EulerView: article organisation within the ACM Classification

Rosario De Chiara
ISISLab - Dipartimento di Informatica
ed Applicazioni "R.M. Capocelli"
Università di Salerno - ITALY
dechiara@dia.unisa.it

Andrew Fish*
School of Computing, Mathematical
and Information Sciences,
University of Brighton - UK
Andrew.Fish@brighton.ac.uk

Abstract

EulerView is a resource management tool which supports operations, such as intersection, on categories. We relate EulerView displays to sets of paths in the transitive closure of the directed dual graph of an Euler diagram, and provide methods for transforming hierarchies with symbolic links into an EulerView display. Linking the display directly to the file system enabled the export of both the directory structure and resource placement which enables users to build small manageable views of their categorisation structure within which to place resources and then export this onto the larger file system. Using the transformation methods developed, we consider the importation of ACM classification into EulerView, enabling users to categorise and store resources such as articles within the imported ACM classification. EulerView is tailored to be appropriate for the ACM CCS application, and new techniques for focus and navigation within EulerView are developed.

1 Introduction and Background

In [7], EulerView was proposed as a means to enable the common user to easily capture the naturally non hierarchical organization of website bookmarking. We propose the extension of this idea, importing existing (non-hierarchical) classifications into EulerView. This will enable user to place their resources within an existing classifications structure using EulerView. Furthermore, it enables them to manipulate the classification structure within the EulerView representation, allowing them flexibility of representation according to their preferences. This will help to address the need for a common method of viewing and interacting with a range of different classification systems (or ontologies). From a resource management perspective, we are enabling the ability to place resources within existing classification structure, to enable user manipulation of such structures, and to facilitate user searching and browsing of resources placed within these structures. Allowing the export of resource placement with EulerView to the file system addresses the dual needs of users in classifying resources within a classification system as well as in constructing a part of the file-system structure that captures the system.

We investigate the use of such techniques for the ACM Computing Classification System (ACM CCS) [1] of research articles. A useful, detailed description of the ACM CCS can be found at [19]. Authors are required to find the appropriate place within the ACM classification structure to classify it. Individual authors may wish to be able to store and retrieve their own papers (or all papers that are deemed relevant to their interests) according to this classification, or a local part of this classification. Similarly, research group leaders or administrators, research departments or institutions might want to keep track of larger sets of relevant articles. The ACM CCS is an important classification system, but there are other potentially useful classification systems in this area. In [19], they propose a unified classification system for papers related to three computing disciplines: computer science, software engineering and information systems. The rationale behind their research is that "the three disciplines share substantial bodies of knowledge and ... existing classifications do not adequately address their combined needs."

1.1 Categorisation and Classification

Recall the following distinction between categorization and classification [13]: "Categorization divides the world of experience into groups or categories whose members share some perceptible similarity within a given context", where category composition depends on the context and on the user of the organization; "classification involves the orderly and systematic assignment of each entity to one and only

*Funded by UK EPSRC grant EP/E011160: Visualisation with Euler Diagrams.

one class within a system of mutually exclusive and non-overlapping classes", where classification usually refers to the assignment of a resource (e.g. a document, URL or photo) to a single class, among classes, often hierarchically organized making clear the complete relationships amongst classes (see [4] for a typical biological example).

The process of categorisation is generally perceived as being less precise than classification: the placement of an item within a classification structure indicates precise global information about that item, whereas placement of an item within a categorisation structure may represent partial information about the item, which is to be interpreted locally, or within a given context. Since hierarchical classification structures are often not sufficient for user-classification needs (and this is even felt to be the case in a real office setting [18]), one can consider non-hierarchical classification structures, such as polyarchies [16] or EulerView [7]. Non hierarchical classifications fit into the classification definition given above if one thinks of two overlapping classes as three classes: two non-overlapping classes together with another class for the intersection (see section 2). It also fits with the notion of categorisation: items in the intersection of categories share some property. So, non-hierarchical classifications could be thought as living somewhere in the middle of this Categorisation-Classification spectrum, allowing the overlap of categories but with a formal underlying model.

