OntoTrack: Fast Browsing and Easy Editing of Large Ontologies

Thorsten Liebig[1] and Olaf Noppens[1]

Dept. of Artificial Intelligence
University of Ulm
D-89069 Ulm
{liebig|olaf.noppens}@informatik.uni-ulm.de

Abstract. OntoTrack is a new browsing and editing "in-one-view" ontology authoring tool. It combines a sophisticated graphical layout with mouse enabled editing features optimized for efficient navigation and manipulation of large ontologies. The system is based on SpaceTree [PGB02] and implemented in Java2D. OntoTrack provides animated expansion and de-expansion of class descendants, zooming, paning and uses elaborated layout techniques like click-able miniature branches or selective detail views. At the same time OntoTrack allows for quite a number of editing features using mouse-over anchor buttons and graphical selections without switching into a special editing layout. In addition, every single editing step is synchronized with an external reasoner in order to provide instant feedback about relevant modeling consequences.

1 Introduction

The availability of adequate tools for end users is a pivotal element in order to push Semantic Web techniques from academia to commercial environments. Simple, flexible, and intuitive user interfaces play an important role within this context. In contrast to current tool evaluations which concentrate mainly on language specific issues (e. g. language conformity) and technical criteria (e. g. turn around ability for interoperability) [AS02] we will focus on adequate visualization, navigation and simple editing of large ontologies in the remainder of this paper.

Currently, many ontology editors use two functionally disjunct interfaces for either editing or browsing ontologies. Editing interfaces are commonly based on vertical expand and contract lists representing the class hierarchy. When selecting a particular class in the list one can inspect and manipulate its corresponding definition using predefined forms in an additional display area. Our experiences with expand and contract style interfaces identified a number of conceptual drawbacks:

– The number of visible classes is limited by the screen height. Even middle sized ontologies very likely require scrolling after some level expansions.

– The larger the ontology the harder it will get to identify the inheritance path from a particular class up to the root of the ontology. This is due to the fact of exclusively two level states. An ontology level is either completely expanded or contracted and is not allowed to display a selection of context relevant classes.
– Depending on the branching factor of an ontology a list representation makes it difficult to compare two different expansion paths concerning level depth or common ancestors.
– Because of the tree based nature of expansion lists multiple inheritance is difficult to represent in general. Multiple ancestors of a class are usually displayed with help of an auxiliary display area. Inversely, this class will appear as "cloned" class in the list of descendants of every super class. This, however, will result in a proportional growth of redundant classes with the number of multiple inheritance statements.
– When defining a class one commonly needs to access and select other classes. This temporally requires additional expand and contract style selection lists for a class hierarchy already on screen.

An extreme example for which the list representation will be inherently unsuitable is the task of showing the complete inheritance path of a class in a large ontology having multiple ancestors.

In order to better support those tasks most tools incorporate an additional graphical browsing interface using tree like, tree map, ven or hyperbolic layout techniques more or less suitable for navigating large ontologies. However, those interfaces do not allow for substantial editing and are designed as view-only plugins in most cases.

Our novel ontology editor, called *OntoTrack*, combines hierarchical layout technologies with context sensitive zooming features and mouse enabled editing abilities optimized for navigation and manipulation of large ontologies. OntoTrack is based on the linked tree diagram approach of SpaceTree [PGB02] which dynamically zooms and lays out tree branches to best fit the available screen space. OntoTrack's ontology layout is driven by an animated "expansion on user demand" strategy making use of elaborated minimization techniques for alternative inheritance paths or descendants. At the same time OntoTrack allows for quite a number of editing features from mouse-over anchor buttons to context sensitive choose lists without switching into a special editing layout.

The remainder of this paper is organized as follows. In the next section we present OntoTrack our new graphical authoring tool for ontologies. In particular, we explain OntoTrack's browsing, editing, and searching abilities as well as its inference features via link-up to an external reasoner. In section 3 we describe the current implementation status and discuss current and future work. Section 4 contains preliminary benchmarking results concerning to some qualitative navigation criteria. We will end with a short summary and some notes about possible enhancements.

