Two Approaches to Representing Multiple Overlapping Classifications: a Comparison

Cédric Raguenaud, Martin Graham, Jessie Kennedy
School of Computing, Napier University, 219 Colinton Road, Edinburgh EH14 1DJ, UK
{cedric, marting, jessie}@dcs.napier.ac.uk

Abstract

One of the tasks of plant taxonomy is the creation of classifications of organisms that allows the understanding of the evolutionary relationships between them. In this paper we describe two different data models that have been designed to support two aspects of taxonomic work: the storage of the information and the visualisation of that information. We show that these two models are different because of their constraints and aims, and we compare their abilities using a number of typical tasks users perform. We also show that although different and able to perform different tasks, each of these models is well adapted to its purpose and tight integration is difficult.

1. Introduction

Plant taxonomy involves the definition and manipulation of classifications of plants. These classifications allow a better understanding and cataloguing of the living world by grouping plant *specimens* that exhibit a common set of properties (e.g. morphology, DNA), which can then be used to name and refer to organisms (e.g. legal documents, conservation strategy). Taxonomic classifications are peculiar in that they must capture the fact that some specimens and some groups referred to by a name are used in different contexts over time, i.e. the classifications are multiple and overlap.

A common way to represent classifications is to use graph structures. During the development of the Prometheus project, whose aim was to build a database (DB) and visualisation tool (VT) to support the working practices of plant taxonomists, two distinct data models emerged: a node-based model for the VT, and an arc-based model for the DB system. Although different in their philosophy, these two models represent the same conceptual data. This document compares these two approaches and shows that with complex DB applications the provision of an interactive VT is a major challenge. We show that these two models differ because of their requirements and aims in terms of interactivity, expressiveness, and features.

This paper is structured as follows: section 2 presents the particularities of plant taxonomy classifications. Sections 3 and 4 present the two models that have been designed and explain their respective rationales. Section 5 compares the two approaches and shows their respective advantages and limits. Finally, we conclude in section 6.

2. Particularities of plant taxonomy

Classifications in plant taxonomy are peculiar and unlike many other kinds of classifications (e.g. library classifications). Here we give a succinct description of plant taxonomy that will be useful to understand the mechanisms we describe in this paper and our motivation. The interested reader is referred to [1; 2; 3] for more information.

Plant taxonomy classifications are population-based, i.e. they categorise populations of *specimens* in a hierarchy of concepts (*taxa*). The concepts (*taxa*) that are used for describing the categories are entities that have a life and an importance in themselves, they are not entities that only classify other objects, as is often the case in other classification problems (e.g. [4]). *Taxa* are therefore objects/instances (e.g. names with their publication, authority, taxonomic type information, and their rank or level) that a taxonomist manipulates and publishes. They are volatile and can be redefined (republished with a new taxonomic type for example) or moved in classifications during the process of a revision.

The rank or level of a *taxon* is important, as it forms the basis on which the International Code of Botanical Nomenclature (ICBN [5]) can be applied. The order of the ranks is strict (e.g. *Species* always below *Genus*), but not all ranks are compulsory (e.g. *Section* can be inserted between *Species* and *Genus*). As a consequence, the number of levels in plant taxonomy classification hierarchies can vary from 2 or 3 to 20. The variation in ranks used in a classification must be taken into account when comparing different classifications.

Unlike other population-based classifications and indeed most classifications, plant taxonomy classifications are not is-a or is-of classifications. Indeed, *taxa* are abstract names, e.g. *Apium*, and it would be wrong to say that *graveolens* at rank *Species* is-a *Apium* at rank *Genus* just because it has been placed in *Apium*. Although ranks are central to the process of classifying, it would also be wrong to say that a *Species* is-a *Genus*. Rather, plant taxonomy classifications are placement classifications, i.e. we can say that *graveolens* has been placed in *Apium* (and therefore becomes *Apium graveolens* after application of the ICBN).

In addition, because the characters on which classifications are based are arbitrary, many classifications of the same *specimens*, possibly but not necessarily involving the same *taxa*, occur throughout history, i.e. *specimens* and *taxa* are placed in other *taxa* and classifications. It is important to be able to trace the decisions that led to particular classifications.

Plant taxonomy classifications therefore imply the manipulation of names that can become classification elements (*taxa*), the classification of *specimens* in a non-strict manner but also the classification of the elements used to classify the *specimens*, and the ability to move all these concepts around. Moreover, the classifications appear at instance level, not class level.

