Modeling Tutorial

LD4P RareMat / ARTFrame Meeting
Columbia University
January 11-12, 2018

Outline

- Context and definitions
- Goals
- Modeling process
- Modeling example (family)
- Basic modeling concepts
- Modeling exercise - first draft
- Modeling principles and strategies
- Modeling exercise - second draft - revisions informed by principles and strategies
- Discussion and assessment

Context and Definitions

Motivation

Modeling in the abstract precedes:

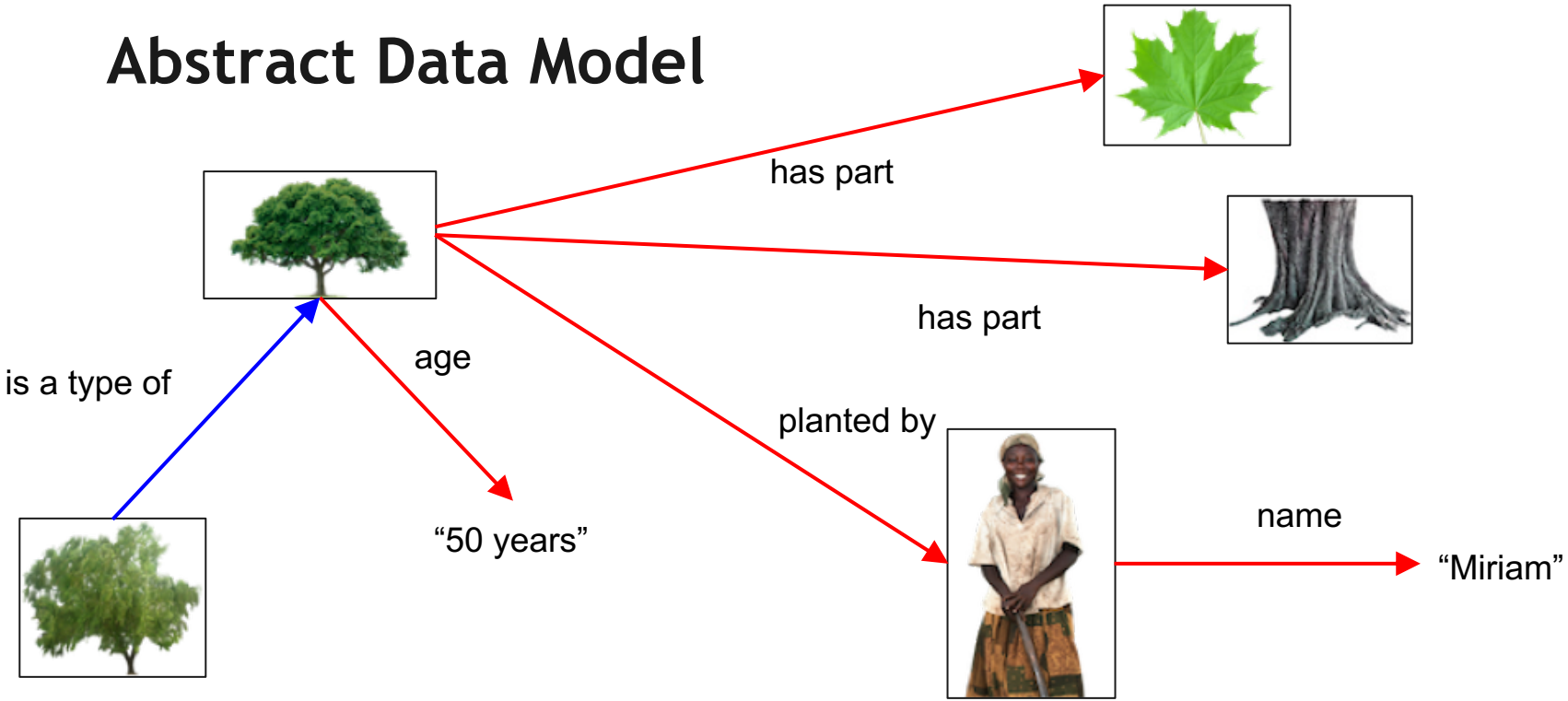
1. Using a language like RDFS or OWL to specify the model, and
2. Writing an OWL file.

