

Ontolingua Tutorial

Adam Farquhar
Knowledge Systems Lab
Stanford University

Schedule - Day 1

Day 1

09:00-10:30 Introduction; what is an ontology? Promise of ontologies. How can we describe ontologies? How can we use and reuse ontologies? What sorts of ontologies are currently available?

10:30-11:00 *break*

11:00-12:00 Modeling. Conceptual modeling versus object-oriented modeling. Reusing knowledge versus reusing code. Tools for building ontologies. Using Ontolingua. Problem definition for hands-on sessions.

12:00-13:00 *lunch*

13:00-14:30 Hands-on I: Ontology Building

14:30-15:00 *break*

15:00-15:30 Review Progress. Tool use and modeling.

15:30-16:30 Hands-on II: Ontology Building

16:30-17:30 Review Progress. Tool use and modeling.

Schedule - Day 2

Day 2

09:00-10:30 Supporting technologies: ontology servers, GFP, translation.

10:30-11:00 *break*

11:00-12:00 Research and development directions.

12:00-13:00 *lunch*

Ontology - What Is an Ontology?

- To communicate, plan, think we need a conceptualization of the world
 - » What kinds of things are there? What are their properties? What are their relationships?
 - » These things define our *ontology*
- We all have ontologies (e.g., of organizations, computers, animals)
 - » Some are very idiosyncratic. Some are shared!
- Communication and interaction require common shared ontologies

Ontology - Problems in Communication

- People, organizations, software programs must communicate
 - » Different needs and backgrounds imply different viewpoints, assumptions, jargon
 - » This divergence is natural and valuable
 - » But leads to problems in communication, interaction, and understanding
- Explicit ontologies are crucial for
 - » Communication
 - » Education
 - » Interoperation
 - » Integration
 - » Adaptive agents

Ontology - Example

- Researchers in molecular biology need to share results and check consistency between their models, data, and reported models and data
- The Riboweb project (Stanford, SMI)
 - » Building an ontology for ribosomes, models, data, reports
 - Molecular structure, experimental data, tests, ...
 - » Encoding (by hand) relevant literature

Ontology - Example

- Doctors, clinics, hospitals, insurance companies, government agencies need to share information
 - » Clinical guidelines, drug interactions, covered procedures, best practices
- Several efforts are addressing aspects of this problem
 - » UMLS (unified medical language system)
 - » SNOMED (standard nomenclature for medicine)

Ontology - Example

- There are many workflow management systems available
- In order to share information across them and support interoperability, we need to define an integrated ontology that covers
 - » Processes, resources, products, services, organizations
- Several groups are involved in such an effort
 - » NIST, WfMC, PIF, TOVE

Ontology - Example

- Collaborative engineering projects need to communicate across discipline boundaries
- Several projects (e.G., PACT, Boeing) have worked to build ontologies for the subdisciplines and span them
- Goals include:
 - » Automated notifications on design modifications
 - » Cross-disciplinary simulation
 - » Improved design process

Ontology - Example

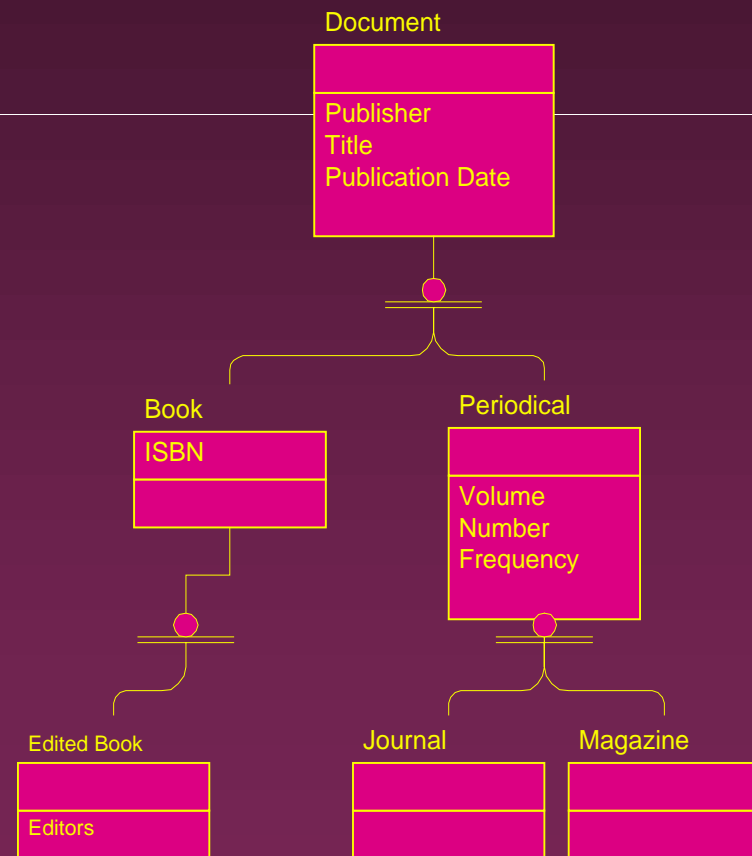
- Natural language understanding and machine translation need large ontologies of
 - » Linguistic categories
 - » Roles
 - » Common-sense objects
- Several projects (WordNet, Pangloss, CyC) have built large, sparse ontologies covering a large portion of common vocabulary

Ontology - Use Versus Reuse

- Large sparse ontologies seems to be widely reusable
 - » They represent a large investment
 - » They make very few commitments
- Small detailed ontologies seem to be most useful
 - » They may represent a large investment of careful research
 - » They make key commitments

Ontology - Example

- What is your ontology for documents?
- What sorts of things does it include?
- What important subfields does it touch?



Ontology As Artifact

- Our ontologies are mostly implicit, hidden
 - » This works when they are already shared within a community
- An explicit ontology is an artifact that must be constructed, structure, manipulated
- Many artifacts have a strong ontological flavor
 - » Glossary, dictionary, encyclopedia
 - » Data dictionary, class library, database schema
 - » Knowledge base

Ontology As Artifact

- An explicit ontology becomes an artifact itself
- Artifacts have incidental properties
 - » Syntax
 - » Design decisions
 - » Scoping decisions
- A good choice of representation language can reduce the incidental properties, but they can never be eliminated
- Do conceptualizations have the same sorts of incidental properties?

Ontology - Summary

- Explicit ontologies support
 - » Shared understanding among people
 - » Interoperability between tools
 - » Systems engineering
 - » Reusability
 - » Declarative specification

Example: Ontology of Organizations

- Concepts

- » Authority, empowerment, commitment, cooperation, achievement

- Competency questions

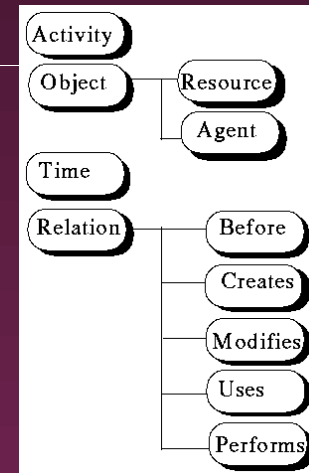
- » What resources does X have authority to assign?
- » Is X allowed to perform activity Y in any situation?
- » What goals is person X committed to achieving?
- » Is a goal achievable by an agent given its current commitments and the commitments of other agents?
- » What goals are solitarily unachievable for a given agent?