2 A New Graphical Authoring Tool For Ontologies

2.1 Browsing Features

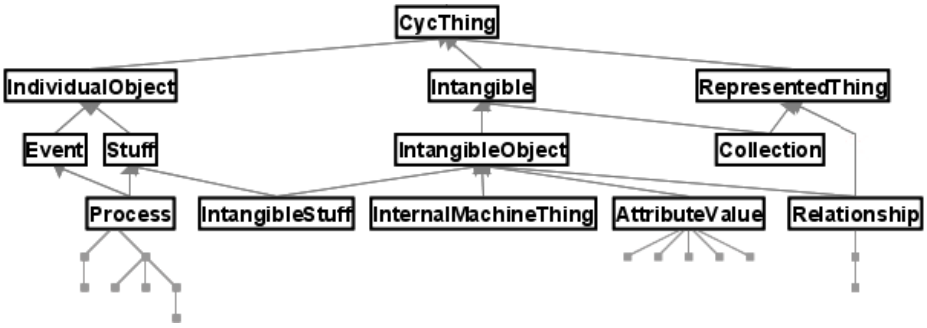

Fig. 1. Partially expanded ontology in top-down layout showing miniature tree thumbnails summarizing not expanded sub-branches.

OntoTrack aims at integrating optimized layout techniques for hierarchies with graphical editing features. Currently, the system is based on SpaceTree [PGB02] an interactive tree browser with dynamic rescaling of branches, optimized camera movement, and preview icons for non expanded sub-branches. Within SpaceTree the expansion of a new tree level is animated and may result in trimming of branches of previous levels when needed. SpaceTree allows for changing the overall orientation of the layout and for explicit de-expansion blocking of user selected branches.

Our attempt was to adapt SpaceTree for browsing and editing ontologies by extending its inheritance centered layout algorithm in a first step. The primary structuring element of ontologies and trees is the inheritance relation. Consequently, SpaceTree's as well as OntoTrack's layout algorithm dynamically adapt their graphs in order to be able to display the complete inheritance path either in a top-down or left-right orientation. As an option, the path from the last expanded class to the root will be outlined. Depth, width and the number of descendants of not expanded sub-branches are symbolized by triangles of varying length, width, and shading or as an iconified branch in order to provide information about deeper levels. In addition, the whole ontology layout can continously be zoomed or paned simply by mouse-down movements. Figure 1 shows an OntoTrack screen capture of an ontology[1] after some level expansions. Here, a classical top-down orientation and a miniature tree thumbnail style has been chosen. In comparison, figure 2 shows the same ontology in a less expanded state using a left-to-right orientation and a triangle thumbnail style. Here, the

[1] Showing the top-level classes in an early version of Cyc.

inheritance path for the last expanded class "IndividualObject" is outlined by a darker node background.

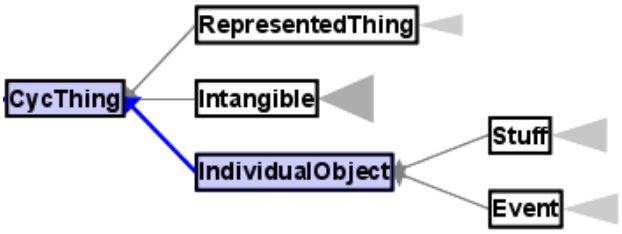

Fig. 2. Ontology of figure 1 in left-right layout and triangle thumbnails.

In the case of expanding a level containing classes having multiple ancestors in currently not expanded branches, those ancestors are drawn as click-able icons. As an example, figure 3 shows the ontology of figure 2 after expansion of class Stuff via middle mouse click. Both descendants of Stuff have multiple ancestors. A further ancestor of Process is an already expanded class Event. In contrast, one ancestor of IntangibleStuff (namely IntangibleObject as can be seen in figure 1) is within the currently not expanded sub-branch of class Intangible and therefore drawn as a click-able icon.

When moving over an iconified ancestor with the mouse pointer a tool-tip message with the corresponding class name appears. Clicking on such an ancestor icon results in an expansion of this class. This strategy guarantees that all ancestors of all expanded classes are displayed either expanded or abbreviated as click-able icons. Having all inheritance paths visible up to the root helps a user to keep orientated concerning to the primal structuring principle of ontologies.