In general, strict classification system interfaces are especially useful for storage and retrieval of information, but are often thought to be too restrictive for users, since: they may be time consuming to use, they may get very large and the visualisations unmanageable very quickly, and changes in the classification structure over time may require the update of all existing resources that are already classified – which may be a difficult chore. On the other hand, categorisation systems such as free-form tagging are much easier and quicker to use, but they lack structure and they require the use of different representations, such as tag clouds [11], to browse through existing resources, requiring a change of representation system from their input.

We advocate a "best of both worlds approach", by keeping an underlying (non-hierarchical) classification system, but presenting to the user interactive views of that data in the form of a simple categorisation structure which may be user-created and easily manipulated. This will bring the benefits of a classification structure, especially in terms of information retrieval, but with the flexibility associated to a categorisation structure.

1.2 Related techniques

Faceted classification is a well-known paradigm that enables a user to navigate a hierarchy of items which are sorted by their facets (certain classifiable characteristics). The idea is to let the user refine a particular visit in the hierarchy by iteratively specify the desired value for each facet [17]. EulerView shares with the facet classification system the idea of providing a means to customize the visualisation of a set of items according to the user's needs. A major difference between faceted classification systems and EulerView is that EulerView uses the concept of *overlapping categories*, which is not available in any hierarchical visualisation techniques; it might be worth investigating integrating the ideas of EulerView related to overlapping categories with faceted browsing. *Treemaps* [5] are common method for visualising hierarchical categorisation structures. They were designed for the purpose of representing large datasets of documents via a suitable space subdivision method, providing in one image an impression of the distribution of a certain property amongst these documents (e.g. the size of every document). The current version of EulerView does not support the use of numerical values together with the classification, although this might be worthwhile. More importantly the main ideas of using multiple labels and reifying non-hierarchical classifications does not need to be limited to TreeView (as with EulerView) but could be used with other hierarchical visualisation techniques where the labelling can be integrated.

2 Reifying non-hierarchical models

The EulerView display [7] was designed, with user interaction in mind, to be more flexible than than a strict classification system and to have an appearance similar to Treeview to aid in accessibility due to its familiarity. The main idea was that each category vertex corresponds to a set of tags with which a resource can be tagged easily, but that a user may construct multiple paths to the view of the same virtual folder (which stores links to the resources). Each category vertex can have multiple labels so that non-hierarchical structures can be displayed as a tree-structure. A formal description of an EulerView display can be found at [9]; they also provide the formal description of the storage model but we do not utilise this here. In this section we describe a method for converting Euler diagrams into EulerView displays. To elucidate on the concept of EulerView we indicate the relationship between EulerView and Euler diagrams. The Euler diagram model is a common non-hierarchical representation, which traditionally uses curves in plane to represent sets (or categories) and uses the spatial properties of containment, intersection and disjointness to represent set containment, intersection and disjointness. For example, the Euler diagram on the left of Figure 1 depicts that E is a subcategory of D, that C is a subcategory of $A-B$ and that D is disjoint from $A \cup B$. Now, within the

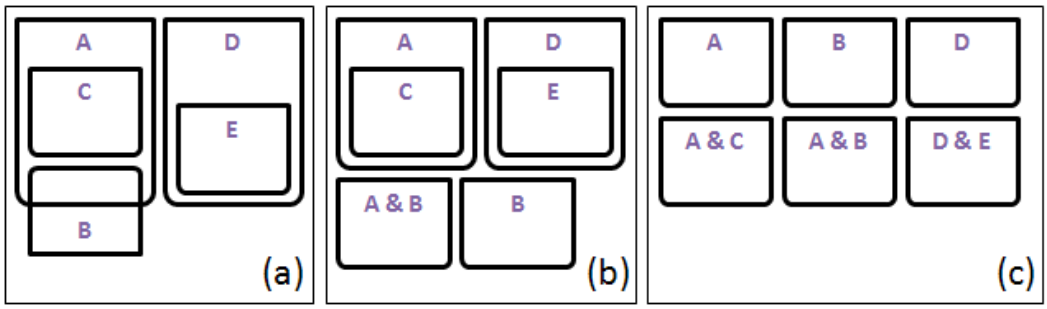

Figure 1. (A): A traditional concrete Euler diagram with 5 categories and 7 zones; (B): A nonstandard, modified Euler Diagram part-way between the two extremes – it has no intersections but utilises the containment relationship; (C): A depiction of the list of zones, or the abstract Euler diagram.

