Supporting User Tasks through Visualisation of Light-weight Ontologies

Christiaan Fluit[1], Marta Sabou[2], and Frank van Harmelen[2]

[1] Aidministrator Nederland BV, Amersfoort, The Netherlands
email: Christiaan.Fluit@aidministrator.nl
[2] Vrije Universiteit Amsterdam, The Netherlands
email: Marta.Sabou@cs.vu.nl,
email: Frank.van.Harmelen@cs.vu.nl

1 Introduction

As is abundantly clear from the other chapters in this volume, ontologies will play a central role in the development and deployment of the Semantic Web. They will be used for many different purposes, ranging across information localisation, integration, querying, presentation and navigation.

Experiences in other fields (Data Mining, Scientific Computing) have shown that visualisation techniques can be successfully employed to support many of these tasks in those areas. The question then naturally arises if visualisation techniques can also be successfully employed on the ontology-based Semantic Web.

The answer to this question of course depends strongly on the nature of the ontologies that we expect to be deployed on the Semantic Web. In our opinion, two specific features of ontologies will be important with respect to visualisation:

We expect the majority of the ontologies on the Semantic Web to be *light-weight*. Light-weight ontologies are typified by the fact that they are predominantly a taxonomy, with very few cross-taxonomical links (also known as "properties"), and with very few logical relations between the classes. Our experiences to date in a variety of Semantic Web applications (knowledge management, document retrieval, communities of practice, data integration) all point to light-weight ontologies as the most commonly occurring type [1].

We also expect that in ontologies on the Web the number of instances will typically be very large compared to the number of classes. The number of classes may be up to thousands (or even tens of thousands), while numbers of instances will easily reach millions in many applications. Two characteristics are very common for such instantiated taxonomies: incompleteness and overlap. First, the set of subclasses of a class is *incomplete* when their union

does not contain all the objects of the superclass. Second, classes that share instances are *overlapping* if no specialization relationship holds between them.

Some systems already exist that exploit visualisation techniques for the Semantic Web. In this chapter, we will first survey a number of existing visualisations in the domain (section 2). This will lead us to identify a number of weaknesses in these existing visualisations, which can be explained by the (sometimes implicit) focus of these tools on one part of the ontology life-cycle and the tasks that take place in that part of the life-cycle. Also, these existing visualisations fall short on the scalability that we expect is required for ontologies with very large numbers of instances.

We then present the Cluster Map visualisation technique (section 3), and discuss a number of user tasks that it supports (section 5). This visualisation was developed at Aidministrator[3], a Dutch software provider, and has been used in several commercial and research projects. Our explanations of the visualisations are all backed up with illustrations from two of these projects, briefly explained in section 4.

2 Related Work

In this section we will review a set of relevant visualisation tools and techniques, judging them on two main criteria. For a more complete overview of the field, the interested reader is referred to [2].

The *first criterion* considers three different stages in the life cycle of an ontology. These stages require visualisations with quite different capabilities.

• Ontology development – The development stage of an ontology consists of creating its schema. At this stage a detailed visualisation of the concepts and their relationships is needed in order to enable a full understanding of the details of the ontology. The visualisation typically involves a small number of concepts and relationships. As the ontology grows, understanding and maintaining it becomes more difficult. The task of understanding it requires more coarse grained overviews with zooming facilities and the possibility to visualise different aspects of the ontology.

• Ontology instantiation – The instantiation of the ontology follows after the creation of the schema. Sometimes the process of populating ontologies is semi-automatic, performed by classifiers. The main concern of this stage is to ensure a high quality instantiation. Visualisation tools that differentiate between a schema and its instantiation are especially useful for this stage.

• Ontology deployment – Tools and methods are needed for analysing, querying and navigating ontological information spaces. These tools can target the needs of end users of the information, or the needs of developers of information systems involving the ontological information.

The *second criterion* for judging visualisations deals with the degree to which the ontological nature of the visualised data is exploited. The surveyed

[3] http://www.aidministrator.nl

techniques range from generic data visualisers (e.g. for trees and graphs), to visualisations that strongly rely on the ontological nature of the data (i.e., that use the semantics of relations such as subclass-of, instance-of and part-of).

Two well-known systems for visualising large data structures are the Hyperbolic Tree [7] for the navigation of trees (see fig. 1), and The Brain[4] for navigating graphs. Applications of these visualisations typically display syntactic structures such as link structures. This of course has some semantics, but this is often very implicit and ad hoc. Additionally, the data structure used by these tools is often a tree or at least tree-like (it is arguable whether The Brain is really suited for navigating arbitrary graphs).

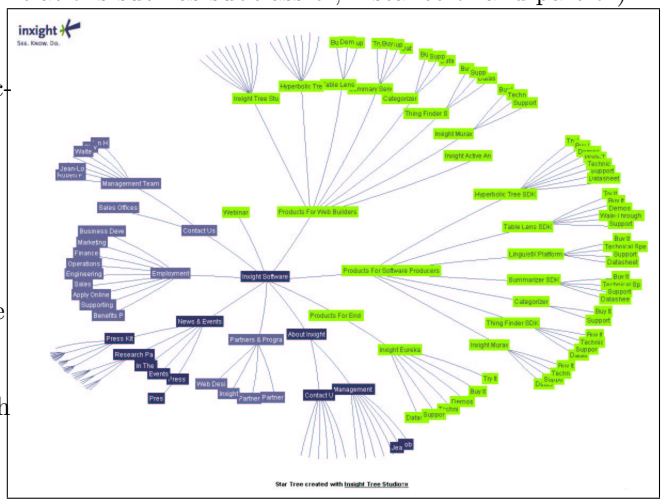

Fig. 1: A Hyperbolic Tree.

Fig. 2. Web Brain: The Brain applied to the Open Directory Project.