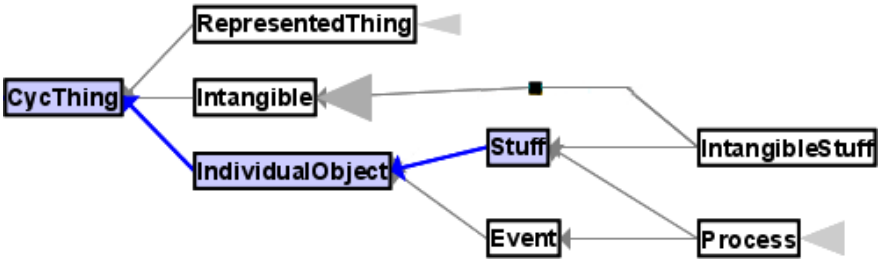

Fig. 3. Ontology of figure 2 after expansion of class Stuff.

2.2 Editing Features

As mentioned before OntoTrack is a browsing and editing in-one-view authoring tool. This allows to re-use already available navigation principles for the task of building and manipulating ontology definitions. The most primitive manipulation feature consists of the direct editable class name field of every class node. Beyond that, OntoTrack's click-and-drop editing features are enabled by switching into the "anchor button" mode. Within this mode anchor buttons appear when moving the mouse over editable entities. Figure 4 shows the anchor buttons of a class IndividualObject displayed in top-down orientation (in left-right orientation the button layout is rotated 90° anti-clockwise). The triangle symbol on top of the class box represents the superclass relationship. With a click on this button one can specify this class to be an descendant of another class selectable with a click on that class. A new sub class can be created with a click onto the bottom triangle. In correspondence with the RDFS and OWL specification the semantics of multiple subclass statements for one class is that of a conjunction in OntoTrack.

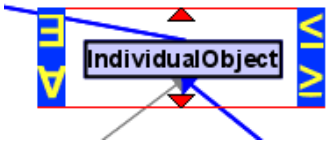

Fig. 4. Anchor buttons of a class in top-down layout.

In addition, OntoTrack offers further editing functions while in its "detailed view" mode. The detailed view mode is activated or deactivated for each class separately using the mouse-wheel up- resp. down-wards while being over the class with the mouse pointer. When activated, OntoTrack uses a slightly adapted UML style class diagram syntax. In contrast to the UML specification our class diagram is divided into two (instead of three) compartments. The top compartment contains the name of the class. The bottom compartment contains a list of property restrictions of this class. In case of OWL Lite each row of this list contains (implicit conjuncted) one existential or universal quantification or unqualified cardinality restrictions displayed in abstract Description Logic (DL) syntax (see [Baa03] for the abstract DL terminology). Figure 5 shows a class with one minimal and one exact cardinality restriction. An existing restriction can be deleted by clicking on the red dot on the right side of the corresponding row. A new restriction is added to a class by using the green dot at the bottom of the class box (see figure 5). The cells of each row are editable via choose lists. An unqualified cardinality restriction provides three choose lists, one for the cardinality operator ($\geq, \leq, =$) one for the value (0 or 1 in OWL Lite) and one for the currently available properties. Quantifications also require three choose lists

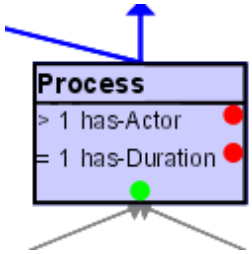

Fig. 5. Class in detailed view mode.

(one for the quantifier ∃ or ∀, one for the property, and one for the qualifying class). Additional editing features like switching between complete and partial definitions are accessible via a right mouse button context menu. As an alternative short-cut we plan to add click-and-drop quantifier and cardinality symbols for specifying properties statements between classes as shown in figure 4 in the near future.

2.3 Inference Feedback

OntoTrack is equipped with an interface to an DL reasoner called RACER [HM01]. All changes after each editing step (e. g. list selection, subclass statement) are send to the RACER system via the TCP-based client interface JRacer. RACER will then make all modeling consequences explicitly available for OntoTrack. Of special interest within our ontology layout is the subsumption relationship which may implicitly be influenced by an editing step. As soon as RACER recognizes a change in the class hierarchy OntoTrack updates the corresponding graphical representation (showing only direct subsumers/subsumees of each class). Those updates are also animated in order not to confuse the user with a new hierarchy layout in one step. Other graphical inference services (which are special cases of the subsumption relationship in fact) cover unsatisfiable class definitions or equivalence between different classes. In OntoTrack an unsatisfiable class will be drawn in red and equivalent classes are outlined with a colored background.