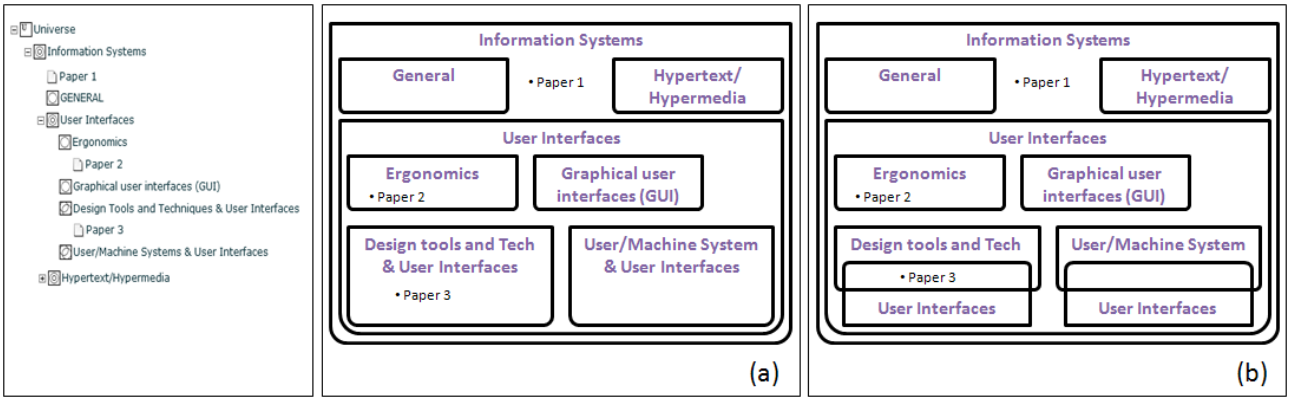

Figure 2. An example of EulerView categorizing papers within the ACM. (a) A modified Euler diagram matching the EulerView (b) a traditional Euler diagram of the same model.

context of the diagram, a resource placed in a zone indicates the tags associated with that resource. So for instance, if a resource r is placed within the zone $A\&B$ (i.e. the region which is inside both A and B only) in Figure 1 then it will have tags A and B but not tags C, D or E.

When formally dealing with Euler diagrams, the abstract diagram is used, which is essentially just a list of zones in the diagram. The abstract diagram for this example, with a slight abuse of notation to improve readability, can be written as: $\{U, A, B, D, A\&B, A\&C, D\&E\}$, where each string indicates the set of curves that the region is contained within and U indicates that the zone outside all of the curves. A visualisation of an abstract diagram, albeit not a very useful one, is shown at the right of Figure 1, where each curve represents a single zone (and so these curves are disjoint). The important distinction here is the label representation: the interior of the curve A will represent the set of resources that are tagged by A only, whereas on the left of the figure, in the traditional Euler diagram, the curve A represents the set of resources tagged by A as well as those in $A\&B$ and in $A\&C$.

The middle diagram in Figure 1 is a modified Euler diagram which depicts a representation that is partway between the two extremes, with the intersections of categories being viewed using zone based descriptions: the intersection category $A\&B$ in the left diagram has been replaced with a curve labelled $A\&B$ in the middle diagram. Notice that if we deleted the curve labelled by $A\&B$ from the middle diagram then we would obtain a traditional hierarchical view indicated by the containment relation.

Hierarchies are commonly captured by a tree structure and an Euler Diagram model can be compressed into a tree-like view by keeping the hierarchical part of the model and reifiying the overlapping category part so that we can display it as a hierarchical view. This provides us with a representation that is partway between the two extremes, with the category intersections being viewed as zone based descriptions.