Even though developed for generic data, one can visualise semantic data with these techniques. WebBrain[5] uses the taxonomy of the Open Directory

[4] http://www.thebrain.com
[5] http://www.webbrain.com

project (ODP[6]) as input for The Brain (see fig. 2). The AquaBrowser [14] visualises concept graphs. These systems primarily display the abstract structure. The instances are not an inherent part of the visualisation but are displayed separately (usually as a textual list).

There are visualisations that show both abstract and instance data at the same time, such as Antarcti.ca's Visual Net[7], which has also been applied to the ODP[8]. The resulting visualisation, MapNet, is shown in figure 3. This visualisation can also scale to large taxonomies and document sets, but has no explicit way to express overlaps (shared instances) between classes. Such overlaps typically occur frequently in classification systems such as Yahoo or the ODP and may reveal additional interesting information about the classes and their instances. Unfortunately, instances that belong to several classes are simply duplicated in the visualisation, providing no indication of their multiple class memberships to the viewer.

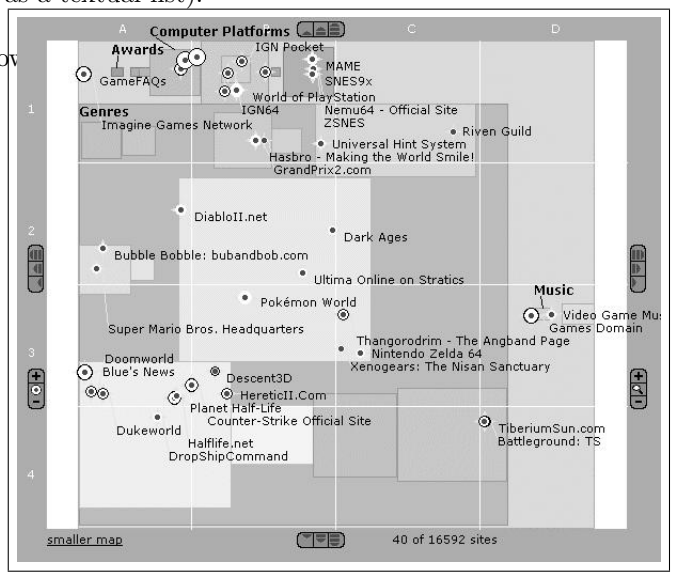

Fig. 3: MapNet.

Many ontology authoring tools [9] offer visualisation facilities that support the needs of the ontology development stage. We describe some of them.

A generic graph visualisation toolkit, which has been successfully applied to the visualisation of RDF data models, is GraphViz [5], developed at AT&T. Tools based on GraphViz, such as IsaViz[9] or RDFViz[10], are mostly employed to understand RDF data. Both schema elements and instances are shown. The visualisation allows the representation of any kind of relationship between classes and instances, being well suited for displaying complex ontologies. The scalability of the visualisation is quite poor: it is hard to understand the visualisation as soon as more relations are added.

[6] http://www.dmoz.org
[7] http://www.antarcti.ca
[8] http://maps.map.net
[9] http://www.w3.org/2001/11/IsaViz/
[10] http://www.ilrt.bris.ac.uk/discovery/rdf-dev/rudolf/rdfviz/

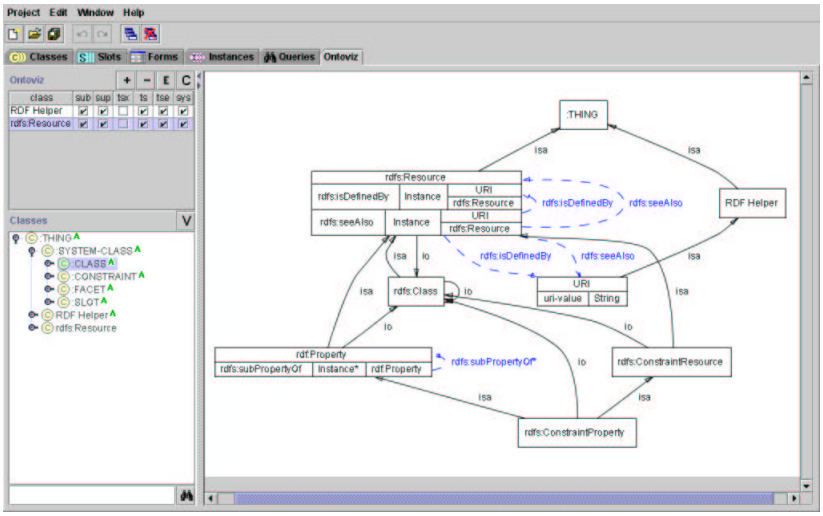

Fig. 4. Protege's OntoViz tab.

The Protege[11] environment supports the modeling of ontologies, being used by hundreds of users world wide, in many knowledge domains. It provides some basic visualisation facilities allowing a user to gain insight in the details of the ontology so that editing it becomes easier. Many of the users of Protege felt the need for tools that provide visualisations of coarser grained structures as well as mechanisms to more easily and effectively navigate the information.

Two plugins extend Protege's basic visualisation. First, the OntoViz tab[12] uses GraphViz to visualise the ontologies (see figure 4). Second, the Jambalaya [13] tab allows people to browse, explore and interact with the knowledge structure. This visualisation allows a more coarse grained visualisation of the ontology class hierarchy, using two different visualisation metaphors. First, a classical tree layout can be drawn, both for the class hierarchy and for the instances. Second, a nested view visualisation allows gradual zooming from general to more specific concepts.

WebOnto [3] is a Java applet coupled with a customised web server, which allows users to browse and edit ontologies over the web. It also allows collaborative development of ontologies. The visualisation was developed for showing a fine grained view of the knowledge. Ontology classes and instances are represented with large, easy to read graphical elements, which makes scalability problematic.