2.4 Searching

OntoTrack also adapts SpaceTree's search features. When looking for a specific class name, even in the selection phase during editing, one can use a string based ontology search. When start typing a search string all matching classes or sub-branch icons are highlighted. Each additional character or deletion in the search string directly results in an updated highlighting of matching parts of the ontology. OntoTrack currently supports three matching mode: exact match, substring match from string beginning, and full substring match. As an option,

the user can then fan out the ontology by expansion of all currently matching classes via one button click.

3 Implementation Status and Current Work

Our ontology authoring tool OntoTrack is still under development. The features described in section 2 are those of the first implementation phase. Some may change in future versions if they don't prove to be useful. It is our considered opinion that performance and scalability are very important properties of user friendly tools and a key for user acceptance. We therefore have chosen Piccolo as our graphical library. Piccolo is an optimized subset of Jazz [BMG00], a fast zoomable interface toolkit based on Java2D.

A first prototype of OntoTrack has been implemented by extending SpaceTree's layout algorithm, which itself uses the Piccolo libraries. Within this version all mentioned browsing features of subsection 2.1 are implemented in full detail. Some of the editing features of subsection 2.2 however are still under development (click-and-drop qualifiers and cardinality statements).

Current work is focused on refining and optimizing the layout algorithms for ancestor thumbnails. Miniature tree layouts for ancestors with multiple inheritance turned out to be difficult in general. Imagine the problem of thumb placing for a short expanded inheritance path together with a long alternative path via a thumbnail miniature tree (or vice versa). Inheritance links between thumbnail classes and already expanded classes are another factor of complexity for placing and cross minimizing layout algorithms. As an additional constraint we want to re-arrange the layout of expanded classes in each possible expansion step as less as possible. Therefore, OntoTrack implements a local optimization layout algorithm triggered by the class the user currently wants to expand.

OntoTrack's file import as well as export uses the RDF parser Jena2[McB01]. Jena2's internal ontology model for classes and properties also serves as OntoTrack's central representation model. Currently, OntoTrack is able to read in and write out OWL Lite ontologies. However, properties as well as global property constraints (domain and range statements) are not editable in OntoTrack at the moment.

Conceptually, we plan to cover a notable fragment of OWL Lite's language constructs while adopting UML's class diagram representation. In a first step we concentrated on OWL Lite's class axioms and restriction statements (see section 2.3.1.1. and 2.3.1.2. of OWL Abstract Syntax and Semantics document [PSHH03]). Next, we want to extend the editor with a parallel representation of properties and property hierarchies. Our goal is an mixed graphical representation based on the hierarchical class layout described above together with editable property edges in combination or as alternative to the list representation of OntoTrack's detailed view mode.[2]

[2] The ezOWL plugin [OC03] for Protégé is an example of a likewise mixed class and property representation to some extend.

4 Preliminary Evaluation

It was not the goal of our preliminary evaluation tests to determine an overall ranking of different ontology editors. Other tools like Protégé [GMF+03], OntoEdit [SSA01] or OilEd [BHGS01] are obviously in a more sophisticated state of development and in some cases tailored to different tasks or users. Our aim was to evaluate our graphical browsing and editing interface against other user interfaces with respect to certain navigation criteria.

First we compared the maximum number of classes to display for a given screen size. Using a screen size of 1280×1024 we counted a number of 50 to 60 displayable classes in expand and contract style ontology browsers using full screen hight (here, the screen width has no effect on the maximum of displayable classes). Using a comparable font size in OntoTrack we were able to expand more than 100 classes using full screen mode.

In contrast, the length for an inheritance path for a branch with classes having a name with an average length of 12 characters has a depth of 13 levels in OntoTrack. In an expand and contract style interface the same number of level expansions approximately take up 30 % of the screen width.

However, in order to have some qualitative results concerning average navigation or editing performance a controlled experiment has to be conducted. A set of experiments comparing three tree-based browsing tools (MS Explorer, a Hyperbolic tree browser, and SpaceTree) showed some performance advantages for the SpaceTree approach concerning tasks like first-time node finding, listing all ancestors of a node, or differentiate between branches with varying numbers of nodes [PGB02]. These results may serve as an indicator with respect to navigation and editing performance of OntoTrack in comparison with expand and contract style interfaces.