This construction of a modified Euler diagram via the decomposition of intersections gives one canonical decom-

position of an Euler diagram. Allowing the use of multiple labels on tree nodes (which indicates the intersections of categories) enables us to display this representation within a tree structure. For example, from right to left, Figure 2 displays: a traditional Euler diagram, a modified Euler diagram, and the corresponding EulerView representation. Here, the EulerView display used in Figure 2 was actually a user-constructed view and the Euler diagram was constructed from it. One can construct smaller views from the Euler diagram, and it is important to realise that it is only necessary to display a set of zones which are required as the set of tags of a resource; that is, we do not need to display any more zones (as category vertices), unless the user wants to. The usual meaning of the placement of a resource within the EulerView is simply that it has that set of tags within its local context. The user can manipulate the local context to show different views of the underlying structure.

Now, for a traditional Euler diagram one can consider its directed dual graph. As the size of the underlying diagram increases, this structure may become large, but for a modified Euler diagram there is only one path to each vertex in this dual graph. The EulerView representation called the *intersection view* is obtained by taking the directed dual graph of the modified Euler diagram. In general, any set of paths in the transitive closure of the directed dual graph of the original Euler diagram gives rise to an EulerView display.

For example, in Figure 3, the top three diagrams show two categories A and B together with their intersection $A\&B$. Overlaid is a selection of paths in the transitive closure of their directed dual graphs; they are directed from the vertex outside all of the contours inwards. That is, if level n denotes a region which is inside n contours then consecutive nodes in each path increase in level. Taking the transitive closure allows us to "jump" between vertices with a difference in level of more than one (e.g. we can traverse from U to $A\&B$ in one step in the top left of the figure, passing over two contours rather than one). The middle and bottom rows of the figure show the corresponding modified Euler diagrams and EulerViews respectively. The left hand side shows the intersection view, whilst the other two correspond to different sets of paths called the *projection view* (middle) and the *combined view* (right). In section 3 we provide schema for the automatic generation of these different EulerView displays within the ACM CCS application.

3 ACM classification system in EulerView

We provide a small excerpt of the ACM CCS, together with a simple schema indicating the translation into possible alternate EulerView representations. In the ACM CCS we have the main hierarchy utilising the `isComposedBy` relationship and this maps naturally to a normal Treeview.

Table 1. Basic Schema for the EulerView display options.

ACM CCS Relationship	EulerView Relationship	EulerView Relationship	EulerView Relationship
`isComposedBy`	Containment	Containment	Containment
`isRelatedTo`	Projection	Intersection	Both

The relationships `isRelatedTo` are effectively symbolic links across the original hierarchy. As such we must decide what to map them to in our EulerView display. Table 1 shows a simple schema which can be used to translate the ACM CCS into the three different EulerView displays: using projections (which can be used to directly simulate the symbolic links), intersections (using multiple category vertex labelling) or both, as is shown in Figure 4. The containment relationship in Table 1 indicates the use of the main hierarchical tree structure.

Figure 4 shows the process of passing from the ACM CCS (a) to the combined EulerView (c) via a suitable XML translation (b) of the original ACM CCS XML code. For this combined view, we translate a "related to" tag in the ACM CCS to a projection node in the corresponding position of the EulerView XML together with an intersection node in the next available place at the same level of the hierarchy as the projection node. The highlighted region in Figure 4(a) shows a symbolic link between two categories indicated by the "related to" relationship. Figure 4(b) shows the translation of the excerpt to the XML used by EulerView where the symbolic link is translated to both an intersection and a projection, and Figure 4(c) shows the combined EulerView representing the excerpt. Note that in the translation we have extended the id system of the ACM CCS to number the subject descriptors.

Article placement. We allow an article to be placed in any category and we allow multiple copies of an item to be placed in different categories. This enables a user to be able to place an item in separate categories rather than having to create the complete intersection category. To assist the user in finding all of the categories associated with an item we allow the user to cycle between the different placements of the items. For deletion of an item, there is an option to delete that particular placement of the item or to delete all placements.