The OntoEdit editor uses a visualisation similar to the GraphViz technique, but mainly focused on supporting interactive editing rather then providing a global overview of large ontologies.

[11] http://protege.stanford.edu
[12] http://protege.stanford.edu/plugins/ontoviz/ontoviz.html

A number of systems exist that use ontological information for the purpose of navigation in large document repositories. We review a few of them. Cougar [6] is an interactive visualisation for information retrieval purposes. It assumes that each document returned by a query has been labeled with one or more labels, indicating the document's main topics. The user interface then allows the user to select maximally three labels (see figure 5). The three document sets associated with these labels are then drawn as a Venn diagram, displaying if and how these document sets overlap. This method of displaying classes and their instances is not only useful as a means to access documents in a result set, but also as a mechanism to study how the classes in the result set relate to each other.

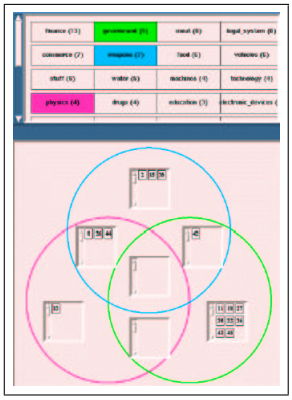

Fig. 5: Cougar.

InfoCrystal [12] is another visualisation technique derived from Venn diagrams that displays classes and their instances. In this visualisation (fig. 6), each class is represented as a labeled node on the circumference of an imaginary circle. Icons are places inside the circle that represent individual combinations of classes (i.e. each icon represent a segment in a Venn diagram). The shape and position of an icon is used to indicate the set of classes it represents. Other attributes, such as size or color, can be used to indicate features such as document density. Provisions have also been made to deal with hierarchically organized classes. Unlike Cougar, the InfoCrystal allows the user to display more than three classes at the same time, but at the cost of a rather cluttered display. An online Flash demo is currently available[13].

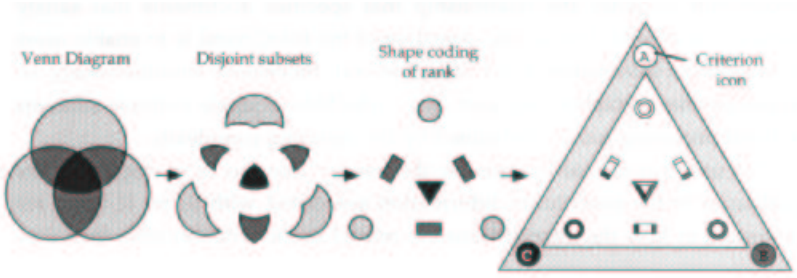

Fig. 6. The InfoCrystal technique explained visually.

[13] http://www.scils.rutgers.edu/~aspoerri/InfoCrystal/demo/InfoCrystal.htm

Conclusions

We observe that the majority of available tools address schema level visualisations. Tools for generic visualisation (The Brain, AquaBrowser) specialize in showing large trees and graphs, which can be seen as very light-weight ontologies (taxonomies). They are mainly used to facilitate navigation of large structures. At the other extreme are visualisations integrated in ontology editing tools (Protege, OntoEdit), allowing the display of the full semantics of a limited part of a schema in order to support the ontology development process.

The number of tools offering instance level visualisations is small. Visualisations in ontology editors often do not make any graphical distinction at the graphical level between schema information and instances. As a result they do not scale to many instances. Also, they fail to show instance level overlaps between classes. Cougar and InfoCrystal are the only visualisations we know of that scale to large sets of instances and that show the overlaps between classes, allowing for instance level analysis of data. Unfortunately they visualise very little schema information.

These observations indicate that there is a clear lack of visualisation techniques that (a) display a simple schema with instances and (b) scale to a large number of instances. We present a visualisation that addresses these issues and that can be used for a variety of tasks.

3 Technology

We have developed the Cluster Map, for visualising populated, light-weight ontologies. It visualises the instances of a number of selected classes from a hierarchy, organized by their classifications.
Figure 7 shows an example Cluster Map, visualising documents from a construction-related domain, classified according to topics discussed in those documents. The big, dark gray spheres represent ontology classes (the topics), with an attached label stating their name and cardinality. When a subclass relation holds between two classes, they are connected by a directed edge. The smaller spheres repre-

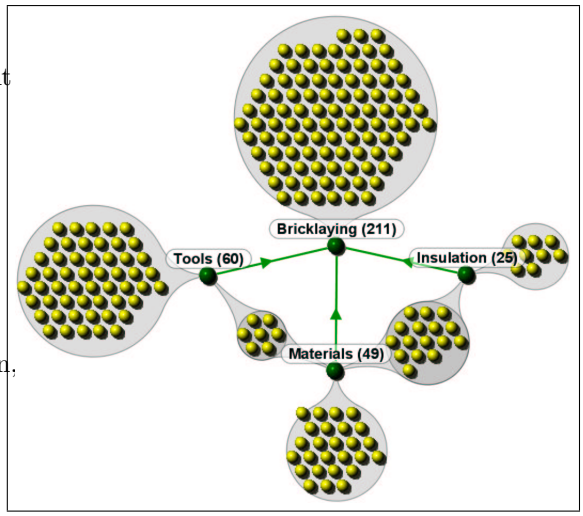

Fig. 7: A Cluster Map example.

sent instances. Balloon-shaped edges connect instances to the class(es) they belong to. Instances with the same class membership are grouped in clusters. Our example contains six clusters, two of them representing overlaps between classes.
Cluster Maps contain a lot of information about the instantiation of the classes, specifically exploiting the overlaps between them. For example, figure 7 shows that the *Insulation* class has a significant overlap with *Materials* (i.e. there are many documents that are both about materials and insulations) but not with *Tools*. Such observations can trigger hypotheses about the available information and the domain in general.