From Abstract to Concrete

- **Data model**
  - An abstract conceptual model that identifies things in the real world, their classifications, properties, and relationships to one another.
  - Implementation-neutral: i.e., the model will be the same regardless of whether it is to be implemented as an ontology, a relational database, etc.
  - Most clearly represented as a diagram.
- **Ontology**
  - One type of formal specification of the concepts (terms) in the data model, including names, definitions, and their interrelationships, at an abstract level.
- **RDFS and OWL languages for expressing an ontology** (OWL tutorial)
- **RDFS and OWL language serializations to encode an ontology in a specific syntax** (OWL tutorial)

Real World

Abstract Data Model



Ontology: Abstract Terms

- Tree
- Leaf
- Branch
- Willow
- Person
- Age
- Name
- Planted by
- Has part
- Is a type of

Ontology: Abstract Term Specifications

- Tree
  - Definition: “A woody perennial plant with a single trunk growing to a considerable height.”
  - A tree may have one or more leaves as parts.
  - A tree has exactly one trunk as a part.
  - A tree has an age, but possibly not known.
  - A tree may have been planted by one or more persons.
- Willow
  - Definition: “A tree of temperate climates that typically has narrow leaves and grows near water.”
  - Subclass of Tree - i.e., all willows are trees.

Ontology: Abstract Term Specifications, cont.

- Age: “The length of time that a person has lived or a thing has existed.”
- Planted by:
  - Definition: “Individual placing a seed or plant in the ground so it can grow.”
  - Applies to plants and seeds
  - Inverse: plants

Tutorials

- Modeling Tutorial
  - From data model to ontology
- Connecting Local Decisions with the Larger Ecosystem
  - Discovering existing ontologies for reuse
  - Ontology Design Patterns for integration
- OWL Tutorial
  - Encoding an ontology as RDFS / OWL
  - Written serializations of RDFS and OWL ontologies

Modeling Tutorial Goals

- Understand the process used to design an ontology.
- Understand basic concepts, principles, and strategies for designing an ontology.
- Gain practical experience in building an ontology.
- Understand criteria for assessing an ontology.
- Be prepared for the next step: developing an OWL ontology.

The Modeling Process

Modeling Process

- Identify the knowledge domain.
  - “Smooth” rather than jagged outline.
  - Even if it’s “everything” - schema.org. (Though not ideal.)
- Enumerate and prioritize use cases within that domain.
- Develop the model as a conceptual diagram first.
- Identify concepts / terms from the abstract model.
  - Tree, Leaf, has part, age, etc.
- Use classes, properties, individuals, and relationships to express these concepts.
- Specify constraints on these concepts where relevant.
  - E.g., a tree has exactly one trunk.
- Do ***not*** include: selection of specific terms and namespaces, precise definitions, formulations in RDFS and OWL. This comes at a later stage.

Family Model

Family Model: Use Cases

Use Cases: Brainstorming

- Someone wants to compile a complete genealogy of their family
- Someone wants to provide a description of their household
- Family as patrons
- Order of succession in a royal family
- Evolution of the concept of family, e.g. to include same sex marriage
- Medical family
- Insurance coverage

Family Model: Concepts

Concepts: Brainstorming

Generations	Places (birth/marriage/etc.)	Places (interment/origin)
Parent / Mother / Father	Dates (marriage/divorce)	Grandparent / Grandmother / Grandfather
Person	Reunions	Religion
Gender	Aunt / Uncle	Kinship
Multiples (twins, triplets, etc.)	Age / Birth / Death	Adoption

Concepts: Brainstorming, continued

Marriage / Divorce	Birth order	Partnerships other than marriage
Sibling	Surrogacy	Pets
Child	Step sibling / parent / child	Half sibling
In-laws	Nationality	Race and ethnicity
Languages	Husband / Wife	Cousin
Niece / Nephew	Foster parent / child	

Basic Modeling Concepts

Triples

- In this section we assume RDFS and OWL modeling, though as we've said, a data model is an abstraction independent of the language used to express it.
- In RDF, every statement is expressed as a triple: subject + property + object
    - English example: John ate the apple.

Types of Resources

The building blocks of triples:

- Classes
- Individuals (with or without URIs)
- Properties
- Literals

Resources

Any resource about which somebody wants to say something: e.g., http://example.com/BremerDom.

Types of Resources: Classes

- Classes are a way of defining meaningful groups or sets into which resources can be placed (i.e., classified).
- The Bremer Dom could be a member of a *Cathedral* class.
- Usually identified by a URI.
    - Exceptions made possible by OWL properties (more in OWL tutorial).

Types of Resource: Individuals

- A resource representing a single thing that we want to make assertions about is an *individual, instance*, or *entity.*
  - We avoid the term *instance* due to potential confusion with BIBFRAME Instance.
- An individual may be a member of one or more classes.
  - The Bremer Dom is an individual of the Cathedral class.
- Individuals may be identified by a URI, or not (“blank nodes”).
  - More on blank nodes in OWL tutorial.
- Individuals serve as the subjects and objects of triples.