Example: Organization Ontology (U. Toronto)

- » Organization unit
- » Organization position
- » Agent
- » Goal
- » Policy
- » Ability
- » Authority
- » Commitment
- » Empowered
- » Achievable goal
- » Solitarily unachievable goal

Example: Process Interchange Format

- PIF is a bridge among heterogeneous process representations
- The core PIF ontology is supported by all targets
- Partially shared views support interoperation between similar systems



Example: Workflow Management Coalition

- A standard terminology is evolving which can serve as a common framework for different workflow management system vendors
- The wfmc glossary
 - » Contains technical definitions for terms to be used in the workflow management coalition specifications and discussions.
 - » Helps to establish consistent usage
 - » For each term the following is provided:
 - Definition, discussion of usage, set of synonyms
 - » The glossary serves as an informal ontology for shared understanding.

Example: Workflow Management Glossary

- Name: process activity
 - » Definition: A logical step or description of a piece of work that contributes toward the accomplishment of a process. A process activity may be a manual process activity and/or an automated workflow process activity.
 - » Usage: A process definition generally consists of many process activities that are connected for the purpose of defining a process flow or state transition network.
 - » Synonyms: step, node, task, work element, process element

KR - Representing Knowledge

- The best understood tool for writing down knowledge is logic
 - » Every sentence has an unambiguous meaning
 - » Combining sentences also has a clear meaning
 - » Of course, we may not anticipate the consequences of a set of sentences!
- But which sentences do we write? What style do we use? What idioms do we use?
- The field of Knowledge Representation (KR) studies methods for writing down knowledge in ways that are computationally useful

KR - Classes and Instances

- A class can be defined by *any* set of objects
 - » The set of people in this room
 - » The set of left-front wheels
- But some sets are worth naming
 - » The class of countries, the class of presidents
- A class is a distinguished named set
- The members of the set are the instances of the class
 - » Bill Clinton is an instance of President
 - » Germany is an instance of Country

KR - Classes and Subclasses

- A subclass is nothing but a subset
 - » Graduate student is a subclass of student
 - Is every graduate student a student?
 - » Ellipse is a subclass of circle
 - Is every ellipse a circle?
 - » A class can have many subclasses
 - Animal
 - Mammal, Reptile, Fish, ...
 - Male, Female
 - Living, Dead

KR - Classes Have Definitions

- Some classes are *completely* defined
 - » A polygon is a triangle iff it has three sides
 - » A vehicle is a bicycle iff it has pedals and has exactly two wheels
 - » Mathematical objects, data structures, human artifacts are often completely defined
- Some classes are *partially* defined
 - » Dog, chair, bridge, house, ship, organization
 - » Sometimes called *primitive concepts*
 - » Partially defined *basic* concepts characterize a large portion of our everyday world

KR - Classes and Meta-Classes

- T-Rex is a subclass of Dinosaur. The T-Rex is extinct. Alice is a T-Rex. Is Alice extinct?
- No! Alice is Dead. The species is extinct.
- Some classes have instances that are classes
 - » The class Extinct-species has species for instances. A species is a class of animals.
- Classes whose instances are classes are sometimes called meta-classes
- Meta-classes are (ontologically) common!

KR - Properties (Own Slots)

- A **slot** is a relation between two things
 - » The age of Alan is 25. A son of Alan is Bob. A son of Alan is Conner.
 - » The average height of Americans is 5'9". The number of buffalo is 5000.
 - » Age-of, son-of, average-height, number are slots
- Slots that belong to individual objects are called **own slots**
- A class can have an own slot, too!

KR - Classes and Template Slots

- Some slots apply to every instance of a class. We call these **template slots**. They are also known as instance slots, or attributes
 - » Age-of is a template slot of Physical-Thing
 - » Son-of is a template slot of Animal
 - » Cardinality-of is a template slot of Set
- Template slots summarize information about every instance
- Attributes in programming languages are a kind of template slots

KR - Facets and Constraints

- A facet says something about a slot on a class. Facets can *constrain* the meaning of a slot.
- Common facets
 - » The value-type of age on people is non-negative
 - » The reverse of son-of for animals is parent-of
 - » The cardinality of biological parents for mammals is 2
- Most programming languages define a small fixed set of facets

KR - Relations, Functions, Slots, and Classes

- The arity of a relation is the number of arguments it takes
- A class is a unary relation!
 - » Person(alan)
- A slot is a binary relation
 - » son-of(alan, bob), age-of(alan,25)
- A facet is a ternary relation
 - » slot-value-type(person, age-of, number)
- A function is just a relation whose last argument is uniquely determined by the others (a.k.a. single valued slot)

KR - Glossary

Frame	Any object, including classes, instances, and relations
Relation	A relation over one or more objects
Class	A distinguished set of objects
Instance	A member of a class
Function	A relation where the last argument is uniquely determined by the others
Slot	A binary relation
Own Slot	A slot belonging to a frame
Template Slot	A slot associated with a class, but with values for instances
Facet	A ternary relation on a frame, slot, value
Constraint	Any assertion that constrains the possible interpretation of frames
Axiom	Any statement taken to be true without proof

Modeling - Ontological v.s. OOP

- Building an ontology appears similar to building an object oriented program, but there are profound differences:
 - » Classes and objects in a program are about data structures
 - » Classes and objects in ontologies are about the world
- There is often a correspondence between data structures in programs and definitions in an ontology

Modeling - Reusing Knowledge or Code

- OOP encourages code reuse

```
class circle {  
    int x; int y; int height; }  
class ellipse extends circle {  
    int width; }  
}
```

- Sometimes code reuse violates ontological principles:

- » Is an ellipse really a subclass of circle?

- Classes in an ontology must reflect the structure of the world, not the structure of the data

Modeling - How Many Chars in a Name?

```
char[20] name;
```

- In a program, it makes sense to define a name as a string of 20 characters
- In an ontology, this is a **bad** idea
- A name is an object with many properties. It is not a string
- Avoid using primitive types as long as possible

Modeling - How Many Bytes in a City?

- In a program, it makes sense to allocate a fixed number of bytes to represent an instance of a class
- In an ontology, this is a **bad** idea
- The ontology is *prior*
- It is not advisable to assume
 - » Attributes will be stored or computed
 - » The order of allocation
 - » Classes will correspond to native classes
- The mapping from ontology to application is neither simple nor obvious

Modeling - Choosing a Language

- The best candidate for an unambiguous language for writing down knowledge is First Order Logic
- But syntax is *not important!*
 - » We can present FOL as an object oriented model
 - » Anything which doesn't fit in the object model can still be said in FOL
- An expressive language makes it easier (possible) to say what we need to say
 - » I'll get you an answer in one hundredth the time, but it will probably be wrong

