On the Visualization of Ontologies*

Yannis Tzitzikas
University of Crete, and
FORTH-ICS, Greece
tzitzik@csi.forth.gr

Jean-Luc Hainaut
Institut d'Informatique
University of Namur (F.U.N.D.P.), Belgium
jlh@info.fundp.ac.be

ABSTRACT

The visualization of ontologies and metadata is a challenging issue with several applications not only in the Semantic Web but also in Software Engineering, Database Design and Artificial Intelligence. This paper aims at identifying and analyzing the more principal aspects of this problem, surveying the work that has been done so far, and at proposing novel ideas that are worth further research and investigation. In particular, it describes the main factors that determine whether an ontology diagram layout is satisfying or not, it elaborates the visualization requirements for both medium and large-sized ontology diagrams, and reports our experiences from implementing a semiautomatic methodology for ontology diagram visualization.

1. INTRODUCTION

The emerging Semantic Web [5] already comprises hundreds of ontologies. To support tasks like ontology selection, specialization and instantiation (i.e. resource annotation), we need methods and tools to support both designers and users in finding the desired ontology, in understanding its structure and in using it, e.g. for querying a repository of ontology-based metadata, or for annotating objects with descriptions expressed in terms of the ontology. Hopefully, the Semantic Web trend has rekindled the interest in systems for conceptual modeling and quite a lot of tools have emerged recently that offer visualization for various kinds of conceptual schemas (XML, RDF, OWL), including Spectable (for taxonomy-based sources) Jamabalaya, KAON OI Modeler (for RDF/S), ONTORAMA (RDFS), EROS (RDFS) and also [41] (a short comparative study of these systems can be found in [33]).

A common practice not only in the Semantic Web, but also in database engineering and information systems analysis and design, is to adopt *diagrams* in order to present in a natural and readable way the contents and the structure of an ontology, of a database schema (e.g. ER diagrams [9]), or of any kind of system design (e.g. UML[1]). It follows that the visual presentation of these diagrams is essential. The 2D or 3D layout of the diagram must comply with the way users use to perceive the concepts. For instance, an element should be placed close to the major element it depends on or it is a part of and the hierarchical nature of a set of concepts (e.g. a taxonomic hierarchy) should be suggested by a top-down arrangement of their representation.

Many algorithms have been proposed in the last two decades to produce automatic layout of conceptual diagrams. General algorithms, though powerful, have proved very poor but for very specific schemas, such as those which are purely or nearly hierarchical. It is a widely accepted opinion that the automatic layout facilities offered by current UML-based CASE tools are not satisfactory even for very small diagrams (for more see [12]). Consequently, the vast majority of layouts created today are done "by hand"; a human designer makes most, if not all, of the decisions about the position of the objects to be presented [29]. Concerning the analysis of the problem itself, i.e. the basic and common aesthetic principles for conceptual graph drawing, there are only a few and limited empirical studies on user preferences for conceptual diagrams. For instance, [34] discusses empirically proved user preferences for UML graphs (of small size), while [12] gives a list of aesthetic suggestions for the same kind of graphs. Of course, specialized algorithms for drawing conceptual graphs have also been developed long ago (e.g. see [40]) and recently (e.g. see [8] for UML state charts). Another form of conceptual graphs is formal concept lattices, i.e. the lattices produced by Formal Concept Analysis [18]. These graphs are quite more simple in structure as they are always DAGs (directed acyclic graphs). Some interesting automatic layout algorithms for this kind of graphs based on force directed placement and genetic algorithms are described in [10]. At last, an interesting layout algorithm aiming at minimizing the occupied area and respecting the designer-provided layouts which is appropriate for collections of objects where each object can be visualized using symbols of different size is described in [22]. The visualization of large conceptual diagrams is even less explored. The classical hierarchical decomposition techniques that are used for visualizing large plain graphs (for a survey see Chapter 3 of [35]), have not been applied or tested on

*(Produces the permission block, copyright information and page numbering). For use with ACM_PROC_ARTICLE-SP.CLS V2.6SP. Supported by ACM.

[1]http://www.uml.org/

conceptual diagrams. In any case, the labels of the nodes that correspond to clusters (or to clusters of clusters, and so on), which are quite important for understanding a conceptual diagram, seems to be a spiny issue. Consequently, only manual collapsing mechanisms (like those described in [21]) are currently available for decreasing the visual clutter and for aiding the understanding of large conceptual graphs.

Diagram drawing is not a panacea. It has been recognized long ago that the usefulness of conceptual diagrams (e.g. ER/UML diagrams) degrades rapidly as they grow in size. As the Semantic Web is based on ontologies potentially including dozens of thousands of classes, new methods and techniques should be deviced in order to tackle the problem of understanding and visualizing large diagrams.

In this paper we focus on the visualization of the structural part of ontologies. So our discussion applies to RDF/OWL schemas, ER diagrams, UML class diagrams. The paper is organized as follows. Section 2 discusses visualization requirements for ontology diagrams, while Section 3 focuses on large ontologies. Subsequently, Section 4 proposes a semiautomatic methodology for visualizing ontology diagrams and reports our experimental findings. Finally, Section 5 concludes the paper and discusses issues for further research.