3. The database approach

The DB side of the project was to support the creation, representation, and manipulation of all details of plant classifications. The following requirements were noted. Firstly objects should be able to capture all taxonomic data and support all taxonomic processes, secondly objects should be independent from the classification, and thirdly the system should provide general-purpose DB functionality and an ad-hoc query language.

Representing the complexity of objects is necessary in order to support taxonomic data, therefore the semantics must be modelled in the DB. In addition, the system must be able to support complex processes such as derivation of information (e.g. deriving names from sets of *specimens*), integrity constraints (application of the ICBN), and complex processes (e.g. searches).

The representation of classifications independently from the objects that are actually classified is important because these objects are entities that are clearly defined as existing independently from any kind of classification. In addition, this allows the classification system to be implemented on top of an existing DB system and taxonomic schema; requiring the redesign of the DB in order to include a classification would not be practical.

The provision of a generic system is important because of the complexity of the processes associated with taxonomy. Indeed, these processes require frequent recursive exploration of the DB, the reorganisation of the graphs, or the assignment of names to sets of specimens according to the ICBN rules. These processes must be supported by an ad-hoc query language.

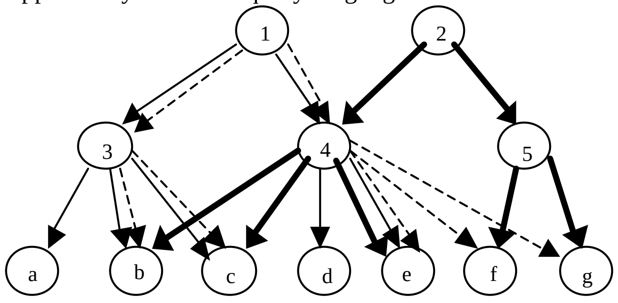

Figure 1. DB - classifications

These requirements led to the definition of an arc-based model for the DB system [6]. Several graph-based DBs exist (e.g. Telos [7], ConceptBase [8], Progres [9], Hyperlog [10]). However, these graphs are simple graphs (especially in the case of Hyperlog where attributes cannot be defined on arcs). They do not support the definition of overlapping graphs that need to be unambiguously identified, as they do not support the definition of semantics for relationships. Our approach is to use relationships as classifying concepts with the equivalent of weights (in weighted graphs) to represent the classification information. We use classes/objects as nodes, and relationship classes/ relationship objects as arcs in an OODB as these are more expressive and provide us with a general-purpose system. These weights (relationship attributes) are not limited to simple integer values: they can be of any type defined in the system (including other arcs). The definition of weights on relationships allows us to describe the distinct trees with their overlaps. Indeed, by following relationships with specific values (e.g. publication information), it is possible to follow a path of a specific graph. But by switching between these values, it is possible to compare and navigate within and among classifications. Figure 1 shows an instance representation of three distinct classifications: a dashed line classification, a thin line classification, and a thick line classification. In taxonomy, these classifications would have been published by distinct authors (the type of arrow in our example represents the publication). The leaf nodes in these classifications could be for example *specimens*. The other nodes could be *taxa* that are used to classify the leaf nodes. We can see in the diagram that the classifications have elements in common: node 3 appears in the thin line and in the dashed line while node 4 appears in all three classifications. Leaf nodes can appear in one classification (node a), in two classifications (node b), or in all three (node e). It is possible to compare nodes 3 and 4 and see that they have some leaf nodes in common. This can give insight in the data (e.g. when two groups partially contain the same specimens, they are partial synonyms). It is also possible to contrast the different meanings of a node according to different classifications: node 4 contains nodes d and e in the thin line classification, nodes e, f and g in the dashed line classification, and nodes b, c and e in the thick line classification.