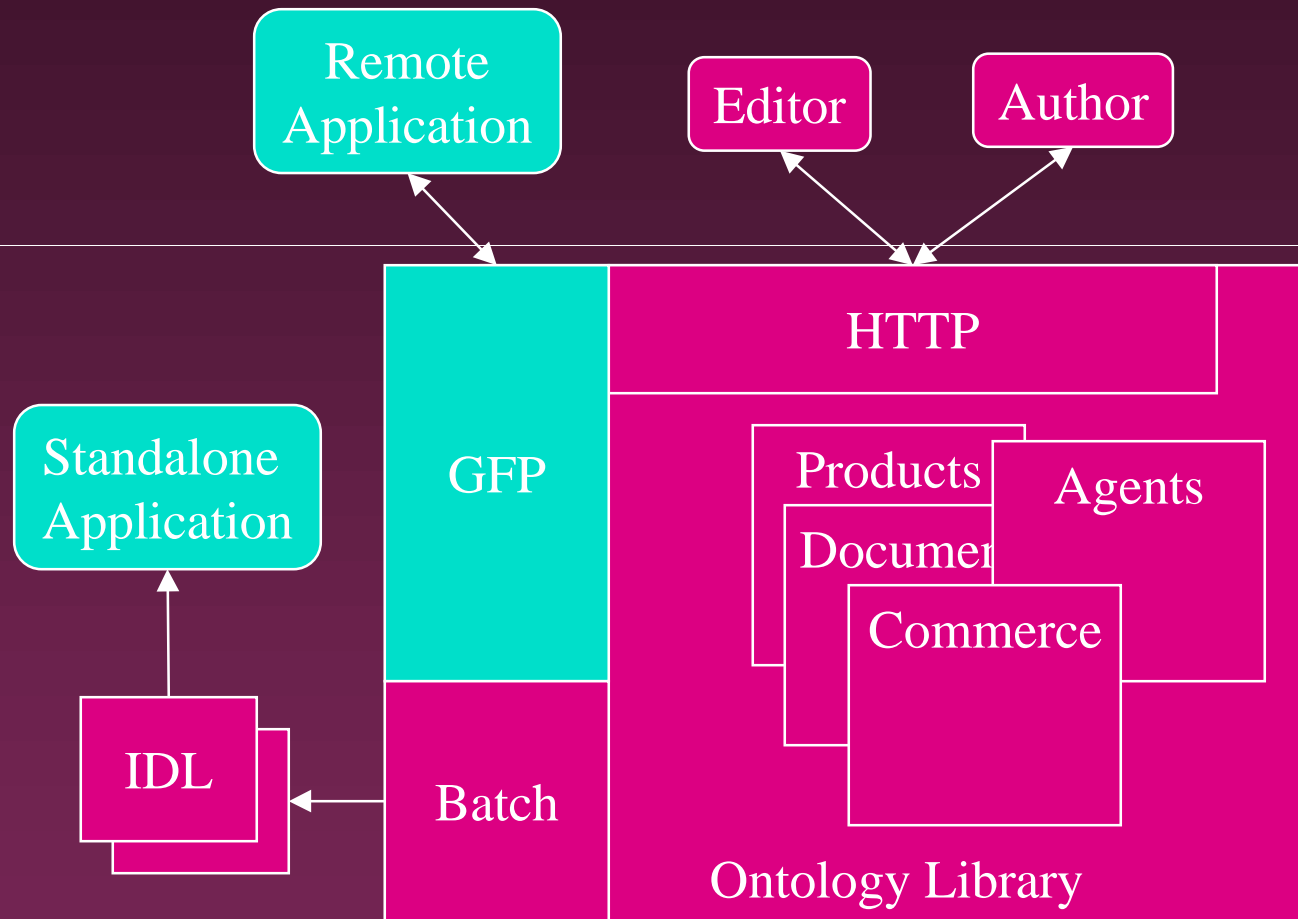
Modeling - Formality versus Precision

- Formal does not mean precise
- Formal does mean that you can understand what you said!
- It is completely possible to make imprecise statements in a formal language
 - » There is a relation between cigarettes and cancer
 - » The English description of vehicle is “A machine that transports stuff from one place to another”
 - » Dog and canine are synonyms
- An imprecise statement may not mean much, but you know what it means!

Modeling - Style

- Names should not encode meaning
 - » bears-of-little-brain, faster-than-a-speeding-bullet
- Try to identify subclasses that *partition* a class
- Try to identify *exhaustive partitions* of a class
- Try to identify multiple exhaustive partitions
- Bundle co-occurring attributes into a class
- Make polymorphic relations and functions

Ontolingua - A Tool for Ontology Use




Ontolingua - Getting Started

- The public server is available at <http://Ontolingua.Stanford.edu>
- Logon
 - » Interact with your web browser
 - » You must have an account to use the system
 - » New accounts are issued immediately
- All work takes place within a *session*, which has a
 - » Duration (logoff cannot be guaranteed)
 - » Description (you can have several)

Ontolingua - Getting Started

- User preferences
- Types of pages
 - » The Library page
 - » The Ontology page
 - » The Frame page
 - A *frame* is a class, instance, relation, or function
 - » The browser page
- What's in a link?
 - » Almost everything is a hyperlink
 - » All words are hyperlinked to their definition page

Ontolingua - The Library



Home comment Prefs File command Quit Reload Dict Docs Bug Find Help

File : Load Ontology Library : All Names

Create : Ontology

Please try to find the time to fill in our [User Interface Survey](#)

Library of Ontologies

Preferred Ontologies

Compartmental-Modeling	[Not loaded]	[Group: JUST-ME]
Federal-Supply-Classification-System	[Not loaded]	[Groups: JUST-ME, JTFLOCAL]
Genesereth-Example	[Not loaded]	[Group: JUST-ME]
Ib-Base-Ontology	[Not loaded]	[Group: JUST-ME]
Ib-Vehicle-Ads	[Not loaded]	[Group: JUST-ME]
Ib-Vehicles	[Not loaded]	[Group: JUST-ME]
Nomenclator-Ontology	[Not loaded]	[Groups: JUST-ME, NOMEN]
Oed-Discussion	[Not loaded]	[Group: JUST-ME]
Onto-Standard-Base.96.09.30	[Not loaded]	[Groups: JUST-ME, ONTOSTD-BETA]
Onto-Standard-Object-Base.96.09.30	[Not loaded]	[Groups: JUST-ME, ONTOSTD-BETA]
Pangloss-Top-Level	[Not loaded]	[Groups: JUST-ME, ONTO-STD]