5 Summary and Outlook

Expand and contract style interfaces for ontologies inherently have substantial drawbacks concerning search and navigation speed, user orientation, and editing flexibility in our opinion. Our new authoring tool for ontologies combines an animated graphical layout with mouse enabled editing features within one view. We are still in an early development phase, but first experiences with our SpaceTree [PGB02] based prototype are encouraging. We therefore see OntoTrack as an easy-to-use interactive ontology editor especially for non-experienced users and even for large ontologies.

Current work focuses on finalizing the layout algorithm, and further editing features. We also plan to extend OntoTrack's search facility for regular expression matching as well as for restriction expressions. The link-up to the RACER reasoner is also a subject of optimization. Currently, OntoTrack needs to query the reasoner for all possible consequences of each user change in order to update its internal representation model. Here, an event triggered notification model on reasoner side would significantly speed up this process. In addition, an adequate explanation module is needed in order to distinguish between 'direct'

consequences (e. g. an unsatisfiable class because of an user manipulation) and follow-up consequences (e. g. the consequences of an unsatisfiable class with respect to other classes). In order to become a competitive application basic features like undo, print, or various exports into other ontology languages have to be implemented in future versions of OntoTrack.

We plan to cover ontology languages with an expressivity at least comparable to that of OWL Lite. Complex class descriptions like nested restrictions or general inclusion axioms may need additional graphical features in a next evolution step. A graphical representation as well as editing interfaces for disjoint classes, coverings and instances are also on our working agenda. An graphical UML representation for some of those have already been discussed in [BKK+01] and may serve as starting point for our application.

References

[AS02] Jürgen Angele and York Sure. Whitepaper: Evaluation of Ontology-based Tools. Technical report, OntoWeb Deliverable 1.3, 2002.

[Baa03] Franz Baader. *The Description Logic Handbook*, chapter Appendix 1: Description Logic Terminology. Cambridge University Press, 2003.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *Proc. of the German conference on Artificial Intelligence, KI2001*, pages 396 – 408. Springer Verlag, LNAI Vol. 2174, September 2001.

[BKK+01] Kenneth Baclawski, Mieczyslaw K. Kokar, Paul A. Kogut, Lewis Hart, Jeffrey Smith, William S. Holmes, Jerzy Letkowski, and Michael L Aronson. Extending ULM to Support Ontology Engineering for the Semantic Web. In *Proceedings of the Fourth International Conference on UML (UML 2001)*, number 2185 in LNCS, pages 342 – 360, Toronto, Canada, October 2001. Springer Verlag.

[BMG00] Ben Bederson, Jon Meyer, and Lance Good. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2(2):171 – 180, 2000.

[GMF+03] John Gennari, Mark Musen, Ray Fergerson, William Grosso, Monica Crubézy, Henrik Eriksson, Natalya Fridman Noy, and Samson Tu. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. *International Journal of Human Computer Studies*, 58(6):737 – 758, June 2003.

[HM01] Volker Haarslev and Ralf Möller. RACER System Description. In *Proc. of the International Joint Conference on Automated Reasoning, IJCAR'2001*, Siena, Italy, June 2001.

[McB01] Brian McBride. Jena: Implementing the RDF Model and Syntax Specification. In *Proc. of the Second International Workshop on the Semantic Web, SemWeb'2001*, Hong Kong, China, 2001.

[OC03] Sooyoung Oh and Moonyoung Chung. ezOWL plugin for Protégé. `http://iweb.etri.re.kr/ezowl/`, 2003.

[PGB02] Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proc. of the IEEE Symposium on Information Visualization, INFOVIS 2002*, pages 57 – 64, Boston, USA, October 2002.

[PSHH03] Peter Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Working Draft, March 2003.

[SSA01] York Sure, Steffen Stab, and Jürgen Angele. OntoEdit: Guiding Ontology Development by Methodology and Inferencing. In *Proc. of the Confederated International Conferences CoopIS, DOA and ODBASE 2002*, pages 1205 – 1222. Springer Verlag, LNCS Vol. 2519, October 2001.