2. WHAT MAKES A DIAGRAM DRAWING A GOOD DRAWING?

Experience shows that the best layout of a diagram rests on three sets of rules:

(I) *Graph-based* drawing rules
These rules rely on general *graph theory* and apply on any kind of graphs (e.g. see [4]). The objective of these algorithms is to make a graph as readable as possible. This includes minimizing the edge crossings, reducing the space occupied by the graph, placing the nodes in an orthogonal grid, displaying symmetries if there is any of them, and satisfying other (in many cases subjective) aesthetic criteria.

(II) *Semantic-based* drawing rules
These rules take into account the semantics of the specific constructs of the data model, like *ISA* relations, *part-of* associations and *one-to-many* associations (e.g. see [40]). They lead to *heuristics* (such as those mentioned in the introductory section) and yield better results for small to medium size diagrams. These rules have triggered much lower research interest.

(III) *Domain-specific* drawing rules
This set of rules are *domain specific* and cannot translate easily into formal rules. For instance, though each `OrderLine` entity equally depends on an `Order` entity and a `Product` entity, experience has shown that users perceive it as being closer to `Order` than to `Product`. This domain still is largely unexplored.

In general, the above rules are in many cases conflicting and no single category, by itself, can provide satisfying layout: for large diagrams the first two lead to unnatural and most often unreadable layouts, while the third one, being manual, is impractical.

One important remark is that we could view all kinds of rules as (a) *constraints* that the desired layout should satisfy, and (b) *objectives* that the desired layout should maximize. Consequently, graph drawing problems could be formalized as constraint satisfaction problems and/or multi-objective optimization problems [26]. For instance, one goal might be: "construct a drawing where all *isA* links point upwards that occupies minimum area and has minimum number of bends and crossings". Unfortunately, most of these optimization problems are computationally hard (a not updated but quite indicative list of graph-theoretic problems that are NP-Complete can be found in [19]) and constraint satisfaction is not a light task either (e.g. see [28]). In addition, finding a language appropriate for expressing this kind of constraints is by itself a challenging and not trivial task[2]. Ideally, we would like to have an algorithm that takes as input a graph and a specification in that language and returns the corresponding layout. But this seems (at least now) a quite utopian objective.

3. REQUIREMENTS FOR UNDERSTANDING AND VISUALIZING LARGE DIAGRAMS

Suppose that one gives us a very large diagram (illustrating an ontology or the database schema of an existing information system) and asks us to understand this diagram without telling us anything about the application domain of the information system. Two basic observations are in order:

(α) The *labels* of the nodes and edges is the basic means for understanding the graph.
Even a small conceptual diagram but with no labels or with labels in Chinese is useless (at least for the authors of this paper).

(β) The main problem for understanding the entire diagram is the *magnitude* of the graph and not its layout. A graph of 100.000 nodes even if it is perfectly positioned, cannot be understood easily (or at all).

The first observation (α), i.e. the critical importance of names, suggests that we must exploit as much as we can the *labels* of the conceptual diagram. This means that methods originated or inspired by the area of *Information Retrieval* [3], of even from the area of *Web searching* [6, 27] could be applied and could prove quite useful in practice.

The second observation (β) suggests that we need methods for generating *focused*, *summarized* and *abstract* views of the graph. Below we discuss three kinds of such methods, namely *context-based browsing*, *filtering*, and *clustering*.

Context-based browsing

This is a quite simple, yet very effective method. Instead of displaying the entire diagram the user could start browsing the diagram starting from any node of the diagram. At each point in time, the neighborhood of the *focal* (selected) node

[2]A mathematical description of these objectives which serve as the set of objective functions of genetic algorithms is described in [10]

is displayed, e.g. as a *star* graph. The user is then able to click on any other displayed node in order to change the focus. In this way, the visualization, and thus the understanding of the conceptual diagram, is done gradually. As the displayed subgraph is not too big we can easily generate an aesthetically good layout (of course, the bigger the radius of the neighborhood is, the more difficult this problem becomes). For instance, this approach (specifically star views of radius 1) has been adopted (and has been proved quite satisfactory in practise) by the browser of the Semantic Index System (SIS) [1] [3] and RDF Portal [14]. The only limitation of this approach is that the understanding of the conceptual diagram is done with the assistance of a computer, i.e. we cannot do it on paper. However, we could also print them on paper using a technique analogous to the way that big roadway maps are printed on books: at every boundary edge of each page we place the number of the page that shows the next part of the map in the corresponding direction (north, south, east, west). As in the case of conceptual diagrams there is no reference system of coordinates, we could use more than 4 pointers at each page and with an appropriate algorithm we could probably print the entire map without having to print one page for every node, but with much less number of pages. Figure 1 sketches graphically this idea. The graph on the left has six nodes and of course we could print it by printing six pages each one displaying the star-view of one node. However, we could also use only three pages as it is illustrated at the right part of Figure 1. This figure shows a set of three star-views of radius 1 that satisfy the following properties: (a) for each node of the graph there is one page that displays all of its adjacent edges, and (b) all edges between any pair of nodes that appear in the same page are displayed.