Unloaded Ontologies

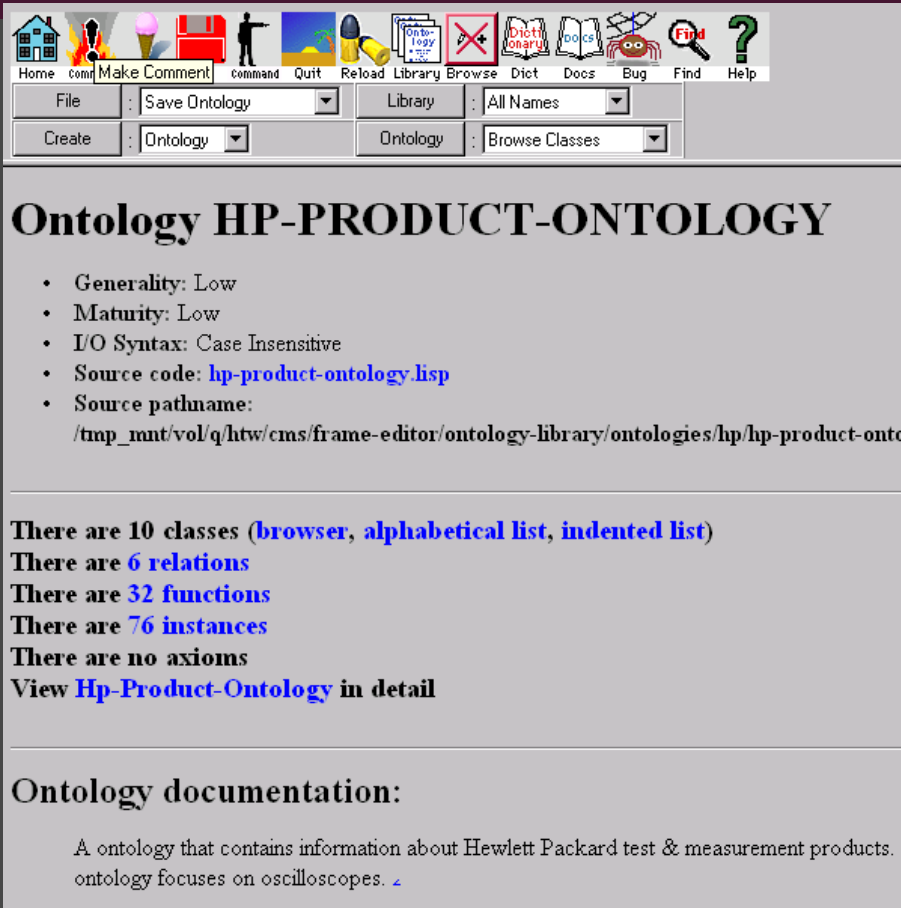
Activities	[Not loaded]	[Group: HTW]
Auto-Guideline	[Not loaded]	[Group: INTERMED]
Bibliographic-Data-In-Verification	[Not loaded]	[Group: HTW]
Car-Rental	[Not loaded]	[Group: UNIVERSE]
Classified-Advertisements	[Not loaded]	[Group: HTW]
Clocks	[Not loaded]	[Group: JTFLOCAL]
Compartmental-Modeling	[Not loaded]	[Group: JUST-ME]
Computer-World	[Not loaded]	[Group: HTW]
Cyc-Interface	[Not loaded]	[Group: HTW]
Cyc-Top-Level-9.20.96	[Not loaded]	[Group: HTW]
Cyc-Upper-Level	[Not loaded]	[Group: HTW]
Database-Metadata-Terramar	[Not loaded]	[Group: SHADE]
Database-Schema-Terramar	[Not loaded]	[Group: SHADE]

Ontolingua - The Library is a Graph

- An ontology can include other ontologies from the library
- Reuse ontological modules!
- You can always augment a definition from a module
 - » But you can never say less!
- Different ontologies may make incompatible extensions

```
+ Hp-Product-Ontology
  - Interface-Ontology
  + Device-Ontology
    Physical-Quantities (open elsewhere)
    Standard-Units (open elsewhere)
  - Interface-Ontology
  + Product-Ontology
    - Abstract-Algebra
    - Frame-Ontology
    - Slot-Constraint-Sugar
    + Standard-Units
      + Standard-Dimensions
        + Physical-Quantities
          - Abstract-Algebra
    + Scalar-Quantities
      Physical-Quantities (open elsewhere)
    - Agents
    - Documents
  Product-Ontology (open elsewhere)
```

Ontolingua - Viewing an Ontology



Home cmd Make Comment command Quit Reload Library Browse Dict Docs Bug Find Help

File : Save Ontology Library : All Names
Create : Ontology Ontology : Browse Classes

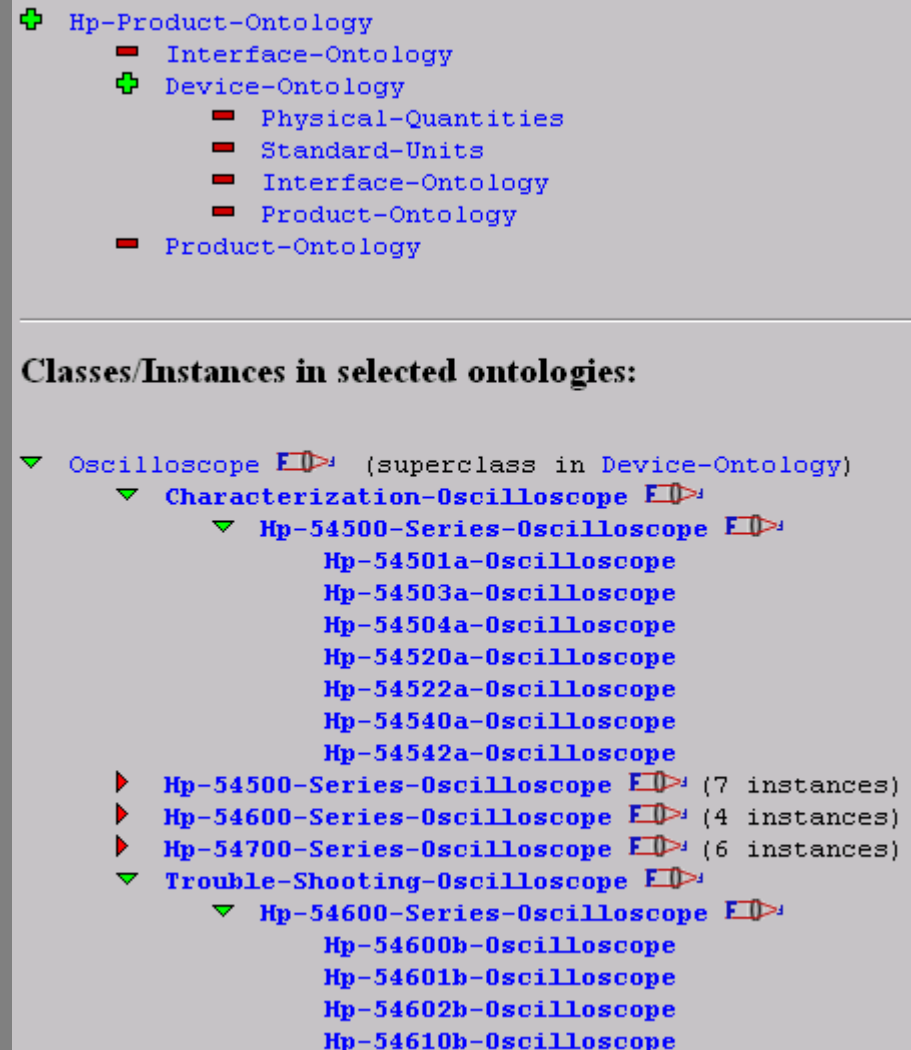
Ontology HP-PRODUCT-ONTOLOGY

- Generality: Low
- Maturity: Low
- I/O Syntax: Case Insensitive
- Source code: [hp-product-ontology.lisp](#)
- Source pathname:
/tmp_mnt/vol/q/htw/cms/frame-editor/ontology-library/ontologies/hp/hp-product-ontol

There are 10 classes ([browser](#), [alphabetical list](#), [indented list](#))
There are 6 relations
There are 32 functions
There are 76 instances
There are no axioms
View [Hp-Product-Ontology](#) in detail

Ontology documentation:

A ontology that contains information about Hewlett Packard test & measurement products. E
ontology focuses on oscilloscopes. ↴



- + Hp-Product-Ontology
 - Interface-Ontology
 - + Device-Ontology
 - Physical-Quantities
 - Standard-Units
 - Interface-Ontology
 - Product-Ontology
 - Product-Ontology

Classes/Instances in selected ontologies:

- ▼ Oscilloscope (superclass in Device-Ontology)
 - ▼ Characterization-Oscilloscope
 - ▼ Hp-54500-Series-Oscilloscope
 - Hp-54501a-Oscilloscope
 - Hp-54503a-Oscilloscope
 - Hp-54504a-Oscilloscope
 - Hp-54520a-Oscilloscope
 - Hp-54522a-Oscilloscope
 - Hp-54540a-Oscilloscope
 - Hp-54542a-Oscilloscope
 - ▶ Hp-54500-Series-Oscilloscope (7 instances)
 - ▶ Hp-54600-Series-Oscilloscope (4 instances)
 - ▶ Hp-54700-Series-Oscilloscope (6 instances)
 - ▼ Trouble-Shooting-Oscilloscope
 - ▼ Hp-54600-Series-Oscilloscope
 - Hp-54600b-Oscilloscope
 - Hp-54601b-Oscilloscope
 - Hp-54602b-Oscilloscope
 - Hp-54610b-Oscilloscope

Ontolingua - Viewing a Class

Class **Tangible-Product**

- Defined in Ontology: [Product-ontology](#)
- Source code: [product-ontology.lisp](#)
- Source pathname: `/tmp_nmt/vol/q/htw/cms/frame-editor/ontology-libr`

Arity: 1

Documentation: A Product that has physical extent. Contrast with Service.

Domain-Of: [Maximum-Storage-Temperature](#), [Minimum-Storage-Temperature](#)

Instance-Of: [Class](#), [Go](#) [Relation](#), [Go](#) [Set](#)

Subclass-Of: [Product](#)

Slots:

[Maximum-Storage-Temperature:](#)

[Minimum-Storage-Temperature:](#)

Axioms for **Tangible-Product**:

Frame References to **Tangible-Product**:

In class [Product](#):

Subclass-Partition: [Go](#) {[Service](#), [Tangible-Product](#)}

Ontolingua - Viewing a Relation

Relation Has-Model-Number

- Defined in Ontology: [Product-ontology](#)
- Source code: [product-ontology.lisp](#)
- Source pathname: `/tmp_mnt/vol/q/htw/cms/frame-editor/ontology-library/ontologies`

Arity: 2

Documentation: A relation that indicates the model number of a product.

Domain: [Product](#)

Instance-Of: [Binary-Relation](#), [Relation](#), [Go](#) [Set](#)

Range: [Model-Number](#)

Function Maximum-Storage-Temperature

- Defined in Ontology: [Product-ontology](#)
- Source code: [product-ontology.lisp](#)
- Source pathname: `/tmp_mnt/vol/q/htw/cms/frame-editor/ontology-library/ontologies`

Arity: 2

Documentation: The maximum temperature at which this object should be stored.

Domain: [Tangible-Product](#)

Instance-Of: [Binary-Relation](#), [Function](#), [Go](#) [Relation](#), [Go](#) [Set](#)

Range: [Temperature-Quantity](#)

Ontolingua - Modifying a Definition



- Make sure that you are in edit mode!
- Edit pens allow you to modify values
- Addition gadgets allow you to add slots, facets, values

Class Product

- Defined in Ontology: [Product-ontology](#)
- Source code: [product-ontology.lisp](#)
- Source pathname: [/tnp_mnt/vol/q/htw/cms/frame-editor/ontology-library/ont](#)

Arity: 1 [+Facet](#)

Documentation: [A class of objects that are typically bought and sold.](#) [+Value](#) [+Facet](#)

Domain-Of: [Has-Model-Number](#), [Has-Special-Discount](#), [Has-Warranty](#), [List-Price](#) [+Facet](#)

Instance-Of: [Class](#), [Go Relation](#), [Go Set](#) [+Value](#) [+Facet](#)

Range-Of: [Object-Sold](#), [Product-With-This-Warranty](#) [+Value](#) [+Facet](#)

Subclass-Of: [Individual-Thing](#) [+Value](#) [+Facet](#)

Subclass-Partition: [{Service, Tangible-Product}](#) [+Value](#) [+Facet](#)

Superclass-Of: [Product-Previously-Owned](#), [Service](#), [Service-Agreement](#), [Tangible-Product](#) [+Class Slot](#)

Slots:

Associated-Documents: [+Value](#) [+Facet](#)

Has-Model-Number: [+Value](#) [+Facet](#)

Has-Special-Discount: [+Value](#) [+Facet](#)

Has-Warranty: [+Value](#) [+Facet](#)

List-Price: [+Value](#) [+Facet](#)

Net-Weight: [+Value](#) [+Facet](#)

[+Instance Slot](#)

Ontolingua - Web Interface

- No pull-down menus, so we use a selection menu together with a submit button
- There are separate editing and viewing modes
 - » Toggle using the browse/edit toggle button to switch between them
- Typically, you only edit one object at a time
- The edit mode looks like the browse mode, but with type-in fields snipped into the page

Ontolingua - Basic Principles

- Presentation versus representation
 - » Object-Oriented presentation, full logical representation
- Creating a new object is different from saying new things about existing objects
- Include existing ontologies from the library
- Augment, but do not edit included definitions
- Measure twice, cut once!

Ontolingua - the Frame Ontology

- The Frame Ontology defines the basic concepts for a rich object oriented representation
- Almost all ontologies should include the Frame Ontology
- Things
 - » A thing can be anything, including a set
 - » An individual-thing is not a set
 - » Most common-sense things are individual-things according to the frame ontology

Class
All-Instances
Arity
Subclass-Of
Subclass-Partition
Class-Partition
Facet
Thing
Individual-Thing

Ontolingua - How do I Save My Work

- Your work is stored on the server
- You may download your work to a local file system using your browser
- You may email your work to any email address
- Any ontology may be translated into several different target languages

Ontolingua - Tips

- Once you have created a subclass, you can use `frame>copy` to create siblings
- Do not reference a frame before creation
- Completion will help with long names
- Find will help you find frames in any loaded ontology
- You can edit anything except the basic type of an object after creation
- Rename a frame or ontology by editing its name

Ontolingua - Tips

- Values may include strings containing html
- Strings are automatically hyperlinked
 - » Use a frame name with dashes, or put it in all caps
 - » Handle plurals in text with 's (tangible-product's)
 - » To hyperlink against a definition use the word tag in an anchor `Object`
- You can hyperlink a subtree against an ontology
 - » `Help`

Ontolingua - Tips

- The web interface handles domain differently in different contexts
 - » When creating a slot, multiple values for domain are interpreted as a union
 - » When creating a general relation, domain values are assigned to arguments in turn
 - » Adding a value to the domain slot of a relations is simple conjunction (as are all multiple values)

Ontolingua - Tips

- It may appear that arity is handled inconsistently between relations and functions
 - » $R(x,y)$ - R has arity 2
 - » $F(x,y)$ - F has arity 3!
- The term $F(x,y)$ is equivalent to the predicate $z=F(x,y)$, which can be written in relational form
 - » $F(x,y,z)$ - F does have arity 3

Design - Basic Procedure

- Sketch out objectives
- Identify example scenarios
- Identify core competency questions
- Sketch out the space
- Incorporate knowledge sources
- Identify possible design decisions
- Iterate, increasing coverage and fidelity
- ❖ This is a pragmatic, effective procedure with little methodological overhead

Design - Sketch out Objectives

- How much coverage is required?
- What tasks are anticipated?
- Do existing sources need to be considered?
- What depth of axiomatization is needed?