This version of EulerView differs from say, Eulr [10], used for photo tagging where the set of all tags in the path back to the root node are assigned to the placement of a photo, since here we wish to only associate the most specific set of identifers (ids) in the ACM CCS to an article. That is, we wish to associate the set of most specific ids in the path back to the root (one could do this by post processing, as-

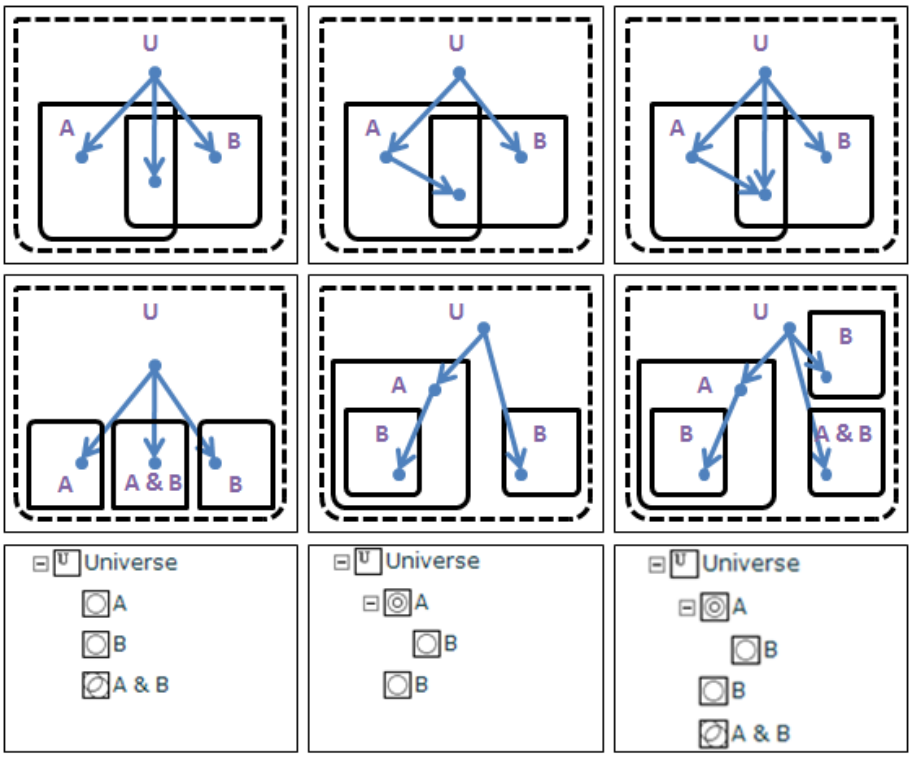

Figure 3. Top: paths in the transitive closure of the dual graph of Euler diagrams; Middle: modified Euler diagrams which have no intersections; Bottom: the corresponding EulerViews, termed the intersection view, the projection view and the combined view.

sociating all of the ids and then removing the unnecessary ones).

Focussing techniques. To assist users in article placement, or searching or browsing tasks, especially using large EulerViews we attempt to reduce the amount of information displayed according to a user setting of context. *Breadcrumbs* [6] is a well known technique that is used to convey to the user their location within a website structure by displaying a trail of crumbs from the homepage to the current page (this is one possible path from the root-node within the website graph). As was already present in [7] we show a "trail of categories" which is the *context* that the currently selected node of EulerView refers to. Further simple aids have recently been incorporated into EulerView such as a *zoomable universe*: a user can zoom in on a category vertex of the EulerView display, displaying that vertex in the place of the Universe vertex, thereby setting that node to be the local context. Whilst zooming the user is aware of the context via the breadcrumbs which always refer to the Universe-rooted display. We also allow simple text searches, automatically zooming in on the next category matching the string, cycling through occurrences of the string upon repeating the search.

A novel focussing technique is that of *branch compression* [1] which allows users to compress a path in the EulerView display to a single vertex, restricting the amount of information in one view. This is a natural option in EulerView since we already allow multiple labels associated to a single vertex. The display on the compressed branches shows the nodes in the compressed branch separated by colons to indicate the path for normal category nodes and an ampersand for projection nodes. This distinction is made here because in the ACM classification setting we are only really interested in the most specific id of the node in each branch of the ACM CCS. So, the placement of an article in a compressed branch node should only be placed at the most specific positions upon decompression. For tagging settings where a resource would be tagged by all labels in path back to root, such as for tagging photos in Flickr using Eulr ([10]) we could simply display the conjunction of all node labels instead.

