The table metaphor: A representation of a class and its instances

Jan Henke

Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
jan.henke@deri.org

Abstract

This paper describes an approach on how to visualize instances and the relation to their classes. The approach is motivated and described in the context of ontologies and the Semantic Web but is general enough to be utilized for any object model visualization.

1 Introduction

For the Semantic Web ontologies are the basic building block. On the one hand they allow for a common vocabulary and thus for communication and interoperation. On the other hand they can be used for logical reasoning and therefore statements can be verified / falsified and knowledge can be inferred.

For ontologies in turn, classes and instances are the building blocks. As in object orientation they are used to model abstract definitions on the one hand and concrete examples of these on the other hand. How the visual editing of ontologies can be gained by a new representation of the class instance relation – called table metaphor – will be described in the following.

In section 2 the requirements of a class - instance visualization are listed before section 3 checks which of these are fulfilled by current approaches. Section 4 will address the table metaphor and section 5 concludes this paper.

2 Requirements

A good visualization should provide a correct transformation as well as a high usability. Leaving out the first would make it useless; leaving out the second would make it unused. Based on this belief some specific requirements shall be described below.

2.1 Correctness

Based on a classification by Shneiderman [5] visualization can be divided into the seven subtasks “Overview”, “Zooming”, “Filtering”, “Details on demand”, “Relations”, “History” and “Extraction”. The requirement of correctness to be described below is derived from the “Relations” task which is about emphasizing which item belongs to which other one.

Class membership
An instance cannot be used if its class membership - and thus its definition - isn’t clear. The other way around, a class without instances isn’t very useful (except for the case of abstract classes) because the definition has never been applied. Therefore it is crucial that a visualization unambiguously reveals this relation in both directions.

Attribute value mapping
The relation between attributes and their values can be compared to the one between classes and instances: While an attribute defines a range (a class) this requirement is fulfilled by the respective value (an instance). For this reason also the visualization of the attribute-value-relation has to be bidirectional and unambiguous.

Level matching
An instance is a concretization of a class. It reduces the abstractness by filling slots with values. Nevertheless an instance is neither a sub object of a class nor the other way around. Despite the different levels of abstraction they belong onto the same level of definition, i.e. an instance is as special as its class – not more special as a subclass (see Figure 1).

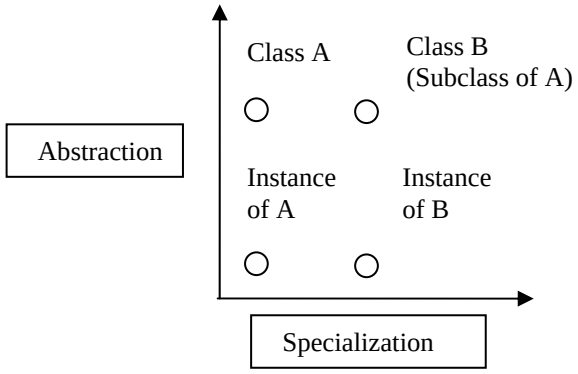

Figure 1 Concreteness vs. Specialization

Because of this, a visualization has to make sure that a class and its instances are positioned on one and the same level.

2.2 Usability

Usability can be described using the five criterions learnability, efficiency, memorability, errors and subjective pleasure [2]. In the following these requirements shall be applied to the special case of visualization.

Learnability
Learnability stands for a minimization of the learning phase duration. This is also required for a visualization so that it can be utilized as fast as possible.

Efficiency
Efficiency describes how much work can be done in a certain amount of time. An increased efficiency is a clear indicator for an improved visualization.

Memorability
After a longer break between two usage sessions it should not be necessary to restart the learning phase. Therefore a visualization should be simple enough to be memorized.

Errors
Errors should appear as seldom as possible and if they appear they should be fixable as easily as possible. For the visualization case this can be translated to the requirement of high clarity.

Subjective pleasure
The less formal criterion of subjective pleasure should not be underestimated. The user should have a good “feeling” using the visualization in order to improve his / her working results.

3 Current approaches

In order to show the lack in current instance visualizations and thus to motivate the table metaphor two current approaches will be described in the following.