The graph layout algorithm used by the Cluster Map is a variant of the well-known family of spring embedder algorithms [4]. Its outcome results in the geometric closeness of objects indicating their semantic closeness: classes that share instances are located near each other, and so are instances with the same or similar class memberships.

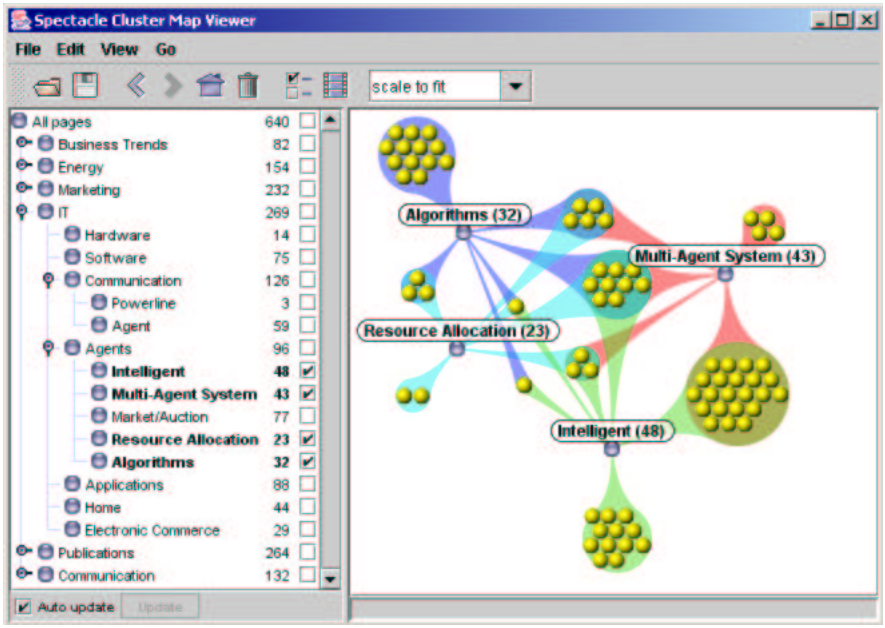

Fig. 8. The Cluster Map's graphical user interface.

The Cluster Map is embedded in a highly interactive GUI, shown in figure 8, which is designed for browsing-oriented exploration of the populated ontology. Users can subsequently create visualisations of a number of classes by marking the check boxes in the class tree on the left pane. The software can animate the transition from one visualisation to the next, showing how the instances are regrouped in clusters. Through interaction a user can also

retrieve information about the specific documents that are contained in a class or cluster.

The Cluster Map is available as a stand-alone viewer and as a Java library. As we will show later on, the visualisation can be fine-tuned in several ways, in order to support certain tasks, improve scalability, etc.

4 Application scenarios

We will illustrate this chapter with examples taken from two realistic applications. One application is a portal generated for a Dutch information provider in the construction industry, Bouwradius (4.1). The second application is developed in the context of the SWAP European IST research project (4.2).

4.1 Bouwradius

Bouwradius acts as an "information broker" between a large number of producers of construction-related publications on the one hand (e.g. websites, magazines) and consumers that wish to receive a tailor-made presentation of all information relevant to them on the other hand (e.g. teachers, students, companies). Bouwradius uses Aidministrator technology to provide portals to several consumer communities. Each portal is optimized for the community's task, view on the data and vocabulary.

The content and structure of the portals depends on two kinds of knowledge structures:

- a thesaurus; a list of approximately 2000 terms, covering the whole construction domain. These terms are used to classify construction-related publications, as described below.
- various concept hierarchies, each modelling the view and vocabulary of a specific user community. Note that these view-specific concepts are not taken from the thesaurus. However there is a link between these concepts and the terms in the thesaurus: each leaf concept in a concept hierarchy is related to a set of terms from the thesaurus. These thesaurus terms define the kind of publications which are relevant for that concept. Higher nodes in the tree are implicitly defined as being the union of their children. A concept hierarchy acts as a structuring backbone for each portal as there is a page in the portal devoted to each concept. Therefore, the content of each portal page is determined by the thesaurus terms associated to the corresponding concept.

As said, the thesaurus is used for document classification. In the past this classification was performed manually by a group of domain experts within Bouwradius. This approach turned out to be very slow and expensive and also gave inconsistent results due to different interpretations of the intended

meaning of terms. Recently we have started to use an automatic classification system (based on machine learning techniques) to perform the classification, resulting in cheap, timely and consistent results. However, extensive training and quality assessment of the system is still necessary to ensure good results.

We have identified several areas in which the visualisation supports management of the portals. First of all, the visualisation can help in acquiring insight into the domain, e.g. by visualising sets of related concepts, allowing the employees to construct an accurate mental model of the available information. Furthermore, they can adapt their view as the information repository changes over time. Finally, visualisation may help in the assessment of the quality of the classification system, e.g. by comparing its results to a reference set of manually classified documents.

4.2 SWAP

The goal of the IST Research Project SWAP[14] is to explore whether peer-to-peer technology can provide an alternative architecture for knowledge sharing on the Semantic Web. This combination of peer-to-peer and Semantic Web technologies promises to be mutually beneficial. On the one hand it will contribute to enrich the search facilities that are currently available for peer-to-peer systems (which are rather limited until now). On the other hand it will give knowledge repositories the capability to provide individual views in a decentralized framework with low administration overhead.

In SWAP each individual PC is treated as a peer. The information of the peer is presented to the SWAP network enriched with machine understandable semantics in terms of the concepts defined by the peer's own ontology. These semantics reflect the peer's individual view on the world. A peer can query the whole network. The peer network uses semantics to enhance query routing and answering. The answers are presented to the user in the terms of his local ontology. Peers benefit from being part of a network: by monitoring the traffic within the network they can evolve their local ontology so that it reflects the knowledge of the community.