Types of Resources: Properties

- Describe the characteristics of an individual.
  - E.g., name(s), construction date, etc. of the Bremer Dom.
- State a relationship between two individuals.
  - E.g., the architects and builders of the Bremer Dom.
- Note for the linguistically inclined: properties are also called *predicates*, although they correspond to the verb in a natural language sentence rather than the entire predicate.

Literals

- Text strings
  - The building name “Bremer Dom” is a literal.
- Can be typed: dates, integers, decimals, strings, etc.
- May also have a language.
  - E.g., “Bremer Dom” (German) vs “Bremen Cathedral” (English)
- Can be objects but not subjects in an RDF triple.

Context Nodes

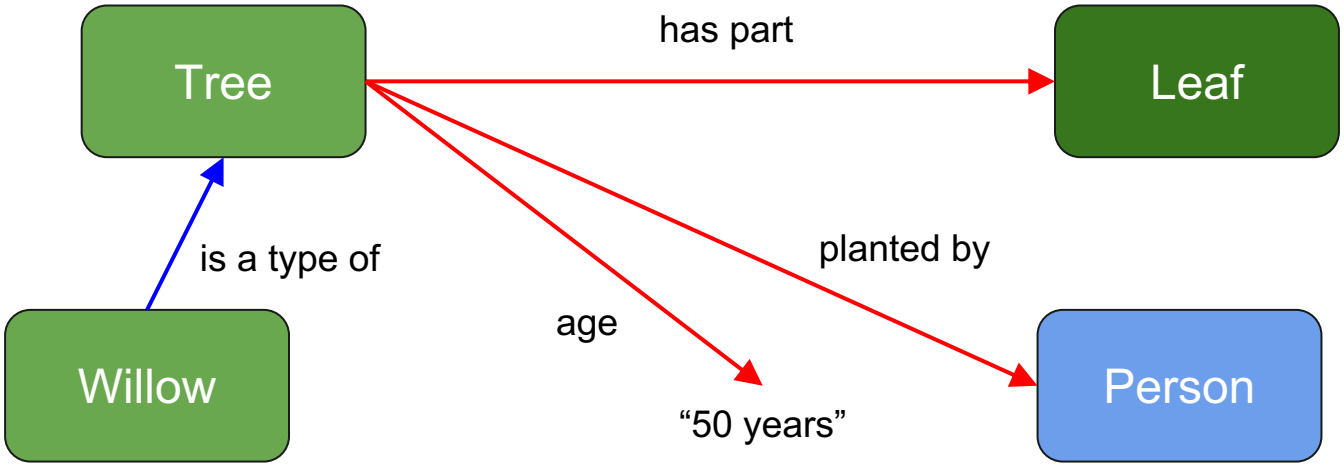
- Represent a relationship between two individuals as another resource rather than simply a property.
- Used to supplement the information that can't be stored using direct relationships between primary entities.
- Examples
  - A marriage could be modeled as a context node that has relationships to two persons, and also has a date, location, etc.
  - ARTFrame Awards model uses an AwardsReceipt type to store information about the date and location of receipt of the award by an agent or bibliographic resource.

More information:
https://wiki.duraspace.org/display/VTDA/Practical+Ontology+Design+Principles+in+the+VIVO+context

Small Group Work

Model development

Task 1: Design the Data Model



Task 2: Abstract Term Specification

- Tree
  - Rough definition: “A woody perennial plant with a single trunk and growing to a considerable height.”
  - A tree may have one or more leaves as parts
  - A tree has exactly one trunk as a part
  - A tree may have an age
  - A tree may have been planted by one or more persons
- Etc.

Review of Small Group Work

Modeling Principles and Strategies

Art vs Science

- **Modeling is more art than science.**
  - Based on experience and practice rather than strict application of rules.
- It's not necessarily a case of hard-and-fast rules but thoughtfully applying principles, best practices, and strategies.
- Many choices depend on
  - Context, use cases, knowledge domain
  - How the data will be queried - what users want to know
  - Weighing one good against another
    - Expressivity vs simplicity
    - Flexibility vs power (inferencing)
    - Reuse and alignment potential vs the best model (more from Steven)
- On the other hand, sometimes one solution really is objectively better than another.

Start from the Data

- **Start from the data, not from a previous representation of the data.**
- Don’t be overly influenced by existing representations of the knowledge domain that you may be familiar with.
- Use these to dig out what the data is, but not necessarily how it should be modeled.
- Be open to examining this data to determine whether there is a real use case for modeling it, and what the priority of that use case is.