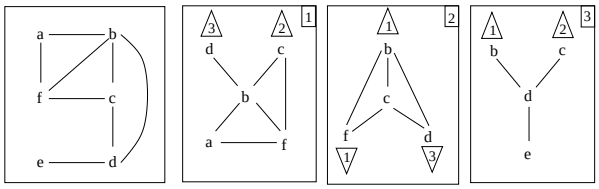

Figure 1: Partitioning a graph of 6 nodes into three star-view like graphs of radius 1

Filtering

Another method to reduce the number of displayed elements is to filter out some of them according to a number of criteria. Some of them are discussed below.

Filtering: Property-based criteria
Here the filtering condition is evaluated upon a property (e.g. the label) of the graph elements. For example, hide/show only those nodes or edges whose label contains (or not) the substring "company". Or hide/show those edges that have the label "1-1" (i.e. those that represent mandatory and single-valued attributes).

[3]Apart from the star-view, the browser of SIS supports hierarchical (tree) views that are appropriate for visualizing and understanding taxonomic or aggregation (part-of) *hierarchies*.

Filtering: Local Connectivity criteria
Here the filtering condition is evaluated upon a connectivity-based property of the graph elements. For example show/hide those nodes whose fan-out/fan-in is lower/greater than a given number (e.g. 2), or show only the relationships of arity 3.

Filtering: Global Connectivity criteria
Here the appearance/occultation of some elements is based on connectivity properties whose evaluation requires considering several nodes and edges of the graph. Below we distinguish some cases:

- *Simple* global connectivity criteria
  For example, show/hide all elements which are connected (directly or directly) with a selected set of graph elements (or show the subgraph of all nodes that can be reached within at most k edges from the focal element). Or, show only those nodes and edges that are related with *isA* relationships with the focus node.

- *Complex* global connectivity criteria
  The ontology query language (e.g. RQL[25] or OWL Lite[4]) can be used in order to specify the desired selection or filtration. However this is not possible if the ontology is unknown to the designer and the user. This means that queries and views (e.g. RVL[30]) could be used only by designers for providing users with various views of an ontology. The existence of a metamodel (this will be discussed later) would allow exploiting the query language in a more generic manner. For example, we could have condition/projection rules that are expressed in terms of the metamodel and thus applicable to any ontology (under that metamodel).

- *Sophisticated* (probabilistic) connectivity criteria (for large diagrams)

  Recall that the Web is actually a huge graph[5]. This suggests that it is worthy to investigate whether the techniques that are applied for Web-searching could be of useful for the problem at hand. As the Web is very big, the *link structure* is exploited in order to deduce the pages that contain valuable information. Actually this is the key difference between Web-retrieval and the classical Information Retrieval. The number of hyperlinks that point to a page provides a measure of its popularity and quality. Also, many links in common between pages or pages referenced by the same page, often indicate a relationship between those pages. This kind of information is exploited in order to identify the pages that have valuable information and thus should be ranked high. For instance, a popular ranking scheme is *HITS* (Hypertext Induced Topic Search) [27]. Another very successful ranking scheme is *PageRank* which is part of the ranking algorithm used by the popular search engine Google [6].

  As the above two techniques work quite well in Web searching, they could be also exploited in the problem at hand. For instance, we could apply the HITS

[4]http://www.w3.org/TR/owl-features
[5]The Google search engine indexes 8.2 billion distinct Web pages (self-reported number, August 2005).

algorithm in order to identify (and display only) the good "hubs" and "authorities" of a very large conceptual diagram. We could apply PageRank in order to display only those nodes whose rank exceeds a threshold. Also notice that these algorithms (either HITS or PageRank) could be exploited for searching these graphs using keywords (in order to rank according to their significance the nodes or edges whose label much the query).

We have already defined a PageRank-like scoring scheme for "taming" very large ER diagrams and it is presented in [45].

Filtering: Formal ontological criteria (or metamodel-based criteria)
Ontological engineering could come into play. Although there isn't any broadly accepted formal ontology (or philosophical theory in general), it seems that there are some notions that show up in one form or another in every application domain, including: *Categories* (e.g. `apples, tomatoes`), *Measures* (e.g. `mass, age, price`), *Composite objects*, (e.g. `cars` have `engines`, `soccer` teams have 11 `players`), *Time, Space* and *Change*, *Evens* and *Processes* (e.g. `raining`), *Physical objects*, *Substances* (e.g. `tomato juice, water`), *Mental objects* and *Beliefs*.

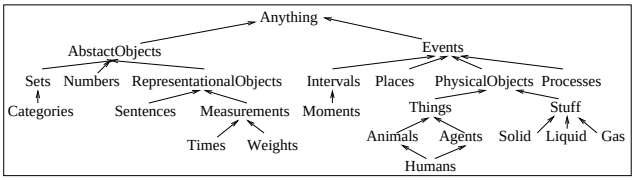

Figure 2: An example top-level ontology of the world

If ontologies were based (or at least annotated) with respect to a formal ontology, e.g. like the one shown in Figure 2 (reproduced from [37]), then we could exploit this ontological information in several ways. For example, we could show/hide some parts of the graph (e.g. show only events, or hide all measures, etc). Moreover, special viewing techniques could aid comprehensibility (e.g. every ontological type could be associated with a different color or symbol).