Our visualisation technique facilitates the end user's interaction with the SWAP network. First, the user has a graphical overview of his own data, by visualising his local ontology. Second, query formulation and result interpretation are also easier with our interactive tool.

The differences between the two application scenarios described in this section demonstrate the wide applicability of the visualisation. One of the most prominent differences between the two scenarios is the intended user. Whereas at Bouwradius the users are people managing a large information repository, in SWAP we envision end users retrieving and interpreting information for their own purposes using the visualisation. Closely related to this is the difference between supporting trained expert users vs. layman. Finally, the two

[14] project number IST-2001-34103, http://swap.semanticweb.org

scenarios differ in one having a centralized information repository whereas the other is a distributed environment in which the available information is typically only partially known.

5 Tasks

We have used Cluster Maps for supporting analysis, querying and navigation tasks in the two domains described above.

5.1 Analysis

The Cluster Map can be used for a variety of analytical tasks. We define analysis tasks as those tasks where the user's desire is to get a global insight into the information, rather than to filter and retrieve parts of it. Those tasks will be treated in the next subsections.

Most applications of the Cluster Map for analysis tasks can be characterized using the following three parameters:

- The dataset to be visualised (e.g. objects in the dataset can be documents).
- The ontology, defining the main classes of the domain.
- The classification; the assignment of objects (from the dataset) to classes (from the ontology).

We will describe a number of generic analysis tasks using different variations of these three parameters.

One dataset, one ontology, one classification

The simplest application is where there is one dataset, one ontology and one classification of the objects in the dataset. Visualisation of the populated classes shows how objects are distributed among classes and how classes consequently overlap. Patterns and outliers in the data are made visible, allowing for confirmation or rejection of hypotheses about the information, ultimately leading to new and confirmed insights into the information domain.

As explained before in the Cluster Map introduction, using the visualisation is typically an interactive process, where the user creates visualisations of several subsets of the complete ontology. This is necessary because visualising all classes at once will usually lead to a very cluttered and unusable visualisation. This raises the question how to select useful subsets of classes. Often, some amount of domain knowledge is necessary to do this well.

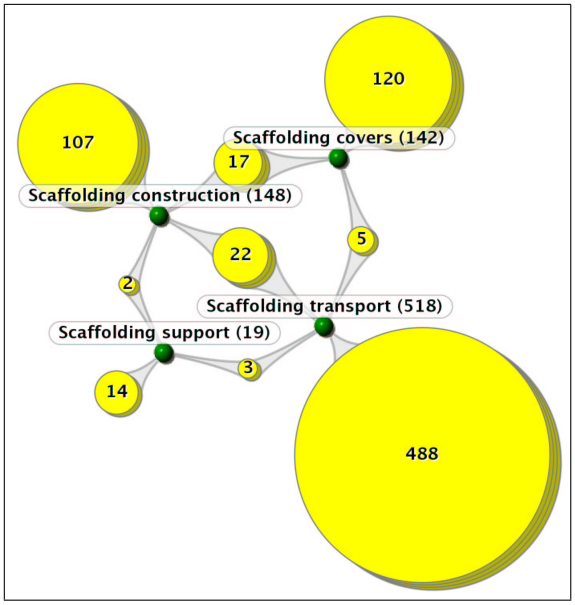

Fig. 9: Classes concerning scaffolding.

In the Bouwradius scenario, we have made visualisations of classes we knew are semantically related (e.g. because they all relate to some kind of construction work), and tried to see if the resulting visualisations could be explained or whether they produced unexpected results. Figure 9 shows an example of classes concerning scaffolding. Note that here we display clusters as a single visual entity rather than its individual members, in order to scale to larger document sets. One observation we can make is that the classes *Scaffolding covers*[15](measures taken to protect construction workers against bad weather) and *Scaffolding support* (measures taken to guarantee a stable scaffolding) do not overlap, even if the other classes overlap with each other. This can be explained because these two aspects relate to different parts of a scaffolding. One thing that could not be explained was the large amount of documents about *Scaffolding transport* that did not relate to any other scaffolding topics. This could trigger a revision of the thesaurus or of the classification system.

One dataset, several ontologies, one classification per ontology

Another strategy for creating visualisations for analysis purposes is to apply several ontologies to the same dataset, thereby allowing for analysis of the same data from different point of views.

Within the Bouwradius scenario, this can for example be done by visualising the same data set according to different view-specific concept hierarchies. Each view will provide answers to different questions reflecting different purposes.

In the SWAP context, a peer can have multiple ontologies ("views"), classifying the same information. For example a user may want to see his files according to file-type in one view, but according to their origin in another.

[15] The presented concepts were translated from Dutch to English by non-professional translators

Several datasets, one ontology, one classification per dataset

One can apply the same ontology on different datasets and see whether the visualisation of a set of classes differs for each dataset. This allows comparison between datasets.

Such visualisations for dataset comparison can be achieved in two ways. One option is to create a separate visualisation for the same subset of classes for each dataset and then compare these visualisations. Alternatively, the datasets are merged into a single dataset and a single visualisation of a set of classes is created. The source of the objects are indicated using e.g. colors or icons.

Figure 10 shows the same classes as figure 9, but this times every cluster is displayed as a pie chart, indicating the amount of documents originating from each particular source. We see that the large cluster at the lower right that could not be explained before is for a very large part originating from a single source. This information may help when refining the thesaurus to make a finer-grained thesaurus.
In the Bouwradius scenario such a strategy may be very beneficial, since the documents that Bouwradius distributes among their consumers originate from a variety of sources. Visualising what information comes from which source will reveal whether certain sources specialize in certain parts of the thesaurus.