Class vs Relationship: Is-a vs Has-a

- **Is a concept best expressed as a class or a relationship (property)?**
- Class = inherent feature of a resource
- Relationship = relation between two resources
- Sometimes depends on context and use cases. E.g., parent
  - Genealogy - relationship
  - PTA / Tax credits and exemptions for parents / Family leave policy - class
- Sometimes one is clearly not useful. E.g., birth parent / child vs adoptive parent / child
  - Everyone is a birth child of someone.
  - Someone can be a birth parent of one child and an adoptive parent of another.
  - So class assignments are misleading and/or redundant.

Object vs Datatype Properties

- **Is a property best expressed an an object property (relationship) or a datatype property?**
- Structured data is generally preferable to unstructured data.
  - Queryable
- Object properties are required when either:
  - There is more to say about the object of the property
  - You want to use a controlled vocabulary as the range of the property
- Some data is truly literal data: dates, names, codes, ages, etc
  - Infinite or near-infinite variability

Atomic vs Composite Values

- **Rely on structured data rather than parsing tools.**
- “Given name” and “family name” is preferable to “name”.
  - No parsing needed.
  - Can easily find all persons with the same family name.
- RDA pre-composed content types
  - “Cartographic tactile three-dimensional form” is a black box - no relationship to “cartographic,” “tactile,” or “three-dimensional” resources.
  - To find all the cartographic resources, you need multiple queries and you need to know the entire list of types.

Context Nodes: Direct vs Indirect Relationships

- In RDFS and OWL, it is not possible to make assertions about properties.
- So if there is more to say about the relationship between two resources, the model should define a context node to apply that data to.
- This may be counterintuitive given the way we normally think about relationships.
- Examples:
  - Marriage
  - ARTFrame / RareMat Awards
  - bibliotek-o Activities

Generalize

- **Define terms as generally as possible without losing needed expressivity.**
- Tree example: “has part” vs “has leaf” + “has trunk” + “has branch” + “has root” + …
- Consider the context and the queries.
- “Has part” allows us to query for all the parts of a tree, without knowing an entire set of predicates.
- You can still query for ***only*** the leaf parts of a tree, not all its parts, by including the object type in the query (“give me all the parts of this tree that are leaves”).
- Sometimes it makes more sense to define different whole-part relationships.
    - A concerto has movements, and it has instrumental parts.
    - When querying for parts of the concerto, you have one or the other in mind and don’t want to fetch all.

Eliminate Redundancy

- Don’t build domain / range into predicate where it’s clear from subject / object type
  - NO: Work hasWorkTitle Title, Instance hasInstanceTitle Title
  - YES: Work or Instance + hasTitle
- Don’t build subject type into object type
  - NO: Work hasTitle WorkTitle and Instance hasTitle InstanceTitle
  - YES: Work hasTitle Title, Instance hasTitle Title
- Provide only one way to represent a relationship
  - Otherwise, two queries are necessary to make sure we get all the data we want.
  - E.g., BIBFRAME types works via either:
    - rdf:type + Work subclass
    - bf:content + bf:Content type

Assertion vs Inference

- Assertion: a statement explicitly added to a data set or ontology
- Inference: a statement derived from existing data based on the ontology specification
- Subclass example
  - Ontology assertion: All Willows are Trees (Willow is a subclass of Tree)
  - Instance assertion: w is a Willow
  - Inference: w is a Tree (no need to assert explicitly)
- Are there examples from the Family Model where the assertion vs inference distinction is applicable?
  - Don’t worry here about ***how*** to express the ontology assertions.
- More detail in the OWL tutorial.

Model vs Application

- **Distinguish what belongs in the model from what belongs in the application.**
- The data model should not be distorted to support UX design.
- Example: Artist's title
    - Conceptually not a type of title but the source of the title (a single person both created the work and gave it the title).
    - But catalogers want to view and edit this as a type of title.
    - The UI can represent this as a title type, while behind the scenes it does not type the title but creates a source relationship.
    - We get both a semantically sound data model ***and*** a UI the catalogers can work with.

Assessment

Assessment of the Model

- Does it represent a defined and coherent knowledge domain?
- Does it generally adhere to known principles, strategies, and best practices?
- Does it satisfy the (most important) use cases?
- Can we express the concepts we identified?
- Does it have a good balance between expressivity and simplicity?
- Can we query the data as simply as possible to get what we want?
    - Consider the SPARQL you would write to address the use cases

Assessment: Community Adoption

- Does it have strong potential for community adoption?
  - Does it contribute to modeling of this domain rather than duplicating existing work?
  - Does it provide a good balance of complexity against use case coverage?

The End