A User Oriented OWL Development Environment Designed to Implement Common Patterns & Minimise Common Errors

Matthew Horridge[1], Alan Rector[1], Nick Drummond[1], Holger Knublauch[2], Hai Wang[1]

[1]Department of Computer Science, University of Manchester, Manchester M13 9PL, UK
matthew.horridge@cs.man.ac.uk
rector@cs.man.ac.uk
[2]Stanford Medical Informatics, Stanford University, USA

Abstract: The Protégé-OWL environment including visualization, debugging, and wizard tools from the CO-ODE programme provides a user oriented environment for developing OWL ontologies. The long-term goal is to help subject matter experts to build and manage their own ontologies. The environment has been designed based on experience in teaching OWL and its precursor languages and on the experience of "ontology patterns" being collated by the Semantic Web Best Practice group. Protégé's plug-and-play architecture is designed to support multiple interfaces – some aimed at neophyte, others at logicians and teaching, others at large scale development – and a range of tools for visualization, debugging, and implementing common patterns. It also provides a smooth transition from frames to OWL. The demonstration will cover a variety of development scenarios. All tools are public domain and available from http://protégé.stanford.edu and http://www.co-ode.edu where a comprehensive tutorial based on the tools can also be found. The project is still evolving and interested users are invited to join the forum and contribute requirements, ideas, and/or additional plugin modules.

1. Introduction

The Protégé-OWL and CO-ODE projects assume that different user interfaces will be required for different users and different types of ontologies. It also assumes that users want to work at the level of patterns and principles rather than at the level of the detailed logical constructs embodied in the OWL language itself. A key source of design ideas has been experience of common errors garnered from experience in giving courses and tutorials in OWL[4]

The guiding principles are to:

- Make the easy thing the right thing – to make the defaults the usual case.
- Make it easy to do the right thing – to provide short cuts for common tasks.
- Ensure that complex tasks are fully and correctly completed – to provide wizards to guide users through complex tasks to the end.
- Make it easy to see what has been done – to provide visualization and debugging aids.

An important side effect of automating common tasks has been that it has reduced the time required to produce standard example ontologies by a factor of four or more.

2. The "Logician's Interface": combining definitions and axioms

The primary interface tested to date is the "Logician's" or "Students'" interface. It provides access to the full features of OWL, but in a more convenient format than previous editors such as OilEd [1]. A major goal has been to get as much information together on a single screen as possible. Hence, class definitions and axioms have been combined by giving each class a subpane for "necessary and sufficient conditions" (the definition) and "necessary conditions" simple subsumptions.

A number of simple common operations are integrated into the classes tab - a major pane of the interface, including: (See fig 1)

- Making all sibling classes disjoint .
- Converting classes from primitive to defined and vice versa (*i.e.* from "partial" to "complete" definitions).
- Moving individual restrictions from "necessary and sufficient" to "necessary" subpanes, either by drag-and-drop and cut-and-paste methods.
- "Cloning" classes to enable easy "copy and edit" style development of similar classes.
- Generating "closure axioms" for any property with a single operation – *i.e.* generating a universal restriction whose filler is the disjunction of all of the fillers of the existential restrictions for the same property.

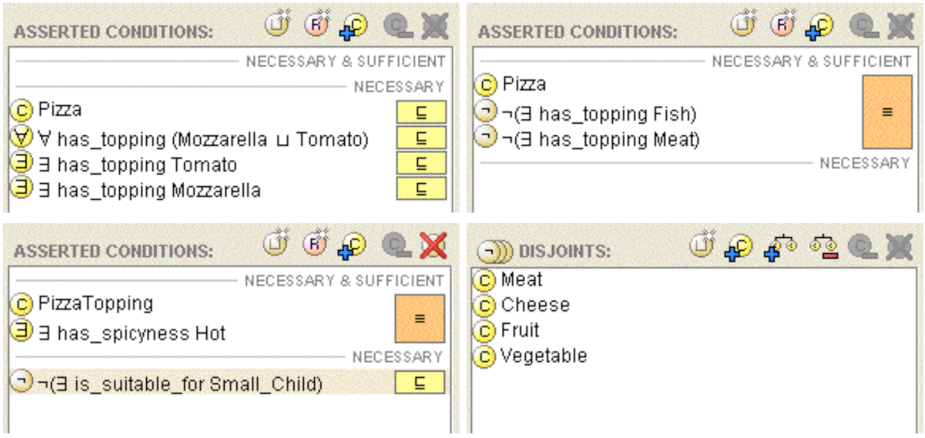

Figure 1: Some widgets from the classes tab: Conditions widget showing a primitive (partially defined) class "Margherita pizza" with a closure axiom, a completely defined class, and a completely class with an additional necessary restriction (axiom). Restrictions can be moved easily between subpanes. The Disjoints widget showing an entire set of disjoints, which are settable with a single click.