Design - Identify Example Scenarios

- Good scenarios will
 - » Appeal to a sufficiently broad community
 - » Include a variety of tasks
 - » Not be too detailed - narrow focus will produce a narrow brittle ontology
 - » Be useful for documentation and understanding in addition to guiding design
 - » Help readers and developers understand the scope and intentions of the authors
- Story problems with intuitively understandable solutions

Design - Competency of an Ontology

- Identify competency questions
 - The ontology must define the knowledge necessary to answer the questions
 - Don't worry about how the questions will be answered!
- For documents:
 - » What is the difference between a particular book, the first edition of a book, the second edition of a book, and a translation into another language?
 - » Find all books in a collection that have been written by McCarthy
 - » Find all books whose publisher was in San Francisco, but is no longer in business

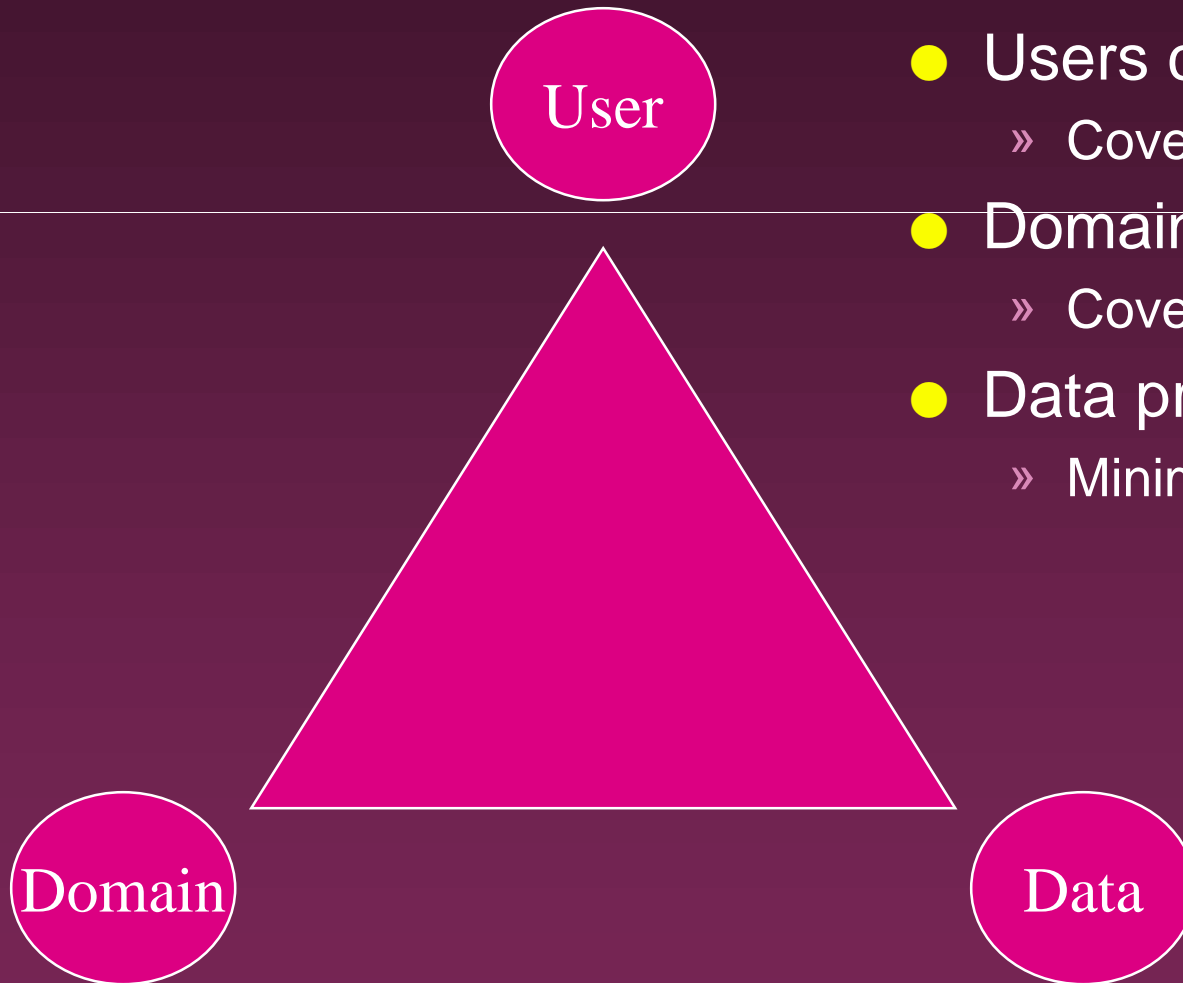
Design - Sketch out the Space

- What are fifty key concepts?
- What are some of their properties?
- What are some of the relations between them?

Design - Find and Incorporate Sources

- Almost all interesting domains already have extensive existing descriptions
 - » Text books
 - » Professional societies
 - » Data dictionaries
 - » Standards documents
 - » Encyclopedias, thesauri, dictionaries
 - » Earlier attempts at formalization (data dictionaries, metadata, standard codes)
- Due diligence pays off in buy-in, acceptance, completeness
- Examples: Standard Industry Codes (SICS), Federal Classification System (FSC), ASME publications

Design - Triad



- Users contribute tasks
 - » Cover 80%
- Domains are infinite
 - » Cover standard terms
- Data provides lower bound
 - » Minimal complete coverage

Practicum - Getting Started

- Customize browser settings
 - » Check document every time
 - » Small, readable font
 - » Minimal menubar to maximize vertical space
- Create an account
- Customize user preferences
 - » Menus>Full
 - » Ontology Library>accessible in header
 - » Images>Show unbordered icons

