Protégé Plug-in Library: A Task-Oriented Tour

Tutorial at
Seventh International Protégé Conference
Bethesda MD, July 6 2004

Samson Tu and Jennifer Vendetti
Stanford Medical Informatics
Stanford University

Goals

• Give a basic introduction to the Protégé plug-in architecture
• Describe a range of tasks in the life cycle of frame-based knowledge base development using Protégé
• Discuss Protégé plug-ins available for these tasks
• Answer questions
• Not to
  • Cover every single plug-in
  • Give a detailed tutorial on any one plug-in
  • Explain OWL plug-ins
Outline

- Introduction
  - Review of plug-in types (Jennifer)
  - Review of plug-in architecture (Jennifer)
  - Describe example application and ontology (Samson)
- Tasks
  - Conceptualization (Samson)
  - Reusing/importing existing resources (Jennifer)
  - Visualizing ontologies (Jennifer)
  - Customizing displays (Jennifer)
  - Dealing with non-standard data types (Jennifer)
  - Navigating and browsing (Jennifer)
  - Validating integrity constraints (Samson)
  - Querying knowledge bases (Samson)
  - Publishing and exporting knowledge bases (Samson)
  - Developing applications (Samson)
  - Managing projects (Jennifer)
- Conclusion (Samson and Jennifer)

Plug-in Types

- Tab widget
- Slot widget
- Back-end
- Import/Export
- Project
Plug-in Architecture

Where’s my plug-in?

Built-in support for an “About Box”
Plug-in Architecture

Built-in support for documentation

Example scenario

- Develop a medical decision-support application that generates recommendations based on clinical practice guideline
- Guideline example: management of common cold
Conceptualization

- Protégé gives little support
- FCATab: Tab to support ‘Formal Concept Analysis’ (FCA)
  - Really an export plug-in to support use of other tool
  - create ‘context’ table from Protégé classes and slots
  - generate input file for other FCA tool (ConExp)

<table>
<thead>
<tr>
<th>Classes/Slots</th>
<th>ad_name</th>
<th>ad_surname</th>
<th>purchased</th>
</tr>
</thead>
<tbody>
<tr>
<td>T43040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Advertisement</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Article</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line1aths</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Billing_Chart</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Columnists</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content_Line1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Editor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Library</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name_Surname</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newspaper</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personname_All</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Protable_Name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WorldWide</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conceptualization of example domain

- A guideline is a set of recommendations consisting of
  - Contexts (e.g. presentation of symptoms)
  - Tasks
    - Actions (e.g. home care or referral)
    - Decisions: choice of action based on preference criteria (e.g. symptoms of serious problem)
  - Patient state encodes information about a particular patient
    - e.g. Observations, prescribed medications
  - Medical concepts represent abstractions of medical terminology
    - e.g. cough, fever, laryngitis
Example ontology in Protégé

Outline

- Introduction
  - Review of plug-in types (Jennifer)
  - Review of plug-in architecture (Jennifer)
  - Describe example application and ontology (Samson)
- Tasks
  - Conceptualization (Samson)
  - Reusing/importing existing resources (Jennifer)
  - Visualizing ontologies (Jennifer)
  - Customizing displays (Jennifer)
  - Dealing with non-standard data types (Jennifer)
  - Navigating and browsing (Jennifer)
  - Validating integrity constraints (Samson)
  - Querying knowledge bases (Samson)
  - Publishing and exporting knowledge bases (Samson)
  - Developing applications (Samson)
  - Managing projects (Jennifer)
- Conclusion (Samson and/or Jennifer)
Reuse/import of existing resources

Protégé Ontologies Library
- Gene Ontology
- HL7-RIM
- Guideline Interchange Format
- OWL Ontology Library
...

