages, which also made it show up globally.
- But we had issues with consistent terminology. They might use a term that means something specific locally, but means something else globally.

Dexy: Navigation

- Remember how docs were published all over the place? Wiki, HTML, etc.
- We wanted standard navigation on all of the Spanner pages so you could move easily between them.
- Is it tagged “Spanner” and “Developer Guide”? Embed script on pages and the pull in links for left nav.

This wasn't quite as fancy as it sounds. We had to do some ugly stuff in a spreadsheet - really! -- to get the navigation in the right order.

But what's important here is that once we established relationships between information resources and concepts, we could do multiple things -- build navigation, search, browsing, etc.

Note: It was named Dexy because it automatically updated “ran” at midnight every night.

Search Integration

Knowledge graph and knowledge panel

- Definition
- Key links
- Code location
- Project team
- Related concepts
- Enhanced search snippets

UNIVERSITY of WASHINGTON



This is where it gets awesome.

Dexy was implemented centrally, so adoption went relatively smoothly. I said we were doing it and my team did it.

But Sifto didn't get that much adoption. Our UI was clunky and engineers didn't want to go to a new UI to search for information even though internal search was poor.

So we went to where they were already: search.

Do you know the knowledge panel in Google search? (Do example search for UW iSchool.) That comes from Google's Knowledge Graph, which is very similar to DBpedia.

With our Glossary, Bookmarks, Sifto, and Dexy data, we knew about the most important concepts at Google and the most important documents about those concepts, and how they were related. (We also had another project to collect and report data on

how up-to-date and “good” those documents were.) We could feed that directly into search results.

- Definition
- Links to key docs
- Link to location in code base
- List of related concepts
- Link to project/team information

We also annotated results snippets (the little bit of text under the title in the list of organic results) to display things like Official Developer Guide or Rated 4.5/5 or last updated 2 weeks ago, etc. This information helped people make judgments about what links to follow -- and eventually was used to inform ranking.

These improvements were really powerful -- and they were based on using folksonomy data and some implicit data and then adding an ontological layer. Still, the relationships we were using were quite simple.

What happened next? The next big thing we did was g3doc, wherein we integrated documentation with the codebase:

- Much easier for engineers to write docs--and more motivation.
- Docs are automatically organized in the same structure as the codebase
- They are also accessible directly from the codebase.
- There's a lot of implicit metadata there.
- There's also easy ways for doc creators to add other metadata like doc type, audience, programming language, etc.

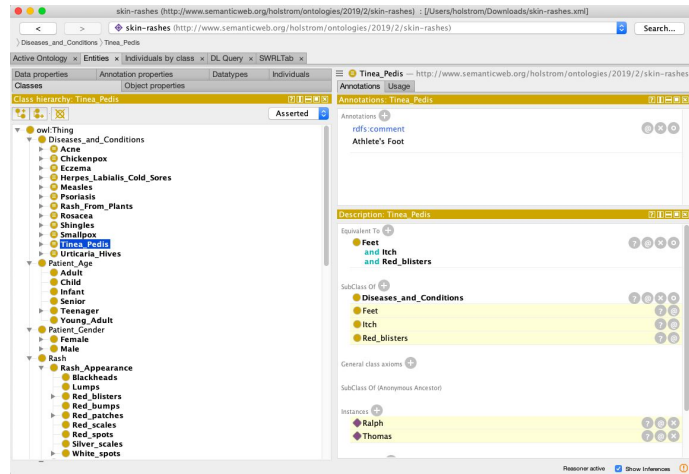
Questions

Was the glossary an ontology? Sifto? Dexy?

Can you name some triples that would make sense in this context?

What were the competency questions for these projects?

Example: Diseases in Protege



UNIVERSITY of WASHINGTON



Make sure to cover:

- Basic taxonomy
- Equivalence statements
- Disjoint
- Individuals
- Inference and reasoner
- Show XML output

Go to cancer ontology to show predicates.

RDF, RDFS, and OWL

- **RDF:** Resource Description Framework
- **RDFS:** Resource Description Framework Semantics
- **OWL:** Web Ontology Language

(As you move from RDF to OWL, you get richer and richer semantics.)

Summary

- Ontologies are powerful and unique because of rich semantic relationships.
- Building an ontology is a multi-step, iterative process. No single process is “correct.”
- Ontologies can be used to achieve many different goals in different domains.