In Figure 5, we show the effects of branch compression: the branch from Information Systems to Graphical User

[1] branch compression is a new technique but it is also briefly mentioned in the survey at [8], as is the zoomable universe.

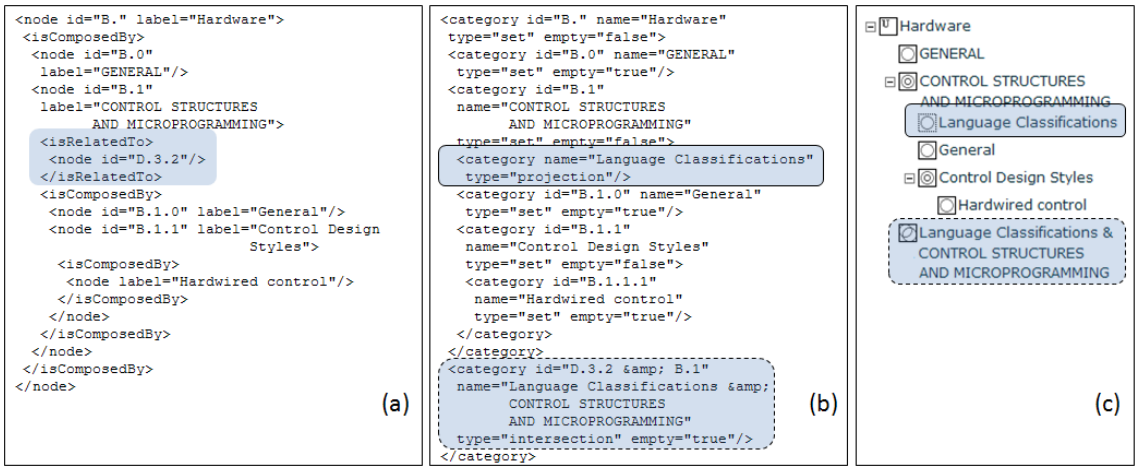

Figure 4. (a) an excerpt from ACM CCS, (b) translation to XML and (c) an EulerView representing the excerpt, showing the combined view.

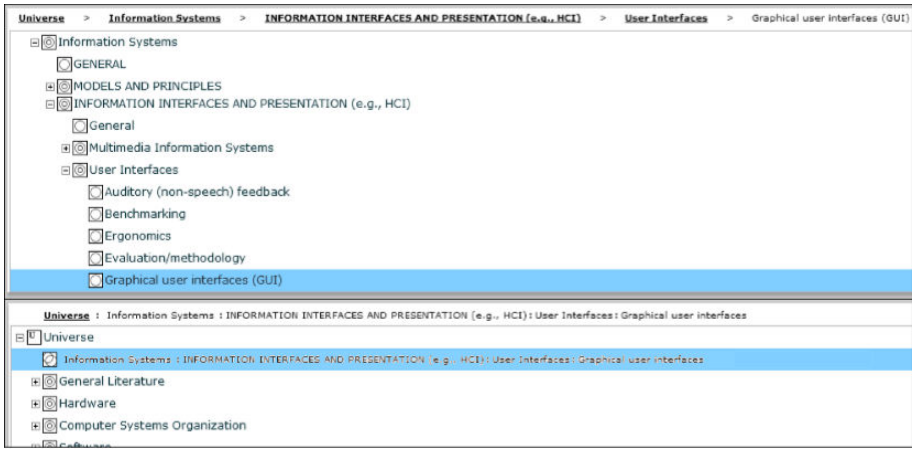

Figure 5. Branches of the EulerView can be compressed to simplify the view

Interfaces shown in the top of the figure is compressed to the single vertex whose label is the compression of the labels in this path. The breadcrumbs for the selected node `Graphical User Interface(GUI)` can also be seen at the top of the figure.

The idea of branch compression in EulerView is to provide the users with more power to alter the focus of the display. For instance, if they are particularly interested in only a few particular categories and not in related paths, they can simply compress the relevant branches and organise them accordingly. Within the ACM classification this sort of facility might help with a user wishing to keep track of his own set of research articles if they are contained within a fairly small number of classifications. For instance, if the user has several commonly used classifications, then storing the relevant compressed branches near each other could make future classification and navigation simpler. One downside to the current implementation is that when using compression and intersection operations, the labels can become long and a means of dealing with this would be useful to be incorporated.