Submissions welcomed and encouraged !!
http://protege.stanford.edu/ontologies/ontologies.html

Reuse/import of existing resources

UMLS Tab Demo
1. Patient reports some combination of the following symptoms:
   - Sore throat
   - Rhinorrhea
   - Cough
   - Fever, usually of < 102° lasting < 72 hours
   - Laryngitis

2. Determine age
   Assess symptoms
Reuse/import of existing resources

- DataGenie – import data from arbitrary databases
- OKBC Tab – import/export OKBC compliant ontologies
- OntoBase – read, navigate, update arbitrary databases
- XML Tab – import arbitrary XML documents
- WordNet Tab – import lexical content from WordNet
- TXRuleML Tab – RuleML to taxonomic class hierarchies
- Jess Tab – Jess scripting for... well... anything really

Visualizing ontologies

- OntoViz Tab – visualize ontologies with GraphViz
- Jambalaya – visualize ontologies with SHriMP (Simple Hierarchical Multi-Perspective)
- TGViz – visualize ontologies with TouchGraph
OntoViz tab demo

Customizing displays

Graph Widget

• Alternative to Forms for creating and populating instances of classes

• Visualize networks of instances and relationships between instances
Graph Widget Demo

Dealing with non-standard data types

- Calendar Widget
- Date Widget
- URL Widget
- Image Widget
- Indirect Instances
Calendar Widget

Date Widget

Date widget automatically enters today’s date
Navigating & Browsing

- **Instance Tree**
  - view instances of classes as root nodes of trees
  - trees contain directly and indirectly referenced frames

- **Knowledge Tree**
  - designate a top-level instance and navigate a tree of “contained” instances
Outline

- Introduction
  - Review of plug-in types (Jennifer)
  - Review of plug-in architecture (Jennifer)
  - Describe example application and ontology (Samson)

- Tasks
  - Conceptualization (Samson)
  - Reusing/importing existing resources (Jennifer)
  - Visualizing ontologies (Jennifer)
  - Customizing displays (Jennifer)
  - Dealing with non-standard data types (Jennifer)
  - Navigating and browsing (Jennifer)
  - Validating integrity constraints (Samson)
  - Querying knowledge bases (Samson)
  - Publishing and exporting knowledge bases (Samson)
  - Developing applications (Samson)
  - Managing projects (Jennifer)

- Conclusion (Samson and/or Jennifer)
Validating integrity constraints

- **Facet Constraint Tab**
  - Protégé facets are constraints on values of slots (e.g. minimum cardinality)
  - FacetConstraint Tab brings all instances with facet constraint violations together in one place

- **PAL Constraint Tab**
  - Protégé Axiom Language (PAL) lets you write integrity constraints across multiple slots and multiple instances
  - PAL constraint tab allows checking of PAL constraints

- **EZPAL Tab**
  - Provides templates for easier authoring of PAL constraints

**Facet-constraint tab**

- Classes with instances that have facet-constraint violations
- Instances that have facet-constraint violations
- Button to start facet-constraint checking
PAL-constraint tab

- Allows specification and checking of complex integrity constraints

- Instances with PAL constraint violations

- PAL constraint being checked

- Example of PAL constraint

EZP AL tab

- Templates for fill-in-the-blanks method of defining PAL constraints

- Templates for classes of PAL constraints
Querying knowledge bases

- **QueryTab**: search instances of a class
- **StringSearch**: search for a string in entire project
- **PALQueryTab**: complex search condition
- **JessTab/JOT/Algernon**: programming tools with tab interface
- **SearchAPI**: Java API for constructing complex criteria for searching instances in entire project

Query tab: search for instances of a class whose slots satisfy some criteria
String search: search for a string anywhere in the project

**PAL Query**

- Allows search with complex search criteria
JessTab as a query interface

(deffunction findApplicableGuideline (?pid)
  (return
   (find-all-instances ((?g Guideline))
     (hasApplicableContext ?g ?pid))))

(deffunction hasApplicableContext (...)
  (...)

(getBrowserTexts (findApplicableGuideline "Assiss")
  (Mock guideline for managing cold"
  Jess")

Exporting and publishing and knowledge bases

- Alternative Protégé backends
  - XML Schema – saves a Protégé project using a fixed Protégé XML schema
  - XML ontology – saves a Protégé project by creating an XML schema based on the ontology in the project
  - RDF, OWL – save in formats used in semantic web formats
- Export functionalities
  - HTML export – Java-doc style html pages for classes and instances
  - XMLtab – XML documents for classes and/or instances
  - JessTab/JOT/Algernon – general purpose programming
  - TMRuleML, -export to specialized formats
- “Publishing”
  - ProtégéWebBrowser – browse Protégé projects on the web
  - Protégé CORBA server – access Protégé projects through CORBA interfaces
HTML export