Unfortunately, current conceptual schemas and ontologies are not based on such an ontological analysis. Moreover, from an existing schema we cannot infer automatically the conceptualization (the intellectual lens) through which the designers of the schema viewed the domain and designed the schema (e.g. see Section 7 of [46]). However, the increasing interest that we observe the latest years around formal ontology (e.g. on top-level ontologies) may affect the form of the conceptual graphs of the next decade.

Another associated remark is that formal ontological information could be expressed as meta-schema information, i.e. the schema could be annotated according to a formal ontological meta-model. For instance, as the semantic model adopted by the SIS system [1], namely SIS-Telos [32], supports unlimited number of instantiation levels (token level, class level, meta-class level, meta-meta-class level, and so on), the viewer of SIS exploits the meta-schema in order to offer readable drawings of big parts of the schema. Specifically, the appearance of nodes and edges in these drawings is based on criteria whose evaluation is based on the classification of the schema to the meta-schema. Apart from this, the browsing the meta-schema itself would aid understanding the domain of a conceptual graph, e.g. whether the schema models the *static*, *dynamic*, *intentional*, or *social* aspects of a domain (according to the classification of [31]). Although UML has a metamodel (see MOF of OMG), this is not useful for our purposes (it does not carry any ontological information).

Filtering: External criteria
For instance, if the ontology (or schema) is populated with data (e.g. annotations or tuples in case we have a relational schema), then we can exploit the data layer in multiple ways. For instance, we can filter out or emphasize the nodes of the graph on the basis of the number of tuples that are associated with them. User feedback obtained during the interaction with the system could also be exploited, in the context of a CAOE (Computer Aided Ontology Engineering) tool.

Clustering

Another method to reduce the number of displayed elements is to form *groups* (or *clusters*) of graph elements, to *collapse* each group into one node, and finally, to display only these composite nodes. This not only reduces the number of displayed nodes but also (and more importantly) the number of displayed edges. Notice that in many cases it is the multitude of edges that makes a graph look messy and unreadable.

Clustering could be performed either manually, semiautomatically or automatically. ER clustering is an example. However, most of the proposals for clustering either require human input [13, 42, 23, 17, 7], or they haven't been tested on large conceptual schemas [36, 2].

In the context of a CAOE tool, the user could use a rectangle to encapsulate the desired graph elements and then ask from the viewer to collapse them into one node. Instead of specifying explicitly the elements of the group the user could specify the desired subgraph by using a number of declaratively (or even procedurally) specified criteria. Roughly, every criterion that could be used for filtering (e.g. any label or connectivity criterion) could be also used for clustering. At last, some forms of clustering could be done automatically, like that the identification of independent (connected) components, cliques etc.

If the above dictated solutions cannot be provided, then the presentation of a large diagram should be done using a *text-based tabular* form. For example, one column could list all classes, while another could list all associations. Selecting one class (resp. association), could make the other list to display the associations (resp. classes) that are connected directly (or indirectly) with the selected element. In addition, double clicking one element could open a star-view for

the selected element.

4. A SEMIAUTOMATIC METHODOLOGY FOR DIAGRAM UNDERSTANDING AND DRAWING

Instead of designing a fully automatic layout algorithm based on semantic drawing rules we decided to design and evaluate an interactive and iterating layout specification mechanism where the designer is actively involved in the layout specification process. There are several reasons behind this choice.

- An interactive process can take into account the user's preferences (including domain-specific preferences), thus it can meet the requirements (I), (II) and (III) of Section 2. Since conceptual graph visualizations are destined to humans (in contrast to other application domains of graph drawing like VLSI design) this is an important argument.
- A gradual layout specification approach is appropriate for studying, evaluating or even inventing semantic-based drawing rules.
- The probability of reaching to a satisfying layout with this approach is higher than trying to invent another fully automatic, but not satisfactory, layout algorithm for conceptual graphs.

4.1 The Layout Specification Process

The steps of this process are illustrated in Figure 3 and are described in brief below. The user can *select* any part of the diagram (from one element to the entire diagram) either manually (mouse clicking or dragging), or by exploiting a set of semantic-based selectors (described in more detail in Section 4.2), and other supplementary techniques (e.g. selection by complement). The user is then able to change the position of the selected elements of the diagram, to *nail down* some elements at desired positions in the 2D space, or to *unnail* elements that are already nailed. The user can then apply one or more automatic drawing algorithms (described in detail in Section 4.3) on the selected part of the drawing. The user is also able to *hide* some parts of the diagram, according to various filtering criteria, or to *collapse* the selected part of the diagram by exploiting a collapsing facility (described in Section 4.5). This process continues until the user reaches a satisfying placement.

