Protégé-2000: A Flexible and Extensible Ontology-Editing Environment

Natalya F. Noy, Monica Crubézy, Ray W. Fergerson, Samson Tu, Mark A. Musen

Stanford Medical Informatics
Stanford University
Stanford, CA
Facts (maybe)

Fact 1: Ontologies are no longer just for knowledge engineers

Fact 2: A number of new Semantic-Web languages and representation formalisms are emerging; no agreement yet

Fact 3: We are developing ontologies, agents, and applications today, without waiting for the standards
Facts and Requirements

Facts:
- Ontologies are no longer just for knowledge engineers.
- A number of new Semantic-Web languages and representation formalisms are emerging; no agreement yet.
- We are developing ontologies, agents, and applications today, without waiting for the standards.

Requirements:
- Domain experts need to understand and maintain ontologies.
- We need adaptable tools which we can tune to support new languages and formalisms quickly.
- We need suites of tools for ontology development and management.
**A Solution**

- Protégé-2000 is an ontology-editing and knowledge-acquisition environment, which has
  - a graphical and easy-to-use interface
  - a flexible knowledge model
  - an extensible plugin architecture
  - an existing set of plugins for
    - ontology merging
    - acquisition of information from online knowledge sources
    - constraint specification and verification
    - .... (we don’t even know them all)
Protégé-2000 Knowledge-Model Components

- Classes
  concepts in a taxonomic hierarchy

- Instances
  instances of classes

- Slots
  first-class objects representing properties of classes and instances

- Facets
  constraints on allowed slot values, such as cardinality, defaults, allowed classes, and so on.
Ontologies in Protégé-2000
Acquiring instance data
Protégé-2000 Architecture

Knowledge model <-> Storage model

User interface
Changing the Knowledge Model
Changing the Knowledge Model

- Templates for new class-level and slot-level properties
  - metaclasses
  - metaslots
Changing the Knowledge Model

- Templates for new class-level and slot-level properties
- metaclasses
- metaslots
Protégé-2000 Plugin Architecture

- URIs
- Complex expressions
- Primitive and defined classes
- Transitive, symmetric properties

Knowledge model

Storage model

User interface
Changing The User Interface

- Users can replace any widget on the form with a different one.
Changing The User Interface

- Users can replace any widget on the form with a different one.
Protégé-2000 Plugin Architecture

- URIs
- Complex expressions
- Primitive and defined classes
- Transitive, symmetric properties

Knowledge model → Storage model

User interface

- Acquire and verify URIs
- Use a structured editor for logical expressions
- Display inferred values for transitive properties
Changing The Storage Model

- Users can change the output file format - alternative "back ends"

- The back-end code can:
  - resolve the remaining differences in the knowledge model,
  - add or remove information,
  - map between Protégé knowledge model and the required knowledge model
Protégé-2000 Plugin Architecture

- URIs
- Complex expressions
- Primitive and defined classes
- Transitive, symmetric properties

Knowledge model

Storage model

User interface

- Acquire and verify URIs
- Use a structured editor for logical expressions
- Display inferred values for transitive properties

- RDF Schema
- OIL
- XML
- JDBC database
Including New Applications

- Knowledge model
- Storage model
- User interface

API

Other KB applications

Other KB applications

- API

- Knowledge model
- Storage model
- User interface
Include New Applications

Integration With A Description Logics Classifier (FaCT)
Using A DL Classifier
Other Plugins

- Diagrammatic knowledge entry
- Ontology visualization
- Ontology merging
- Ontology acquisition from UMLS and WordNet
- Constraint verification

All these plugins become automatically available for different languages
As A Result, We Get A Tool That

- can be used for ontology development in different (overlapping) representation formalisms
  - translate models from one formalism to another
- can be easily customized to a new language
  - knowledge model
  - user interface
  - persistent storage
- can incorporate other applications
Our Vision

- Complex, distributed systems built from plug-and-play components
- Systems that allow evolution throughout their life cycles via substitution of new components
- Repositories of components for use in new designs and for updating previous applications

for both ontologies and components of knowledge-based systems!
http://protege.stanford.edu