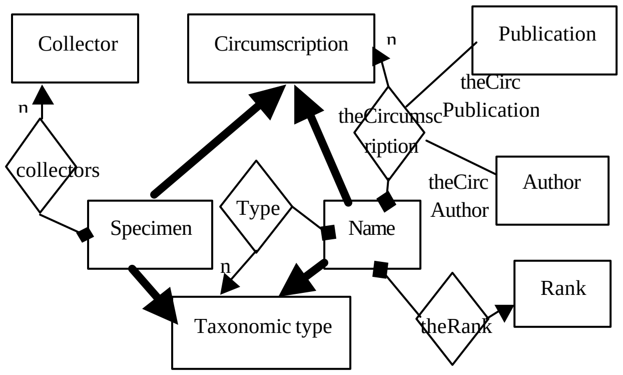

Figure 2. Sample database schema

The provision of relationships as first-class concepts also allows us to clearly distinguish between object and classification information. Our model supports the definition of semantically rich relationships (e.g. aggregation with specific semantics such as lifetime dependency or sharing) which can be used in order to describe the contents of a composite object. The presence of first-class relationships allows us to create specific relationships to represent classifications and manipulate them independently from the classified objects (e.g. tree reorganisation due to the application of the ICBN).

Relationships effectively act as classifiers (or classifying mechanism). The action of creating such relationships between two objects implies that these objects are classified. Furthermore, these relationships are the only objects in the system that are aware of the classifications and they contain all the necessary information to distinguish them from each other. Figure 2 shows a small portion of the DB schema that uses this mechanism and that will be used in the next section where diamond boxes represent relationship classes, square boxes represent object classes. Relationships are directed. Lines that start with a diamond represent aggregation. Thick arrows represent inheritance.

4. The visualisation approach

The visualisation side of the project was to provide a mechanism for the comparison of finished classifications (i.e. where the names of all *taxa* are defined and fixed). The aim of the VT was to enable taxonomists to interactively search and browse the relationships of and correlations between these multiple taxonomic classifications. Rather than rely on traditional GUI approaches such as scrolling lists of results, the interface is instead based on the Information Visualization (IV) [11; 12] approach to interface construction. IV interfaces tend to be highly graphic and designed so that a user can perceive information, rather than having to process it cognitively, which for some information can prove an overwhelming task. Simply put, IV aims to allow people to 'see' information and inferences rather than having to think about it. Our visualisation displays distinct classifications separately, as earlier testing [13] had shown that integrating them into a full global visualisation of the overall graph was incomprehensible to the taxonomists for all but the simplest tasks.

The data used in the visualisation is a simplified subset of the taxonomic data used by the DB model. The structures of all classifications are known in advance and matching elements between them is done on the basis of names only. There is no provision for searching on attributes such as herbarium.

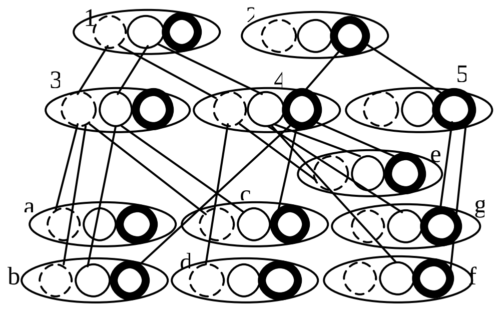

Figure 3. VT - classifications

As *taxon* names are re-used (albeit in different contexts) across classifications, the visualisation data model concentrates name and classification information within nodes. These 'name' nodes are represented as the labelled elliptical entities shown in Figure 3. Data concerning these names but unique to specific classifications, such as parent and child relationships, are allocated to multiple sub-objects within these nodes, with one sub-object describing the state of one *taxon* name within a classification. This data is represented by the inner, circular representations. Figure 3 shows the same classification as in Figure 1, that span the name nodes. Edges themselves are defined as either child (multiple) or parent (one per sub-object) pointers internal to a particular classification, and have no information attached to them. These relationships are shown by the lines between the circles. For example in Figure 3 we can see that 4 is a child of the 1 in the dashed and thin classifications, but a child of 2 in the thick classification. The edges themselves are merely pointers, not objects or 'decision-makers' in their own right. In essence, all the classifications are described separately, and bundled together at given points by the 'name' nodes.

Together the classifications and the linking between them afforded by the nodes build an overall structure we call a DAMG (Directed Acyclic MultiGraph), a restricted class of general graphs related to DAGs (Directed Acyclic Graphs) with the following properties: Directed when following links exclusively from parent to child links. i.e. direction has meaning. Acyclic if and only if directed. Following parent-child links recursively through the structure will never bring a path back to where it started. Accordingly, self-loops are not allowed on a node (an edge with the same end and start node.) Multigraph – Multiple edges can exist between any pair of nodes. This would indicate the same immediate relationship existing between two elements in different classifications. Self-loops are not allowed (a restriction on general multigraph theory to help achieve acyclicity). Layered – The structure can be broken into distinct layers, where no direct links exist between nodes of the same layer. This is implicit in the taxonomy example, where *taxa* at a certain rank are composed only of, and therefore linked to, *taxa* from lower ranks.