More generally, we imagine the branch compression is likely to be beneficial for classifications with deep trees, assisting in shortening the length of the left-to-right paths. Now, the ACM classification has only a fairly small depth (in terms of the number of vertices that can appear on any path), but when considering the possibility of displaying branches beneath projection nodes this structure deepens. As usual, user testing would be useful for examine the actual advantages and disadvantages of branch compression.

Navigational features. A projection node brings the extra navigational features of symbolic links. We also allow the

user to cycle through the nodes with exactly the same label which is effectively a symbolic link. Here we really mean the id of the node and not just the label, which we have hidden from the label in Figure 4, since it is not strictly necessary for the task of article placement or retrieval, but this information is stored and could be displayed if desired. The reason for using the id and not the label is that the same label (e.g. the label "General" is repeated in many places over the ACM CCS and does not refer to the same categorisation, but the id of those nodes does distinguish them).
We also allow the option of cycling through nodes with the same *total label set* (the set of all labels on nodes back to the root) to allow a user to navigate to all nodes representing the same category. This makes the current display option of having items only appear where they are placed reasonable since the user can find the other items in this category fairly quickly, whereas otherwise they might not be aware of other placements.

Choices of constructed views. The imported ACM in a projection view in EulerView could simply be used as a TreeView with symbolic links. The intersection view creates intersection categories of related nodes, thereby enabling the user to both classify an article with multiple classifications in one go and to enter the structure at such an intersection, but has the drawback that it is necessary to scroll vertically through a larger number of categories and if one path is used to find a relevant placement of an article then we have lost the related links. The combined view (creating both of the intersection and the projection nodes) enables the benefits of both. One reason that the use of EulerView's intersection and projection nodes may be useful is because if the categories are related in the ACM CCS then there is a reasonable chance that a user might want to classify using both of the related categories.
Now the use of projection nodes can be extended to allow the placement of articles beneath them which adds the feature of being able to multiply classifying an article by these related categories in one go. However, some of the "related to" links in the ACM CCS do not point to the most specific nodes of the classification and a user might really want to not to use the intersection of more specific categories at a lower level. One could display the whole branch corresponding to the projection node, but in general, if the "related to" relationship contained any cycles then one must impose a constraint to stop the repetition of nodes with the same id in the same branch, for instance. Adding these branches from the projection node would allow the user to navigate horizontally and to use this as a multiple classification facility rather than needing to jump via the symbolic links. The branch compression facility may then be more useful due to the longer paths being displayed.

4 Conclusion

We have indicated a method for turning a non-hierarchical model (Euler diagrams) into an EulerView which has a tree structure, via sets of paths in the transitive closure of the directed dual graph of the Euler diagram. Although EulerView allows users to construct multiple paths of their own choice to resources, we can present the user with some automatically generated EulerView displays. We have provided a sample of three such options: the projection view, the intersection view and the combined view. These can be used to transform a hierarchical structure with symbolic links into an EulerView. Linking to the file system enables the user to export the directory structure and resource placement directly to the file system; so a user can build small manageable views of their categorisation structure within which to place resources and then export this onto the larger file system if they wish.
We have investigated the potential for larger scale usage by importing the ACM CCS into EulerView enabling the user to place and store articles within the imported classification structure. EulerView has been tailored to deal with the ACM CCS, enabling multiple placement of articles together with facilities to aid navigation and user focussing. Without EulerView a user who wishes to classify an article might navigate through the ACM website structure to find the correct classification, or perhaps they might just search for an article on a similar topic appearing in the same conference series, say, and use those keywords. Utilising EulerView could aid the user in looking at their own previously classified articles to help with categorisation as well as to consider nearby classifications. The ability to store articles in the file system using the ACM classification might be useful for a research administrator of a computer science department, for instance, where they wish to investigate the spread of the recent work of the members of the department.
We have implemented features such as a zoomable universe and branch compression which should provide some assistance in narrowing the field of focus for the user as the size of imported classifications grows. The new branch compression feature is unique to EulerView, taking advantage of the fact that we are allowing multiple labels on a single category vertex in the EulerView display. Integrating the ideas of EulerView such as the use of multiple labels and branch compression with other common hierarchical visualisations, effectively making them non-hierarchical, may prove beneficial, especially if these visualisation are more suited to some particular task.
In the future we intend to perform user evaluation of the EulerView, especially with regard to their usage in larger scale systems. To date, user studies have provided very positive feedback about user's perception of the features of EulerView, but quantitative measurements have not yet been obtained. Some questions within EulerView that we would