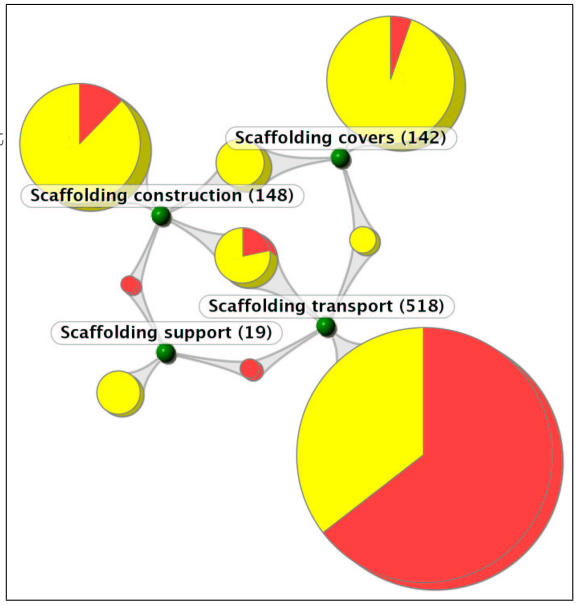

Fig. 10: Documents from different sources.

A slightly different approach is to visualize the differences between the current Bouwradius document repository and all documents from a new source, in order to see whether the new source is a useful addition. For example a new source can contain many documents in scarcely populated parts of the thesaurus and therefore it is a useful addition for Bouwradius' repository.

In the SWAP scenario, peers may be interested in applying their own ontology to documents supplied by a new peer, again to investigate if the data of the new peer is a useful complement to the peer's own dataset. Sometimes a community of peers share an ontology, according to which they organise their data. For example, within a bank, a fixed ontology can exist according

to which experts classify their own documents. By visualising the data of different experts according to the single ontology, one can determine how their knowledge differs according to the terms of the ontology. Certain experts will have more knowledge in certain areas. Also, they might possess documents which cannot be classified by the existing terms and therefore require an update of the knowledge model.

One dataset, one ontology, several classifications

The parameter that has been constant up to now has been the classification: the assignment of dataset objects to ontology classes. In some scenarios there are several "classifiers" active, raising the possibility that the same object is assigned to different classes by different classifiers. We will show that visualising the differences in the classification of a dataset may be useful for a number of reasons.

Until recently, Bouwradius used a large group of domain experts to manually classify documents. When several people classify the same document, there will typically be differences in their judgment of which classes are appropriate, caused by differences in domain knowledge or even their interpretation of the precise semantics of a concept. Visualising these differences can enable management to better streamline the work of the human classifiers.

Recently, we have started applying an automatic document classification system at Bouwradius. In order to guarantee high quality output of such systems, the quality is often assessed by applying the system on a set of manually classified documents. This results in two classifications of the same dataset: the manual classification and the classification produced by the system. Visualising the differences between the two classifications can help us to gain insight in and optimize the output quality of the classifier.

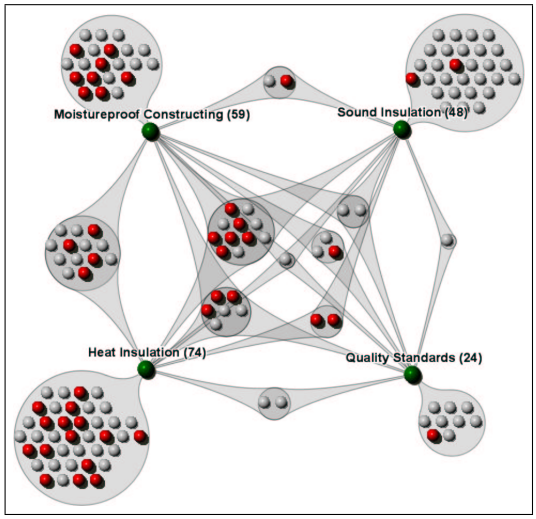

Fig. 11: Visualising classification errors.

Figure 11 shows an example visualisation of four classes taken from a test set. We use a color coding to identify those documents that received a different classification from the system (the darker spheres indicate the errors). This shows a more qualitative overview of the precision of the system compared to the percentage of correct classifications, as is often used.

An alternative is to visualise the classes as returned by the system and high-

light those documents that have a different manual classification. This gives a qualitative overview of the recall of the system.

Of course the recall and precision per class could also be provided in a simple table, listing the two percentages for each class. However, visualising them using a Cluster Map additionally shows how these errors relate to the overlaps between classes. This is an attractive feature because a lot of errors result from the system being inable to properly differentiate between classes that are semantically close (and therefore are likely to share instances). If many errors occur in clusters connected to many classes, this may indicate problems with differentiating between classes. If on the other hand errors are more uniformly spread over the clusters of a class, this may indicate problems with the training set of that single class. In the end, this visualisation has the potential to increase the trust and understanding of the user in the system, because it sheds a light on why the system is making errors.

Still, in order to have a full overview of the quality of the system, one would like to combine the information contained in both visualisations. This visualisation would have to reveal both the produced and the ideal classification of the incorrectly classified documents. How to create such a visualisation is considered to be future work.

Monitoring

Finally, we consider a class of analysis tasks which we call monitoring, where information is analysed as it changes over time. This task cannot really be seen as another configuration of the three parameters we used before. Rather it seems that monitoring introduces a fourth parameter, i.e. time. This means that every previous analysis task may in theory be extended with a time dimension. In the most complex case this leads to a scenario of one or more evolving datasets, one or more evolving ontologies and one or more evolving classifications. However, in practice one or two parameters are often constant, at least on the short term.

For example, in the Bouwradius scenario, it is very useful to see how the total information repository changes with respect to the thesaurus: which classes increase or decrease in size, which overlaps increase or decrease, etc. Possible outcomes are that certain parts of the ontology suddenly start to correlate because of recent events in the domain, or that certain parts of the ontology should offer a more fine-grained classification of documents because of a sudden increase of documents in certain classes.