3.1 Class tree duplication

One approach in current ontology editors – like for instance Protégé [4] – is to display instances connected to a copy of the class tree.

Protégé provides different tabs for classes and instances. If the instance tab is selected the class tree is displayed again – and once a class is selected, its instances will be displayed. Thus the class membership can easily be recognized.

Also the relation between attributes and their values is revealed unambiguously which fulfills the second criterion of the correctness requirement.

The requirement of level matching cannot really be decided. As mentioned above when a class is selected its instances are displayed. But it is not really clear whether they are still on the same level as the class.

Beyond this, the usability requirement of subjective pleasure can hardly be fulfilled by an approach that repeats the class tree in two different views, as it is done in Protégé.

3.2 Purely textual visualization

Another approach – as can be found in Oiled [1] e.g. – is to offer no graphical representation of the class instance relation but to display it purely textually.

In Oiled there can be found both a tab for classes and one for instances – comparably with Protégé. The difference consists in the fact that there's no class tree available in the instance tab. Instead of this the class membership can only be found in a respective combo box – thus purely textually.

The visualization of attributes and their values utilizes a table and is nicely usable.

The not very smooth class tree repetition of Protégé is not used – but unfortunately, it has not been replaced by any other feature. Because of the missing graphical connection between classes and their instances the level matching criterion cannot be applied.

4 The table metaphor

As shown above current visualization approaches hardly fulfill the mentioned requirements. Therefore a new idea – called table metaphor – shall be introduced.

The table metaphor represents a class and its instances by a table. More precisely this means a class is represented by a table header while each table row stands for an instance (see Table 1 – three instances of the class "City" with the attributes "Name" and "Inhabitants" are displayed). Thus an ontology – consisting of a schema and a knowledge base – can be seen as a tree of tables.

Table 1 Instance table example

Name	**Inhabitants**
Berlin	3 420 000
Hamburg	1 640 000
Munich	1 220 000

4.1 Correctness

Using the criterions described in the requirements section the correctness of the table metaphor will be shown below.

Class membership
The table metaphor allows for an easy recognition of class membership. Whenever a certain class is selected the respective table can be displayed.

Attribute value mapping
Using the table metaphor a certain attribute value is always displayed underneath the respective header cell. Thus it is always clear which attribute a value belongs to and which values have been assigned to a certain attribute.

Level matching
The header and the body of a table can be clearly distinguished thus the class and the instance part are explicitly separated. On the other hand a table body is not a sub object of a table header which fulfills the level matching criterion.

4.2 Usability

In the following it will be shown that a high usability can be expected of object model editors that apply the table metaphor.

Learnability
The presented approach is very simple. Tables are applied manifold in everyday life. Thus the leaning phase can be expected to be minimal.

Efficiency
Because of the dissemination of tables they can be read and understood quickly. Beyond this available widgets that allow for column wise sorting e.g. should also improve efficiency by far.

Memorability
Because hardly anything has to be learned in order to understand the table metaphor the problem of insufficient Memorability cannot appear.

Errors
Also the error minimization is guaranteed by the simplicity of the approach. If error correction strategies should be necessary nevertheless this has to be solved at implementation level.

Subjective pleasure
This criterion can hardly be predicted. But if the approach is well understood and thus increases efficiency also the subjective pleasure should be influenced in a positive way.

5 Conclusions

In this paper a visualization approach – called table metaphor – was introduced. A real world implementation of it can be inspected in the Distributed Ontology Management Environment (DOME) [2] – to be found at http://www.omwg.org.

References

1. Bechhofer, Sean; Horrocks, Ian; Goble, Carole; Stevens, Robert. OilEd: a Reason-able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001.
2. Henke, Jan. Architecture Design of an Editing & Browsing tool, 2004
3. Nielsen, Jakob. Usability Engineering. Morgan Kaufmann - An imprint of Academic Press, 1993
4. Noy, N. F.; Sintek, M.; Decker, S.; Crubezy, M.; Fergerson, R. W.; & Musen, M. A.. Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2):60-71, 2001.
5. Shneiderman, Ben. Designing the User Interface – Strategies for effective Human-Computer Interaction. Addison-Wesley-Longman, 1998