like to address are: whether or not the alteration of the views to allow intersection categories assists or hinders users in their tasks, especially on a larger scale; and whether the extra features of branch compression and zoomable universes really do assist the user in focussing. Comparison with other tagging methods, such as free form tagging would be useful to help investigate in which situations either method has advantages.

There are many avenues of ongoing and future work such as the integration of EulerView with tools such as Flickr [3, 10], del.icio.us [2, 9] and citeulike for resource management, or even potentially with ontology management and semantic web technologies [14, 15]. The importation of other classification systems, such as the one developed in [19] which presents a unified classification system for the computing fields of computer science, software engineering and information systems, would broaden the scope of use of EulerView. The extension of the EulerView representation to explicitly incorporate other boolean operators, enabling the display of structured queries is in progress; the use of notions of stepping stones and pathways in [12] which facilitates the breaking up of queries into sub-queries will also be investigated. Another avenue is the integration with Euler diagram generation techniques for small selections of categories; this will facilitate use of diagrammatic components tied in with the EulerView thereby allowing users a choice of format for interactivity.

References

[1] ACM CCS: ACM Computing Classification System [1998 version]. www.acm.org/class/1998.

[2] del.icio.us. http://del.icio.us.

[3] Flickr: Organize your photos. http://flickr.com/photos/organize.

[4] Wikipedia: Taxonomy. http://en.wikipedia.org/wiki/Linnaean_taxonomy.

[5] Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002.

[6] Mark Bernstein. The bookmark and the compass: orientation tools for hypertext users. *SIGOIS Bull.*, 9(4):34–45, 1988.

[7] Rosario De Chiara and Andrew Fish. Eulerview: a non-hierarchical visualization component. In *VLHCC '07: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)*, pages 145–152, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Rosario De Chiara and Andrew Fish. Eulerview with projections: non hierarchical visualisation. In *6th Eurographics Italian Chapter Conference (to appear)*. The Eurographics Association, 2008.

[9] Rosario De Chiara and Andrew Fish. eul.icio.us: Euler diagrams for del.icio.us. In *12th International Conference Information Visualisation (to appear)*. IEEE Computer Society, 2008.

[10] Rosario De Chiara, Andrew Fish, and Salvatore Ruocco. Eulr: a novel resource tagging facility integrated with Flickr. In *Proceedings of the AVI 2008 Advanced Visual Interfaces Conference*, pages 326–330. ACM Press, 2008.

[11] Douglas Coupland. *Microserfs*. Harper Collins, 1995.

[12] E. Fox, F. Neves, X. Yu, R. Shan, S. Kim, and W. Fan. Exploring the computing literature with visualization and stepping stones & pathways. *Communications of the ACM*, (4):53–58, 2006.

[13] Elin K. Jacob. Classification and categorization: a difference that makes a difference. *Library Trends*, 52(3):515–540, 2004.

[14] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. Swoop: A web ontology editing browser. *Journal of Web Semantics*, 4(2):144–153, 2006.

[15] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[16] George Robertson, Kim Cameron, Mary Czerwinski, and Daniel Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–430. ACM Press, 2002.

[17] Greg Smith, Mary Czerwinski, Brian Meyers, Daniel Robbins, George Robertson, and Desney S. Tan. Facetmap: A scalable search and browse visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):797–804, 2006.

[18] Thomas W. Malone. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Inf. Syst.*, 1(1):99–112, 1983.

[19] Iris Vessey, Venkataraman Ramesh, and Robert L. Glass. A unified classification system for research in the computing disciplines. *Information & Software Technology*, 47(4):245–255, 2005.