Figure 12 shows three Cluster Maps, each visualising the same classes at a different moment in time. Not only do we see that certain classes and clusters grow over time, but also that a new overlap of classes is introduced in the last map. This may indicate changes in what is published about these topics.

One of the novelties of SWAP is that knowledge evolves due to interaction between peers. A peer should update its ontology according to the information exchanged in the network. The way the local ontology is updated is an open

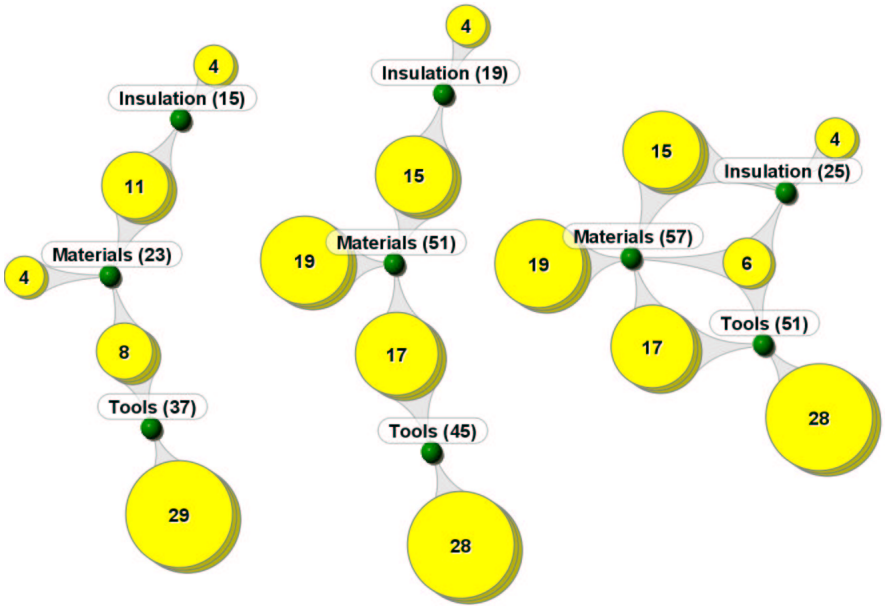

Fig. 12. Three classes changing over time.

issue. A completely autonomous peer would monitor the network and adapt the knowledge of the peer accordingly. This would lead to an always up-to-date knowledge. However, because of the high change rate, the user would not be able to understand the changes. To solve this, i.e. to maintain the user's understanding, the system should ask for acknowledgement for every change. This is an overload that no user would accept: continuous decisions about the knowledge structure. Therefore a balance has to be maintained between the degree of autonomy and usability of the system.

The ideal situation is that the system evolves by itself, but it is able to present the changes to the user in a way that makes them easy to understand. Using the animated transitions between different Cluster Maps one can illustrate the transition from one knowledge state to the other. This lowers the cognitive effort associated with understanding changes of knowledge structures.

5.2 Query

The goal of a query task is to find a narrow set of items in a large collection that satisfy a well-understood information need [8]. The Cluster Map viewer can be applied in the four stages of the search task [11], as described below.

Query formulation - the step of expressing an information need through a query is a difficult task. Users encounter difficulties when having to provide

terms that best describe their information need (vocabulary problem). Furthermore, combining these terms in simple logical expressions using "AND" and "OR" is even more complicated. See [10] for a demonstration of this problem.

In the Cluster Map viewer the classes that describe the domain of a data set are explicitly shown (left pane), making the vocabulary choice much easier (see figure 8). To formulate a query a user only needs to select the classes of interest. There is no need to specify Boolean expressions at this stage since, as we will explain further, the visualisation of the results offers the answer to many frequent boolean queries in an intuitive manner.

Imagine that a user wants to go on holiday to the French Loire, with a group of four persons that would share two rooms in a three star accommodation. One can formulate this query in many ways, using synonyms of the most important search terms. However when using our interactive interface the user easily finds the terms that best describe his needs: "Loire", "4 persons", "2 rooms" and "3 stars".

Initiation of action - After a set of classes are selected in the left panel, the search is launched at a mouse-click.

Review of results - the results of the query are graphically presented to the user. For a set of selected classes the following is shown:

- (1) the union of the classes (disjunction of all query terms)
- (2) all possible intersections of the selected classes (conjunction of some query terms)
- (3) the intersection of all classes (different conjunctions of all query terms) - if it exists. This is a particular case of (2).

Note that the results of simple Boolean expressions are intuitively shown in the map. If the user wants a disjunction of the query terms (1) he will analyse all the presented objects. Indeed, Fig. 13 shows all available holiday destinations which are in Loire, all destinations for groups of four persons, all destinations with two rooms and all offers with three stars. As an added value the user sees how the corresponding classes overlap (2). Fig. 13 reveals sets of items that satisfy certain query terms only, for example all destinations with two rooms in Loire. A more interesting (and probably more frequent) case is when users want the conjunction of the terms. This is also the case in our example. In that scenario, two extreme situations can happen:

- the result set is too large (under-specification)
- the result set is empty (over-specification)

If the result set is empty, the user can at least find objects that partially satisfy the query. The degree of relevance of certain clusters is suggested by their color: the more relevant the darker the shade of the color. We refer to the phenomena of dropping some of the requirements as query relaxation. For our example query, the result returned by the system shows that there

are no items that satisfy all criteria. However, the user can relax his query. Two destinations are still interesting if the customer dropped one requirement (either quality of accommodation or location).