3. Wizards and automating process

Just as in programming, ontology development is often best thought of in terms of "design patterns". The CO-ODE extensions provide a plugable wizard architecture to implement common patterns. This work is closely following the work of the Semantic Web Best Practices and Deployment working group.

The Wizard architecture is itself plugable to encourage others to create wizards to automate common patterns. Important wizards to date include:

- Create a list of disjoint sibling classes – soon to be extended to creating disjoint trees.
- Create "value partitions" – *i.e.* a property, a class representing a "Quality" ("Value type") and a set of subclasses exhaustively partitioning the Quality to act as "values" [5].
- Creating enumerated classes (nominals).
- Using a form to enter the values for existential restrictions on a set of properties for a group of classes – the "Property matrix" wizard (See fig 2).
- Representing relations as values.

A range of further wizards are planned including tools for "normalisation"[3] ("untangling") and for transforming existing classes or generating new classes from existing classes according to an example pattern.

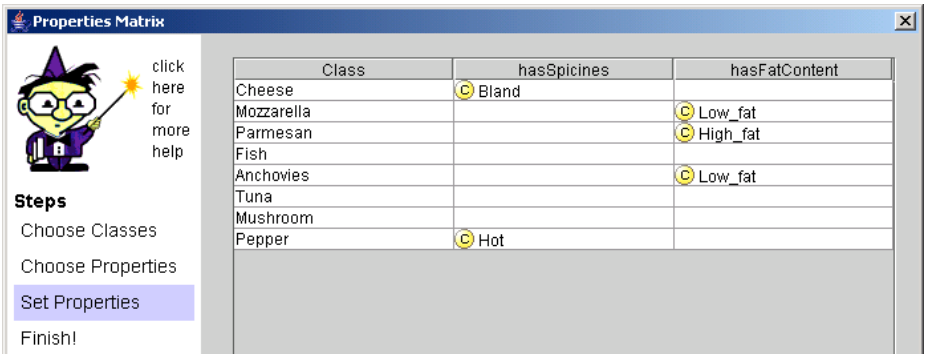

Figure 2: Fragment of "Property Matrix Wizard" allowing existential restrictions for several properties to be added quickly to a set of classes.

4. Visualisation Explanation and Debugging

Understanding the detailed meaning of OWL and locating classification errors in even moderately large ontologies is notoriously difficult. Since any subclass of an unsatisfiable class is unsatisfiable, and any existential restriction filled by an unsatisfiable class is unsatisfiable, errors propagate.

The CO-ODE extensions to Protégé OWL provide two types of help:

- Simultaneous display of asserted and inferred hierarchies and list of changes made by the reasoner to the classification.
- A comprehensive graphical visualization tool.
- Paraphrase pop ups to make class descriptions clearer.
- Debugging aids that highlight the cause of a class being unsatisfiable.
- Comparison and ontology evolution aids – the PROMPT tool [2]
- A scripting interface to Python to facilitate one off tasks and ontology migration.

Figure 3 OWLViz visualization tool

5. Architecture, Requirements & availability

The projects are collaborating to use Protégé's plug-and-play architecture to make it easy to create multiple interfaces, wizards, debugging aids, explanation aids based. It provides dynamic mapping between Protégé's frame and slot model based on OKBS and OWL's model of class, restrictions and axioms.

All software is written in Java and available for Windows, Mac and Linux.

All software is open source and can be obtained, along with further details, tutorials, and examples from http://protégé.stanford.edu and http://www.co-ode.org.

Contacts: Alan Rector or Matthew Horridge: {rector | horridge}@cs.man.ac.uk

References

1. Bechhofer, S., Horrocks, I., Goble, C. and Stevens, R., OilEd: a Reason-able Ontology Editor for the Semantic Web. in *KI2001, Joint German/Austrian conference on Artificial Intelligence*, (Vienna, 2001), Springer-Verlag, 396--408.
2. Noy, N.F. and Musen, M.A., PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. in *Seventeenth National Conference Artificial Intelligence (AAAI)*, (Austin Texas, 2000).
3. Rector, A., Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. in *Knowledge Capture 2003*, (Sanibel Island, FL, 2003), ACM, 121-128.
4. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H. and Wroe, C., OWL Pizzas: Common errors & common patterns from practical experience of teaching OWL-DL. in *European Knowledge Acquisition Workshop (EKAW-2004)*, (2004), (in press).
5. Rector, A (ed). Representing Specified Values in OWL: "value partitions" and "value sets", W3C Editor's Draft, 2004,http://www.w3.org/TR/swbp-specified-values/