Practicum - Basic Use

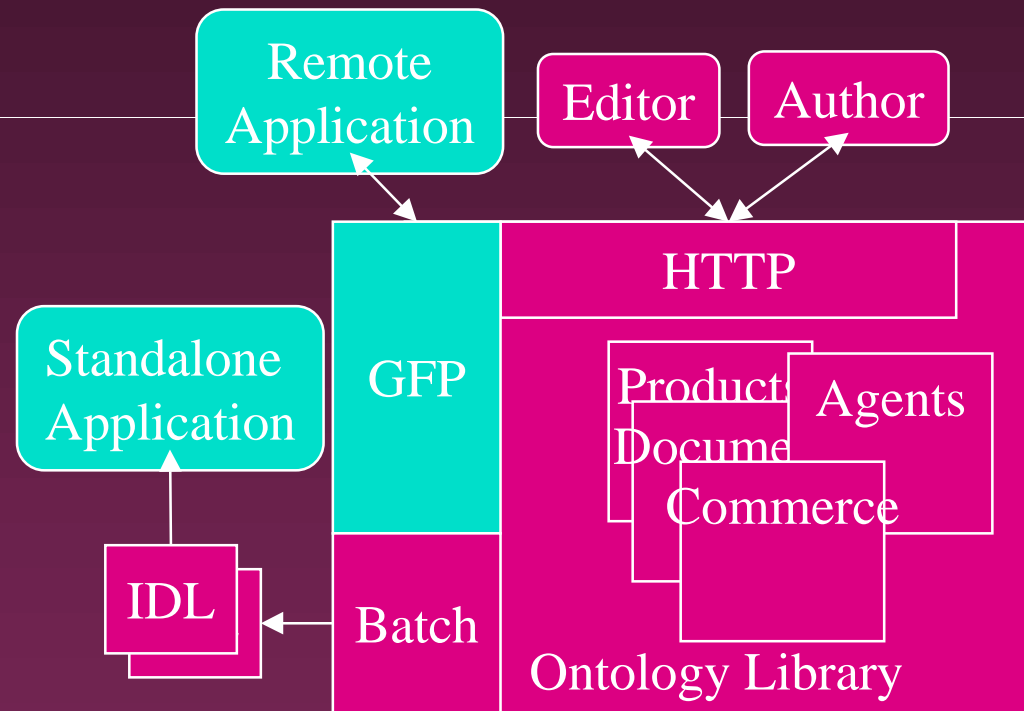
- Look at an ontology in the library
 - » HP-Product-Ontology
 - » Use class browser/slot browser
 - » Try focusing on Characterization-Oscilloscope
- Create a new ontology
 - » Return to library page first!
- Create a class
- Create a subclass (undo/redo)
- Create a slot on the class
- Assert a facet on the slot

Practicum - Getting Serious

- Follow ontology design steps
- Sketch out (on paper) an ontology of documents for your task
- Enter document ontology
- Refine and extend ontology

USE - Ontology Server Architecture

- Three modes of use
 - » Runtime query
 - » Translation
 - » Development



Use - Runtime Query

- Remote applications may query an ontology server
 - » To determine if a term is defined
 - » To determine the relationship between terms
 - » To manipulate the contents of an ontology
- Typical applications include
 - » Browsing and editing tools
 - » Ontology analysis tools
 - » Knowledge based applications
 - » Mediators

Use - T-Helper

- An AIDS treatment protocol advisor developed at Stanford's SMI
- Matches patients with treatment protocols
- Queries the ontology server to determine relationships between terms
 - » Is zidovine an antiretroviral drug?
- Uses a substantial ontology of drugs and treatments
- Uses the generic frame protocol (GFP) to access the server

Use - GFP

- The Generic Frame Protocol (GFP) provides a common object oriented API to knowledge representation systems
 - » CLIPS, Classic, Theo, Loom, Ocelot, Sipe, Ontolingua, Clos, ...
- Features
 - » Queries, updates, network access
 - » Multiple language bindings (C, Lisp, Java)
 - » Function specifier language for server-side execution
 - » Layered architecture

Use - GFP Knowledge Model

- Similar to the representation defined earlier
- Frames can be class, slot, facet, instance, value
- Assertions are conjunctive
- Multiple inheritance
- Open set of facets for defining constraints

Use - GFP Operations

get-kbs
find-kb-of-type
get-frame-matching
get-class-direct-supers
get-class-supers
get-instance-direct-types
get-slot-value
get-slot-values
get-frame-slots
get-frame-facets
get-facet-values
put-slot-values
put-facet-values
add-slot-value
delete-frame
create-kb

- GFP defines a wide range of operations
- Access to data and schema
- Access to classes, instances, and metaclasses
- The underlying system need not be a KB!
- The underlying system need not be object oriented

Use - GFP Layered Architecture

- Full protocol defines about 200 operations
- For a simple read-only back end, about 15 operations must be defined
- For a simple read-write back end, about 35 operations must be defined
- All other operations have default implementations
 - » get-kb-classes can be defined using get-kb-frames and class-p
 - » add-slot-value can be defined using get-slot-values and put-slot-values

Use - GFP Optimization

- Network protocols need to provide methods for achieving reasonable performance
- Caching
 - » The Lisp and Java clients implement caching-network-kbs that cache the results of all GFP operations
 - » Prefetching is currently being developed
- Server-side Execution
 - » Function specifiers allow for compound operations to be executed on the server

Use - GFP C Code

```
if(kb = find_ontology("FRAME-ONTOLOGY", connection)) {
    frames=gfp_get_class_direct_subs(frame, kb, gfp_true);
    if(frames){
        slots=gfp_get_frame_slots(frame, kb,
                                   gfp_all, gfp_all, gfp_true);

        if(slots){
            print_frame(frame,slots,frames);}
        }}
if(glast_error_code!=GFP_NO_ERROR)
    explain_error();
```

Use - GFP Java Code

- GFP KB is the primary class
 - » CachingKB, NetworkKB, TupleKB
- Currently developing Java editing apps using

```
kb.get_slot_values (frame, slot,  
                   GfpKb._T,      // local-only-p  
                   GfpKb._auto, // slot-type  
                   GfpKb._all); // #values
```

Use - Translation

- Translation is a challenging problem
 - » Semantics - ensure that the meaning is preserved
 - » Syntax - ensure that target syntax is correct
 - » Style - ensure that target idioms are preserved

Use - Current Translation Technology

Class c2RailBridge

- Defined in Ontology: [Target-schema_1.0](#)
- Source code: [target-schema_1.0.lisp](#)
- Source pathname: [/tmp_mnt/vol/q/htw/cms/](#)

SUBCLASS-OF: [c2RailLink](#)

SUPERCLASS-OF: [c2RRBrdg](#)

INSTANCE-OF: CLASS, INTERFACE, [Go](#) [BOU](#)
ARITY: 1

Slots:

location:

SLOT-VALUE-TYPE: [Go](#) [string](#)

root_name:

SLOT-VALUE-TYPE: [Go](#) [string](#)

root_version:

SLOT-VALUE-TYPE: [Go](#) [string](#)

IDL Translation of c2RailBridge

```
interface c2RailBridge : c2RailLink {  
};
```

Class c2Entity

- Defined in Ontology: [Target-schema_1.0](#)
- Source code: [target-schema_1.0.lisp](#)
- Source pathname: [/tmp_mnt/vol/q/htw/cm](#)

SUBCLASS-OF: [c2Root](#)

SUPERCLASS-OF: [c2Computer](#), [c2Countermeas](#)

INSTANCE-OF: CLASS, INTERFACE, [Go](#) [BOU](#)
HAS-ATTRIBUTE: [location](#)
ARITY: 1

Slots:

location:

SLOT-VALUE-TYPE: [string](#)

root_name:

SLOT-VALUE-TYPE: [string](#)

root_version:

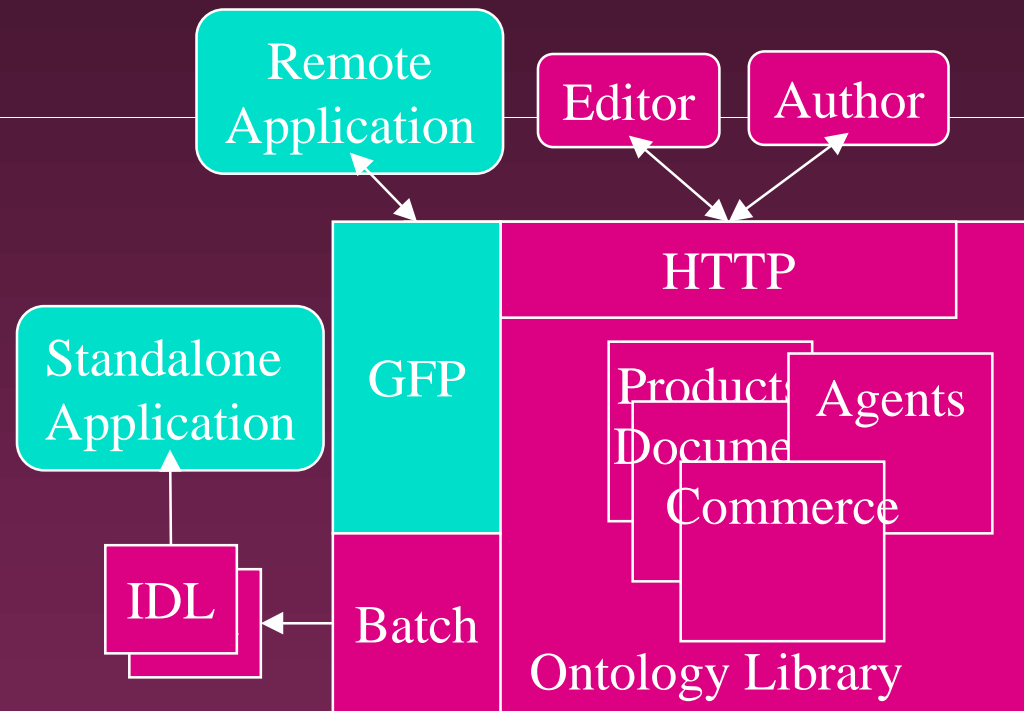
SLOT-VALUE-TYPE: [string](#)

IDL Translation of c2Entity

```
interface c2Entity : c2Root {  
  attribute string location;  
};
```

USE - Ontology Server Architecture

- Three modes of use
 - » Runtime query
 - » Translation
 - » Development



Use - Runtime Query

- Remote applications may query an ontology server
 - » To determine if a term is defined
 - » To determine the relationship between terms
 - » To manipulate the contents of an ontology
- Typical applications include
 - » Browsing and editing tools
 - » Ontology analysis tools
 - » Knowledge based applications
 - » Mediators

Use - T-Helper

- An AIDS treatment protocol advisor developed at Stanford's SMI
- Matches patients with treatment protocols
- Queries the ontology server to determine relationships between terms
 - » Is zidovine an antiretroviral drug?
- Uses a substantial ontology of drugs and treatments
- Uses the generic frame protocol (GFP) to access the server

Use - GFP

- The Generic Frame Protocol (GFP) provides a common object oriented API to knowledge representation systems
 - » CLIPS, Classic, Theo, Loom, Ocelot, Sipe, Ontolingua, Clos, ...
- Features
 - » Queries, updates, network access
 - » Multiple language bindings (C, Lisp, Java)
 - » Function specifier language for server-side execution
 - » Layered architecture

Use - GFP Knowledge Model

- Similar to the representation defined earlier
- Frames can be class, slot, facet, instance, value
- Assertions are conjunctive
- Multiple inheritance
- Open set of facets for defining constraints

Use - GFP Operations

get-kbs
find-kb-of-type
get-frame-matching
get-class-direct-supers
get-class-supers
get-instance-direct-types
get-slot-value
get-slot-values
get-frame-slots
get-frame-facets
get-facet-values
put-slot-values
put-facet-values
add-slot-value
delete-frame
create-kb

- GFP defines a wide range of operations
- Access to data and schema
- Access to classes, instances, and metaclasses
- The underlying system need not be a KB!
- The underlying system need not be object oriented

Use - GFP Layered Architecture

- Full protocol defines about 200 operations
- For a simple read-only back end, about 15 operations must be defined
- For a simple read-write back end, about 35 operations must be defined
- All other operations have default implementations
 - » get-kb-classes can be defined using get-kb-frames and class-p
 - » add-slot-value can be defined using get-slot-values and put-slot-values

Use - GFP Optimization

- Network protocols need to provide methods for achieving reasonable performance
- Caching
 - » The Lisp and Java clients implement caching-network-kbs that cache the results of all GFP operations
 - » Prefetching is currently being developed
- Server-side Execution
 - » Function specifiers allow for compound operations to be executed on the server

Use - GFP C Code

```
if(kb = find_ontology("FRAME-ONTOLOGY", connection)) {
    frames=gfp_get_class_direct_subs(frame, kb, gfp_true);
    if(frames){
        slots=gfp_get_frame_slots(frame, kb,
                                   gfp_all, gfp_all, gfp_true);

        if(slots){
            print_frame(frame,slots,frames);}
        }}
if(glast_error_code!=GFP_NO_ERROR)
    explain_error();
```

Use - GFP Java Code

- GFP KB is the primary class
 - » CachingKB, NetworkKB, TupleKB
- Currently developing Java editing apps using

```
kb.get_slot_values (frame, slot,  
                   GfpKb._T,      // local-only-p  
                   GfpKb._auto,  // slot-type  
                   GfpKb._all);  // #values
```

Use - Translation

- Translation is a challenging problem
 - » Semantics - ensure that the meaning is preserved
 - » Syntax - ensure that target syntax is correct
 - » Style - ensure that target idioms are preserved

Use - Current Translation Technology

Class c2RailBridge

- Defined in Ontology: [Target-schema_1.0](#)
- Source code: [target-schema_1.0.lisp](#)
- Source pathname: [/tmp_mnt/vol/q/htw/cms/](#)

SUBCLASS-OF: [c2RailLink](#)

SUPERCLASS-OF: [c2RRBrdg](#)

INSTANCE-OF: CLASS, INTERFACE, [Go](#) [BOU](#)
ARITY: 1

Slots:

location:

SLOT-VALUE-TYPE: [Go](#) [string](#)

root_name:

SLOT-VALUE-TYPE: [Go](#) [string](#)

root_version:

SLOT-VALUE-TYPE: [Go](#) [string](#)

IDL Translation of c2RailBridge

```
interface c2RailBridge : c2RailLink {  
};
```

Class c2Entity

- Defined in Ontology: [Target-schema_1.0](#)
- Source code: [target-schema_1.0.lisp](#)
- Source pathname: [/tmp_mnt/vol/q/htw/cm](#)

SUBCLASS-OF: [c2Root](#)

SUPERCLASS-OF: [c2Computer](#), [c2Countermeas](#)

INSTANCE-OF: CLASS, INTERFACE, [Go](#) [BOU](#)
HAS-ATTRIBUTE: [location](#)
ARITY: 1

Slots:

location:

SLOT-VALUE-TYPE: [string](#)

root_name:

SLOT-VALUE-TYPE: [string](#)

root_version:

SLOT-VALUE-TYPE: [string](#)

IDL Translation of c2Entity

```
interface c2Entity : c2Root {  
  attribute string location;  
};
```