We should stress here the need for repository-based CAOE tools that allow keeping and managing several versions of ontologies, ontology visualizations and diagrammatic forms. This means that the above process can be applied several times to the original ontology, resulting in several different visualizations for it. Furthermore, an advanced CAOE tool, could store all the steps of the layout process. This would allow "playing" the sequence of drawings reversely, i.e. from the final (and more brief if collapsing has been performed) to the real (big and complex) one. This would definitely aid communication and understanding. In our case, we have implemented the above methodology by extending the CASE tool DB-MAIN [16, 20].

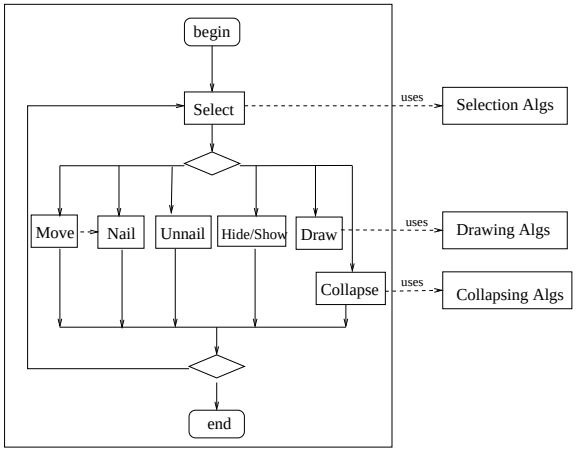

Figure 3: An interactive and iterating layout specification process

4.2 Selectors

We have designed a set of selectors that are appropriate for ER diagrams. The constructs of ER diagrams resemble the constructs used for defining the structural aspect of ontologies[6]. The designer starts by selecting one ore more elements of the diagram as described in the previous Section. Let use C to denote the set of selected elements. Subsequently, the designer can apply one or more selectors that can "grow" C according to various criteria. The following set of selectors were proved useful in practise:

- *IsA-based selectors*: (a) Select SUBtypes, (b) Select ALL SUBtypes, (c) Select SUPERtypes, (d) Select ALL SUPERtypes, and (e) Select entire ISA Hierarchy. These selectors exploit the generalization/specialization relationships of the ontology.
- *Relationship-based selectors*: (a) Select Children, (b) Select Descendants, (c) Select Parents, (d) Select Ancestors, and (e) Select entire Family.
  These selectors are based on the mapping (multiplicity) constraints that are attached to the relationships (or associations) of the ontology (recall weak entities of ER diagrams).

In general all filtering conditions described in Section 3 (e.g. conditions expressed using the query language) can be used to select a part of the diagram. A selector that is best-suited for understanding a very large ontology is described in Section 4.4.

4.3 Drawing Algorithms

On the selected part of the diagram the user can apply a variety of algorithms. Of course, these algorithms do not change the position of the elements that are nailed. In our case the user has at his/her disposal the following algorithms:

[6]Entities, relationships and generalization links correspond to classes, associations and generalization links.

- Force-directed placements algorithms adapted for conceptual diagrams.

  Specifically we provide the algorithm described in [45]. This algorithm is actually a combination of the *spring-model* (proposed and developed in [11, 24, 15]) with the *magnetic-spring model* (proposed in [39, 38]) in a way that is appropriate for ER diagrams. In brief, entity types are viewed as equally charged particles which *repel* each other. Relationship types and *isA* relationships are viewed as springs that *pull* their adjacent entity types. Moreover, we assume that the springs that correspond to *isA* links are all magnetized and that there is a global magnetic field that acts on these springs. Specifically, this magnetic field is parallel (i.e. all magnetic forces operate in the same direction) and the *isA* springs are magnetized unidirectionally, so they tend to align with the direction of the magnetic field, here upwards. Figure 4 illustrates this metaphor. More details and examples are given in [45].

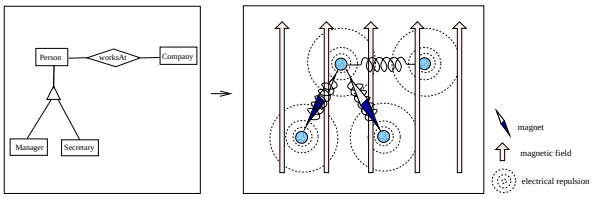

Figure 4: Viewing an ER diagram as a mechanical system

- The *BaryCenter* algorithm. According to the barycenter method, first described in [44, 43], each element that is not nailed is placed at the barycenter of its adjacent nodes. This method spreads nicely the elements of the selected group to the area determined by the nailed elements.

- The *Repulsion* algorithm. Here entity types are construed as equally charged particles which *repel* each other. This algorithm can beautify areas with overlapping elements and does not disturb otherwise the diagram. The final (equilibrium) positions of the graph elements is again obtained using an iterative method.

- The Sugiyama's tree-drawing algorithm. Sugiyama's algorithm is a typical variety of algorithms for obtaining hierarchical drawings of acyclic graphs. It can be exploited by the user in case the diagram at hand has deep *isA* hierarchies. This algorithm gives upwards polyline drawings where the nodes and bends are constrained to lie on a set of equally spaced horizontal lines.