Searching and linking within this structure can now take place using the following traversal techniques. Traversal from a particular node for a given classification is simply a question of performing standard depth-first and breadth-first searching mechanisms for trees. Only the tree in which traversal takes place, indicated by a simple integer index, is tracked to keep traversal operations within the correct node sub-objects. Traversal between classifications is simply a matter of switching between the sub-object within the nodes that hold relationship information, and in this way relationships between the classifications can be explored. Perhaps a useful metaphor can be given by describing the classification links as different underground or metro lines as displayed on a map, and the name nodes as “transfer stations” that allow a traversing algorithm to hop off one “line” (classification) and onto another.

This approach to modelling the taxa data gives access to ready-made hierarchies within the overall graph structure, as in effect we construct a restricted graph from hierarchies rather than vice versa, eliminating the problem of extracting individual taxonomic classifications from the overall graph. It also makes display of the visualisation easier, as due to the fact the classifications are kept separated, drawing them separately is a matter of display layout rather than model processing. Furthermore, speed is an important factor for an interactive visualisation, and having all the classifications connected together but easily distinguished makes the operations performed on the visualisation’s data model extremely efficient compared to the case for a general graph.

5. Comparison

This section looks at common operations that are performed by users, compares how they are supported by the two systems, and shows the implications of the two models. In the comparison, we concentrate on the mechanisms of the VT and the query language and model of the DB system.

As shown, the VT and DB models are different due to their different requirements. The VT displays information in an intuitive and simple way that allows users to quickly understand the information presented and to allow exploration. It benefits from the fact that the equivalent of a “*select* * from each classification” is permanently displayed and requires no processing after initialisation. This is in contrast to the DB where a query must be executed each time any data is to be returned to the user. In addition it reduces complex objects to simple text strings for which there is no sub-string searching. The DB however captures complete information, i.e. any ad-hoc query a taxonomist may need to perform. Therefore the underlying schema is more complex, e.g. the description of a *taxon* name is a composite object that contains at least six other composite objects. This is necessary to support features such as the automatic calculation of names. The DB is therefore able to generate data suitable for the visualisation, but the reverse is not true.

The comparison of classifications requires the consideration of groups that appear at similar ranks. For example, if one classification contains ranks Family, Sub-family, Tribe, and Genus, and another Family, Tribe, and Genus, the Sub-family rank in the first is ignored to aid comparison. The VT can work on arbitrary ranks, with bottom level nodes being regrouped accordingly to which ranks are currently marked as active. This enables users to compare directly two classifications with different structures, as long as the leaf nodes are of equal rank. This feature requires more work in the DB and can be achieved via an operator designed to work on graph structures and able to reconstruct graphs after a query has been executed (the *follow* operator [2]). (Q1) shows such a query returning all classifications defined using the theCircumscription relationship restricted to elements that are not at ranks Sub-family.

(Q1) select * from theCircumscription c where c.origin.theRank.destination != “Sub-family” and c.destination.theRank.destination != “Sub-family” follow theCircumscription

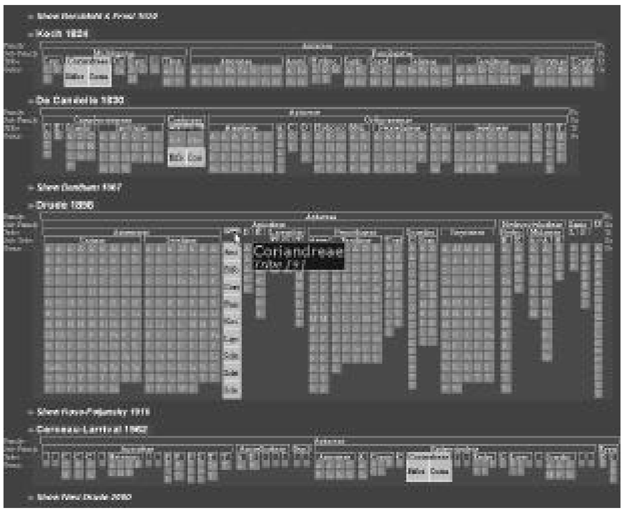

Figure 4. Selection of one node and a group of nodes across four distinct classifications.