- Creates “Javadoc-style” HTML pages for Protégé classes (& instances)

XML tab as exporter

- Export classes and instances in separate XML files
Publishing: Protégé web browser

- Requires installation of servlet-capable web server (e.g., Apache Tomcat)
- Present read-only Protégé KB on the web
- Possible to add annotations
- Possible to get screen shots of Protégé GUI associated with each instance
- Possible to download projects

Application development: different approaches (1)

- Use Protégé’s Java application programming interface (API) to access and manipulate knowledge base
  - **JSave** creates Java class stubs based on Protégé classes
  - Export Protégé knowledge to other environments
    - RDF XML files read as input to applications that use XML as inputs
    - **BeanGenerator, CLIPSTab**: export to agent/rule-programming environments
Application development: different approaches (2)

- Plug-ins: Use programming paradigms that have been made interoperable with Protégé
  - JessTab, Algernon: Rule-based programming
  - Prolog tab: Logic-based programming
  - JOT (Jython), JessTab, Algernon: Scripting environment
- In the future: Protégé come with pre-made problem-solving methods that can be configured for different applications
  - PSMTab: allows mapping of domain knowledge bases to “method ontology” of generic problem-solving methods

Application development: JessTab example

- Jess: Java Expert System Shell developed at Sandia National Laboratory, USA
  - Forward-chaining rule engine that match
  - Powerful scripting language
- JessTab: integrate Jess with Protégé
  - Mapping between Jess and Protégé
    - Protégé classes mapped to a Jess fact template
    - Protégé instances mapped to Jess facts and Jess facts mapped to instances
    - Changes to mapped facts in Jess reflected in Protégé; changes in Protégé reflected in Jess
  - Allows Jess to be run within Protégé GUI
  - Possible to embed both Jess and Protégé in Java program
JessTab: Rule-based programming in Protégé

- If a patient has problem, then conclude that he/she has more generalize problem

\[
\text{(defrule ObservationDeductions} \\
\quad \text{(object (is-a Observation) protégé class)} \\
\quad \text{(code ?code) protégé slots}} \\
\quad \text{(patient_id ?pid) creating protégé instance} \\
\quad \text{?super<-(object(is-a CodedConcept))} \\
\quad \text{(test (superclassp ?super ?code))} \\
\quad \text{=} \\
\quad \text{(make-instance of Observation creating protégé instance}} \\
\quad \text{(code ?super) creating protégé instance}} \\
\quad \text{(patient_id ?pid) creating protégé instance}} \\
\quad \text{(source guideline) creating protégé instance}} \\
\)}

Outline

- Introduction
  - Review of plug-in types (Jennifer)
  - Review of plug-in architecture (Jennifer)
  - Describe example application and ontology (Samson)
- Tasks
  - Conceptualization (Samson)
  - Reusing/importing existing resources (Jennifer)
  - Visualizing ontologies (Jennifer)
  - Customizing displays (Jennifer)
  - Dealing with non-standard data types (Jennifer)
  - Navigating and browsing (Jennifer)
  - Validating integrity constraints (Samson)
  - Querying knowledge bases (Samson)
  - Publishing and exporting knowledge bases (Samson)
  - Developing applications (Samson)
  - Managing projects (Jennifer)
- Conclusion (Samson and/or Jennifer)
Managing Projects

- Project Browser – manage libraries of Protégé projects

- Prompt
  - Move classes/instances up and down inclusion lattices
  - Extract portions of ontologies
  - Compare versions of the same ontology

Prompt Tab Demo

MANAGING MULTIPLE ONTOLOGIES

- **Compare** your current ontology to a different version of the same ontology.
- **Merge** two ontologies and add the resulting merged ontology to your current project.
- **Extract** a portion of another ontology and add it to your current project.

Choose the included project that will be modified:

file://Documents%2F...
Summary

User Interface
- Slot Plugin
- Tab Plugin
- Export/Import Plugin

Core Protégé
Protégé Default User Interface

Knowledge Model
Protégé Knowledge Model API
Protégé Knowledge Model

Persistent Storage
- CLIPS
- R2IF
- Other files
- Generic Schema
- Other schema
- Flat file storage
- RDBMS storage

Export/Import Plug-ins
Project Plug-ins