Additional utilities, for scaling the diagrams or for aligning the selected elements horizontally or vertically are also provided. From our experimentation we realized that the employment of force-directed algorithms makes the interactive layout specification process very flexible. The user is not only able to nail down some elements at desired positions but he/she is also able to set interactively various configuration parameters (stiffness of springs, magnitude of electrical/magnetic force). Moreover, and in case a part of the resulting drawing is not satisfying (e.g. due to an unnecessary crossing or to an element placed in the middle of an isA hierarchy), the user could just change the position of some elements - without nailing them down - and rerun the drawing algorithm.

4.4 Understanding Very Large Diagrams

Here we describe a specialization of the general process illustrated in Figure 3 that can aid the user to understand a previously unknown and very large ontology like the one shown in Figure 5. Of course, the Show/Hide step can be combined with any of the filtering criteria described in Section 3. A novel method that falls into this category is the one presented in [45]. The idea is to try to rank the elements of the ontology according to their importance (the objective here is to identify the major elements of the ontology). Subsequently, a top-k selector is used to select the k highest ranked elements and turns the rest elements invisible[7]. A drawing algorithm is then used to specify the layout of these top-k diagrams. In this way we provide layered visualizations of large ontologies which can aid users to gradually explore and understand the ontology at hand. This is achieved by providing the user with top-k diagrams for successively increasing values of k (e.g. 3, 7, 11, 18, etc). Figure 6 illustrates the idea. To further aid understanding, the new appeared elements at each step are visualized using a different color. This interaction scheme proved very useful for understanding an very large and unknown ontology.

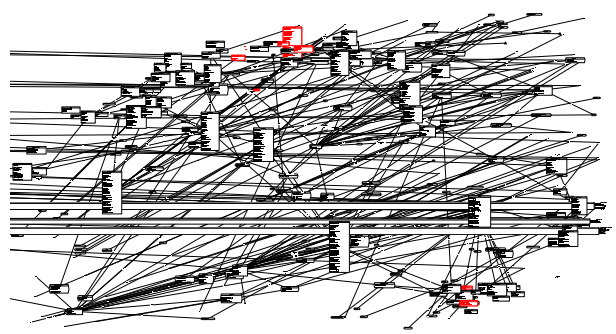

Figure 5: Excerpt (< 10%) of a large ER diagram drawn using a force-directed placement algorithm

4.5 Collapsing

Having selected a part of the diagram the designer is also able to collapse it into one node, hereafter called cluster-node, with a designer-provided name. This node is connected with all nodes that are connected with the nodes of the cluster and it is positioned at the center of the rectangle occupied by the cluster.

Let us now describe in brief how the cluster node should be connected with the rest of the graph. Recall that our objective is to reduce the complexity of the diagram but to

[7]Other variations are also provided, e.g. select the k most highly ranked relationships and those entities that are ends of these relationships.

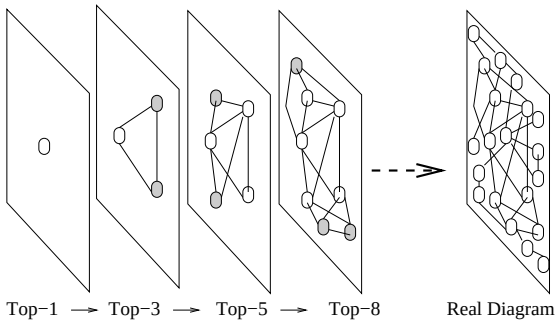

Figure 6: Layered visualization based on top-k diagrams

retain as much useful information as possible. Figure 7 illustrates some indicative cases where relationship types are represented by undirected line segments, isA links with arrows, and dotted rectangles are used to enclose the selected cluster. In the majority of the cases we can only connect the cluster node using a special relationship type. Clearly, example (R1) falls into this case as only relationship types cross the boundaries of the cluster. In (R2), apart from relationship types, there is one isA link but it has to be collapsed as a relation type. It is only in example (I1) that we can connect the cluster node using isA relationships. However, we cannot do the same in examples (I2) and (I3) for obvious reasons. At last, example (IR) illustrates a special case where the cluster is "completed" with all supertypes and subtypes of the nodes in the cluster. This is a quite natural and handy option for the designer.

5. CONCLUDING REMARKS

In this paper we analyzed the more principal aspects of visualizing ontologies. We surveyed the work that has been done in this area and suggested directions that worth further research and investigation. Subsequently, we proposed an interactive layout specification approach accompanied with an arsenal of graph drawing algorithms adapted to the needs of conceptual diagrams, plus a ranking and a collapsing mechanism.

The conclusions drawn from implementing this approach and evaluating it upon several conceptual schemas of real applications can be summarized as follows. Conceptual diagram drawing is not a one-shot task. Therefore CAOE tools should support a semiautomatic methodology for ontology visualization and drawing. Force-directed placement algorithms are well suited for this scenario and they also allow incorporating external knowledge, e.g. domain specific rules expressed in the form of a force. Global complexity reduction operations are very important for understanding big ontologies and for this reason ranking, clustering and collapsing should be supported. We believe that future CAOE should support all the above functions.