When users want to see the occurrence of a particular *taxon* or *specimen* across other classifications, they can click on that element in one classification using the VT. When this happens, the element is highlighted not only in that classification, but also in its appropriate positions within other classifications. This results from the selection of one node within the underlying restricted graph, and is therefore a very simple procedure for the visualisation model. The node selection attribute, which determines display colour in the visualisation, is global for that node across all classifications, so all its representations are automatically drawn in the same colour. This can be seen in Figure 4 by the representative node, Coriandreae, which is highlighted across all classifications. This feature can be achieved in the query language by querying arcs and selecting those that target the specified specimen and returning all the classifications that contain it. (Q2), which is based on the sample schema shown in Figure 2, shows such a query for a *specimen* identified by X (e.g. OID). A similar query can be written for *taxa*. Note that *origin* and *destination* represent the direction of arcs/relationships and are reserved keywords.

(Q2) select c.destination, c.theCircPublication, c.theCircAuthor from theCircumscription c where c.destination = X

When users want to see the distribution of a group and its member groups and specimens across classifications, the whole group in one classification can be visually selected. The selected *taxon* is highlighted along with all the *taxa* and eventually specimens that are its descendants. As the selection attribute is global across all classifications for a node, this operation involves only a depth-first traversal of one classification tree from the node selected to set the attribute in the relevant nodes. This can also be seen in Figure 4. These groups' nodes are then highlighted in the other classifications, showing their distribution. Using the DB, this query can be written as shown in (Q3). It shows the mechanism for a group or specimen identified by X (e.g. its OID). The query first finds recursively all the groups and specimens that were placed in the selected group and then finds all the classifications in which these groups appear. Finally it returns triples that contain a group (*taxon*), information about the classification, and the group associated with it.

(Q3) select c.origin, c.theCircPublication, c.theCircAuthor, c.destination from theCircumscription c where c.origin.(theCircumscription.origin)* = X or c.destination.(theCircumscription.destination)* = X

When users want to dynamically explore the distribution of specimens or groups, brushing can be selected and the mouse moved over the display. Brushing is a temporary visual marking of nodes as the mouse passes over ('brushes') node representations. The nature of the visualisation model, being pre-calculated and enabling node objects to span many classifications, enables these two operations to be performed sufficiently quickly to allow this behaviour. The same processes would be too expensive for a DB system, as a large number of queries would need to be run at each move of the mouse with the display immediately updated. Running a query involves parsing the query, making syntactic and semantic checks, accessing the data dictionary and indexes, fetching objects from the disk, resolving references, and returning objects possibly through a network. On the contrary, the VT only requires pointer following in memory to find the necessary data.

Users sometimes need to see which nodes are unique to a particular classification. In the VT this requires selection of the top *taxa* of all classifications except the one we wish to find unique nodes in. This sequence of events results in the model setting the selection attribute for all nodes, except those that only occur in the classification of interest. The VT can then display this information, usually inverting selection values to highlight the unique nodes. This is the heftiest operation in terms of processing for the visualisation model, involving n-1 (where n equals the total number of classifications) depth-first traversals from the root nodes of classifications. The same feature can be achieved in the DB system by first finding the set of all nodes in a single classification, then checking the nodes that are not referenced by arcs from other classifications.

(Q4) select n from Name n where n in (select c.origin from theCircumscription c where c.theCircPublication = X or c.theCircAuthor = Y) and n not in (select tc.origin from theCircumscription tc where tc.theCircPublication != X and tc.theCircAuthor != Y)

When users want to see how specimens that are grouped together evolve across time, i.e. whether the groups are broken or not in different classifications, the *sibling* mode can be activated. The *sibling* mode operation is more complicated than previously mentioned operations for the visualisation model. The user selects a specimen, and all the specimen's siblings are highlighted using a colour specific to the classification in which the specimen's sibling relation occurs. In the event of specimens being related in two or more classifications, the first classification is taken to decide the colour. Traversal of the tree is limited to one edge up to a node's parent within a classification and then down to possible multiple children within that same classification (performed on every classification related sub-object within the chosen node). Therefore, this selection set is all nodes that share a parent node in at least one classification with the chosen node. The algorithm necessary to perform this task in the DB is a recursive algorithm. Most query languages do not allow queries to call themselves in a parameterised way. Although supporting recursion, the DB system is unable call queries recursively. The only solution would be writing specific code, i.e. it could not be done with an ad-hoc query language.