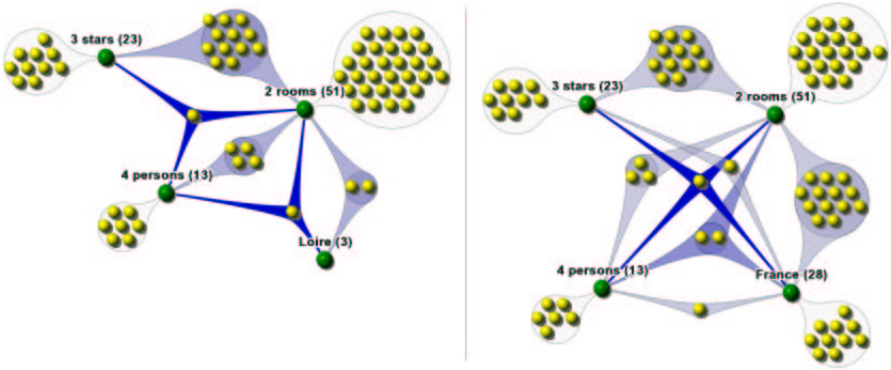

Fig. 13. Using Cluster Map for query relaxation(left) and broadening(right)

Refinement - according to the conclusions of the result interpretation step, the user can narrow or broaden the scope of his search by refining the query.

If the result set is too large, the user can replace some classes with more specific subclasses. Therefore his query is narrowed. At the other extreme, in case of an empty set, some classes can be replaced by their superclass, broadening the scope of the query. Note that both narrowing and broadening the scope of the query are possible due to the ontological nature of the domain description. The Cluster Map viewer facilitates choosing more specific or more general classes.

In the depicted scenario, the user can broaden his search by replacing Loire with its superclass, France (right of Fig. 13). Now there is a single holiday destination that matches all criteria. At the other extreme imagine that the customer would like to go to France and that the system returns a huge cluster that satisfies all the criteria. Instead of looking at each returned object, the customer can refine his query with a more specific class, for example Loire. This would narrow the scope of the query and return a smaller set of options.

Support for the query task is valuable in the context of SWAP. The Cluster Map viewer fulfils the functionality of a query interface between the user and the peer-to-peer system. First, such an interface eases the task of query formulation. Second, graphically presenting the results facilitates a better review of the results. Finally, it allows further actions in case that the user is not satisfied (relaxation/refinement).

5.3 Navigation

Finally, Cluster Maps can be used for graphical navigation of information spaces. We have employed them as image maps in web sites based on ontological data. Two navigation scenarios have been implemented, as described below.

In the first scenario, the Cluster Map is used in addition to another, more traditional navigational structure (a textual tree). It plays the role of a site map that is invoked by the user when needed. It presents an overview of the whole data set: it shows the most important classes, their relations and instances. An interesting aspect is the way the data is accessed: one can access a whole class (click on a class) or an overlap of interest (click on a cluster). The result is a representation of the contents of the selected entity in the form of a textual list. Entries in this list present extra information about the instances as well as a link to the actual data. The role of the map is to facilitate a quick understanding of the available content and to provide quick access to individual items.

In the second scenario, the Cluster Map is always present as the only navigation facility. Maps gradually present deeper levels of the ontology: the user starts at the top of the ontology and can navigate towards more specific topics by clicking the classes of interest (drilling down into the information). At any point, the map shows the current class, its parent and its subclasses. For the current class, its elements are also presented in a textual list. This semantical zooming facilitates a leveled understanding of the data.

6 Summary

We expect the Semantic Web to be no different from other fields of Computer Science: visualisation of data and data-structures can be an important tool. For the Semantic Web, this will mean the visualisation of ontologies and metadata.

The type of visualisation that must be employed depends strongly on the nature of the data to be displayed, and on the task that is to be supported with the visualisation.

Concerning the nature of the data to be displayed, we expect that many of the ontologies on the Semantic Web will be so-called "lightweight" ontologies: simple hierarchies of partially overlapping classes, with little or no cross-taxonomical properties. Furthermore, we expect the number of instances will outweigh by far the number of classes in such an ontology.

It is therefore surprising that most of the visualisation tools in the literature focus mostly on the visualisation of complex ontology schema's without showing the related instances, or vice versa, if they are capable of showing large numbers of instances, then they tend to ignore ontological information.

Having identified this gap in available ontology-based visualisation tools, we have presented a visualisation technique that is specially tailored to visualising light-weight ontologies with large numbers of instances, exploiting both the hierarchical and the overlapping nature of the class hierarchy. After having presented the generic technology for this visualisation, we have shown how this visualisation technology can be employed to support a variety of users in a variety of tasks (data analysis, querying and navigation).

References

1. J. Davies, D. Fensel, and F. van Harmelen, editors. *Towards the Semantic Web: Ontology-driven Knowledge Management*. Wiley, 2002.
2. M. Dodge and R. Kitchin. *Atlas of Cyberspace*. Addison-Wesley, London, 2001.
3. J. Domingue. Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1998, April 1998.
4. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
5. E. Gansner and S.C. North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 1999.
6. M. Hearst. Using categories to provide context for full-text retrieval results. In *Proceedings of the RIAO '94*, Rockefeller, New York, 1994.
7. J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualising large hierarchies. In *ACM conference on human factors in Software*, pages 401–408, 1995.
8. G. Marchionini. *Information seeking in electronic environments*. Cambridge University Press, Cambrigde, UK, 1995.
9. OntoWeb. Deliverable 1.3: A survey on ontology tools, May 2002.
10. B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE symposium on visual languages (VL '96)*, pages 336–343, 1996.
11. B. Shneiderman. *Designing the user interface*, pages 509–551. Addison-Wesley, Menlo Park, 1998.
12. A. Spoerri. InfoCrystal: a visual tool for information retrieval and management. In *Proceedings of the second international conference on Information and knowledge management*, Washington, D.C., United States, 1993.
13. M.A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. In *In Workshop on Interactive Tools for Knowledge Capture*, Victoria, B.C. Canada, October 2001.
14. A. Veling. The Aqua Browser: visualisation or large information spaces in context. *Journal of AGSI*, 3:136–142, 1997.