Also note that currently ontologies are exchanged in a layout-missing format (e.g. RDF). The same is true for UML as XMI[8] does not capture layout information. So there is an extra need for standards for the exchange of visualization information across different CAOE tools. These standards should not only allow exchanging information (e.g. coordinates) but also force models (with their configuration parameters) and other external information that could be exploited for visualization.

[8]http://www.omg.org/technology/documents/formal/xmi.htm

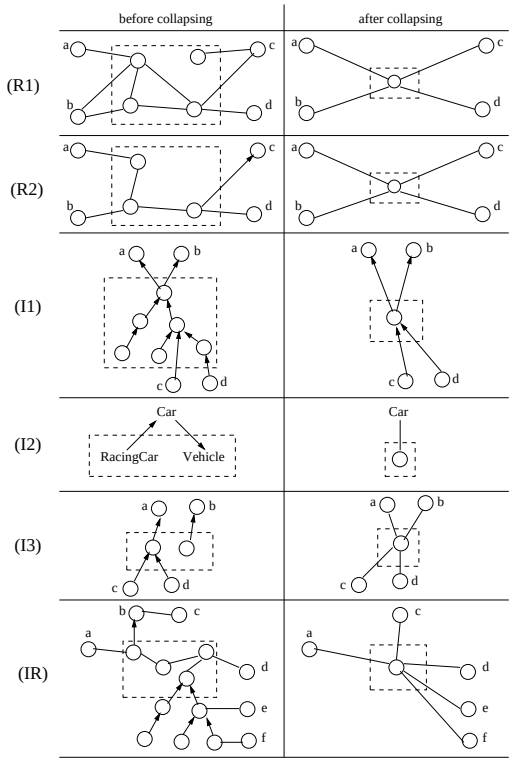

Figure 7: Connecting the cluster node

6. REFERENCES

[1] "The Semantic Index System (SIS)". Information Systems Laboratory, ICS-FORTH, Greece. (http://www.ics.forth.gr/isl/r-d-activities/sis.html).

[2] Jacky Akoka and Isabelle Comyn-Wattiau. "Entity-Relationship and Object-Oriented Model Automatic Clustering". *Data and Knowledge Engineering*, 20(2):87–117, 1996.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *"Modern Information Retrieval"*. ACM Press, Addison-Wesley, 1999.

[4] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis Tollis. *"Graph drawing: algorithms for the visualization of graphs"*. Prentice Hall Englewood Cliffs (N.J.), 1999. ISBN/ISSN : 0-13-301615-3.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". *Scientific American*, May 2001.

[6] Sergey Brin and Lawrence Page. "The Anatomy of a Large-scale Hypertextual Web Search Engine". In *Proceedings of the 7th International WWW Conference*, Brisbane, Australia, April 1998.

[7] L. J. Campbell, Terry A. Halpin, and Henderik Alex Proper. "Conceptual Schemas with Abstractions: Making Flat Conceptual Schemas More Comprehensible". *Data and Knowledge Engineering*, 20(1):39–85, 1996.

[8] Rodolfo Castello, Rym Mili, and I. Tollis. "A Framework for the Static and Interactive Visualization of Statecharts". *Journal of Graph Algorithms and Applications*, 6(3):313–351, 2002.

[9] P. Chen. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[10] Richard Cole. "Automatic Layout of Concept Lattices using Force Directed Placement and Genetic Algorithms". In *Proc. of the 23th Australiasian Computer Science Conference*, pages 47–53. Australian Computer Science Communications 1, IEEE Computer Society, 2000.

[11] P. Eades. "A Heuristic for Graph Drawing". *Congressus Numerantium*, 42, 1984.

[12] Holger Eichelberger and Jurgen Wolff von Gudenberg. "UML Class Diagrams - State of the Art in Layout Techniques". In *Proceeding of Vissoft 2003, International Workshop on Visualizing Software for Understanding and Analysis*, pages 30–34, 2003.

[13] P. Feldman and D. Miller. "Entity Model Clustering: Structuring a Data Model by Abstraction". *The Computer Journal*, 29(4):348–360, 1986.

[14] FORTH-ICS. The ics-forth rdfsuite: High-level scalable tools for the semantic web, 2005. http://139.91.183.30:9090/RDF/.

[15] T. Fruchterman and E. Reingold. "Graph Drawing by Force-directed Placement". *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[16] F.U.N.D.P. "DB-MAIN". (http://www.info.fundp.ac.be/∼dbm/).

[17] Munish Gandhi, EdwardL Robertson, and Dirk Van Gucht. "Levelled Entity Relationship Model". In *Procs of the 13rd Intern. Conf. on the Entity Relationship Approach, ER'94*, pages 420–436, Manchester, U.K., December 1994.

[18] Bernhard Ganter and Rudolf Wille. *"Formal Concept Analysis: Mathematical Foundations"*. Springer-Verlag, Heidelberg, 1999.

[19] Michael R. Garey and David S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. W. H. Freeman and Company, New York, 1979.

[20] Jean-Luc Hainaut. "Transformation-based Database Engineering". In *Transformation of Knowledge, Information and Data: Theory and Applications*. IDEA Group Pub., 2004.