6. Conclusion

We have described two models designed to support different aspects of taxonomic work which both represent multiple overlapping classifications. They are graph-based, but are radically different: one is node-based and the other is arc-based. The differences in philosophy are due to the way the two systems interact with users and their requirements.

The model influences the ability of the system to respond to specific user requests. For example the visualisation interface is a fast, efficient system that supports rapid response to user actions. This is achieved by specialised, optimised code written as the interface is hard-coded with knowledge of what users expect to do. However, we have also seen that this specialised code does not support all information useful to taxonomists. On the contrary, the DB system is slower than the visualisation interface due to the lack of specialised code, which makes it unsuitable to support some of the features of the VT. This general-purpose approach allows the system to support a wider range of information (e.g. updates) and to answer a broader range of queries.

We have also seen that the degree of interactivity with the user is important. If it is high (e.g. during exploration in the VT), the user acts as a processing unit. If the visualisation makes use of user perception, e.g. by representing classifications explicitly, processing regarding classifications is moved from the VT to the user. An earlier prototype where the visualisation was a graph similar to the arc approach used by the DB was shown to be ineffectual by taxonomists [13]. The DB however is more independent from the user, therefore needs to capture more concepts, making the queries more difficult to express and process.

The choice of model depends on the requirements defined for the system. In Prometheus, both are used: the DB is used to assist taxonomists in the process of creating classifications by checking data, deriving additional data and manipulating it; the VT uses data exported from the DB for exploratory purposes. However, a major challenge in DB/visualisation research is developing VTs that interact directly with the DB and accommodate updates in the data dynamically.

7. References

[1] M.R. Pullan et al., "The Prometheus Taxonomic Model: a practical approach to representing multiple taxonomies," *Taxon*, vol. 49, no. 1, 2000, pp. 55-75.

[2] C. Raguenaud, J. Kennedy and P.J. Barclay, "The Prometheus Taxonomic Database," *Proc. IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000)*, Arlington, Virginia, USA, November 2000, pp. 63-70.

[3] C. Raguenaud, J. Kennedy and P.J. Barclay, "The Prometheus Database for Taxonomy," *Proc. Scientific and Statistical Database Management (SSDBM 2000)*, Berlin, Germany, February 2000, pp. 250-252.

[4] A. Pirotte et al., "Materialization: a powerfull and ubiquitous abstraction pattern," *Proc. Very Large Data Bases (VLDB'94)*, Santiago, Chile, 1994, pp. 630-641.

[5] W. Greuter et al., *International code of botanical nomenclature (Tokyo Code)*, Koeltz Scientific Books, 1994.

[6] C. Raguenaud and J. Kennedy, "Relationships as classifiers," *Submitted to DEXA*, 2001.

[7] J. Mylopoulos et al., "Telos: Representing Knowledge About Information Systems," *ACM Transactions on Information Systems*, vol. 8, no. 4, 1990, pp. 325-362.

[8] M. Jarke et al., "ConceptBase - a deductive object base for meta data management," *Journal of Intelligent Information Systems. Special Issue on Advances in Deductive Object-Oriented Databases*, vol. 4, no. 2, 1995, pp. 167-192.

[9] A. Schürr, *PROGRES, A Visual Language and Environment for PROgramming with Graph REwriting Systems*, Technical Report AIB 94-11, RWTH Aachen, Aachen, Germany, 1994.

[10] A. Poulovassilis and M. Levene, "A nested-graph model for the representation and manipulation of complex objects," *ACM Transactions on Information Systems*, vol. 12, no. 1, 1994, pp. 35-68.

[11] N. Gershon and S.G. Eick, "Visualizations new tack: Making sense of information," *IEEE Spectrum*, vol. 32, no. 11, 1995, pp. 38-56.

[12] S.K. Card, J.D. Mackinlay and B. Shneiderman, Eds, *Readings in Information Visualization: Using Vision to Think*. The Morgan Kaufmann Series in Interactive Technologies, Morgan Kaufmann, San Francisco, 1999.

[13] M. Graham, J.B. Kennedy and C. Hand, "A Comparison of Set-Based and Graph-Based Visualisations of Overlapping Classification Hierarchies," *Proc. AVI 2000*, ACM Press, Palermo, Italy, May 23-26, 2000, pp. 41-50.