[21] Jouni Huotari, Kalle Lyytinen, and Marketta Niemela. "Improving Graphical Information System Model Use with Elision and Connecting Lines". *ACM Transactions on Computer-Human Interaction*, 10(4), 2003.

[22] Yannis E. Ioannidis, Miron Livny, Jian Bao, and Eben M. Haber. "User-Oriented Visual Layout at Multiple Granularities". In *Proc. of the 3rd International Workshop on Advanced Visual Interfaces*, pages 184–193, Gubbio, Italy, May 1996.

[23] Peter Jaeschke, Andreas Oberweis, and Wolffried Stucky. "Extending ER Model Clustering by Relationship Clustering". In *Procs of the 12th Intern. Conf. on the Entity Relationship Approach,ER'93*, pages 451–462, Arlington, Texas, USA, December 1993.

[24] T. Kamada. *"On Visualization of Abstract Objects and Relations"*. PhD thesis, Dept. of Information Science, Univ. of Tokyo, Dec 1988.

[25] G. Karvounarakis, V. Christophides, and D. Plexousakis. "RQL: A Declarative Query Language for RDF. In *Eleventh International World Wide Web Conference (WWW)*, Hawaii, USA, May 2002.

[26] Ralph L. Keeney and Howard Raiffa. *"Decisions with Multiple Objectives: Preferences and Value Tradeoffs"*. John Wiley & Sons, 1976.

[27] Jon Kleinberg. "Authoritative Sources in a Hyperlinked Environment". In *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, USA, 1998.

[28] Phokion G. Kolaitis. "Constraint Satisfaction, Complexity, and Logic". In *Proceedings of the 3rd Hellenic Conference on AI, SETN 2004*, Samos, Greece, May 2004.

[29] Simon Lok and Steven Feiner. "A Survey of Automated Layout Techniques for Information Presentations". In *Procs of the 1st. Int. Symp. on Smart Graphics*, Hawthorne, NY, 2001.

[30] Aimilia Magkanaraki, Val Tannen, Vassilis Christophides, and Dimitris Plexousakis. "Viewing the semantic web through RVL lenses". *Journal on Web Semantics*, 1(4):359–375, 2004.

[31] John Mylopoulos. "Information Modeling in the Time of the Revolution". *Information Systems*, 23(3):127–155, 1998.

[32] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. "Telos: Representing Knowledge about Information Systems". *ACM Transactions on Information Systems*, 8(4), October 1990.

[33] Arthur Ouwerkerk and Heiner Stuckenschmidt. "Visualizing RDF Data for P2P Information Sharing". In *Procs of the workshop on Visualizing Information in Knowledge Engineering, VIKE'03*, Sanibel Island, FL, 2003.

[34] Hellen C. Purchase, Jo-Anne Allder, and David Carrington. "Graph Layout Aesthetics in UML Diagrams: User Preferences". *Journal of Graph Algorithms and Applications*, 6(3):255–279, 2002.

[35] Aaron J. Quigley. *"Large Scale Relational Information Visualization, Clustering, and Abstraction"*. PhD thesis, University of Newcastle, Australia, August 2001. (http://www.it.usyd.edu.au/∼aquigley/thesis/aquigley-thesis-mar-02.pdf).

[36] O. Rauh and E. Stickel. "Entity Tree Clustering: A Method for Simplifying ER Design". In *Procs of the 11th Int. Conf. on Entity-Relationship Approach, ER'92*.

[37] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[38] K. Sugiyama and K. Misue. "A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm". In *Procs of Graph Drawing Conference, GD'94*, pages 364–375, 1994.

[39] K. Sugiyama and K. Misue. "Graph Drawing by Magnetic-Spring Model". *Journal on Visual Lang. Comput.*, 6(3), 1995.

[40] R. Tamassia, C. Batini, and M. Talamo. "An algorithm for automatic layout of entity-relationship diagrams". In *Procs of the 3rd International Conference on Entity-relationship approach to software engineering*, pages 421–439. Elsevier North-Holland, Inc., 1983.

[41] Alexandru C. Telea. "Visualizing RDF(S)-based Information". In *Procs of IV'2003*, Hawthorne, NY, 2003.

[42] T. J. Teory, W. Guangping, D. L. Bolton, and J. A. Koenig. "ER Model Clustering as an Aid for User Communication and Documentation in Database Design". *Communications of the ACM*, 32(8):975–987, 1989.

[43] W. T. Tutte. "Convex Representations of Graphs". *Proceedings London Mathematical Society*, 10(3):304–320, 1960.

[44] W. T. Tutte. "How to Draw a Graph". *Proceedings London Mathematical Society*, 13(3):743–768, 1963.

[45] Yannis Tzitzikas and Jean-Luc Hainaut. "How to Tame a Very Large ER Diagram (using Link Analysis and Force-Directed Placement Algorithms)". In *Proceedings of 24th Int. Conf. on Conceptual Modeling, ER'2005*, Klagenfurt, Austria, October 2005.

[46] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Mediators over Taxonomy-based Information Sources". *VLDB Journal*, 14(1):112–136, 2005.