Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface

Stuart K. Card, David Nation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304 USA
card@parc.com, dnation@acm.org

ABSTRACT
This paper proposes Degree-of-Interest trees. These trees use degree-of-interest calculations and focus+context visualization methods, together with bounding constraints, to fit within pre-established bounds. The method is an instance of an emerging "attention-reactive" user interface whose components are designed to snap together in bounded spaces.

Categories and Subject Descriptors
H.5.2 (Information Interfaces and Presentation): Graphical user interfaces. I.3.6. (Methodology and Techniques): Interaction techniques.

General Terms
Human Factors

Keywords
Degree-of-Interest Trees, DOI Trees, focus+context, information visualization, attention-reactive user interfaces, fisheye displays, hierarchical display, tree

1. INTRODUCTION

Current technology makes it feasible to bring increasingly large amounts of information to bear in computer applications. This paper explores one instance of a general strategy for constructing interfaces for high-information applications. The strategy is the Attention-reactive User Interface (AUI). Such an interface consists of two parts. One part is a method for continuous prediction of the user's instantaneous Degree-of-Interest (DOI) over items in the field of information. The other part is a dynamic visual display of the information that uses the DOI calculation to reduce the time cost of information access or to manage attention. DOI calculations could be used to allocate display resources, decide which elements to display, change representation, highlight, or take initiative in a mixed-initiative dialogue (see Figure 1).

AVI2002, Advanced Visual Interface (Trento, Italy, May 22-24, 2002): 231-245.

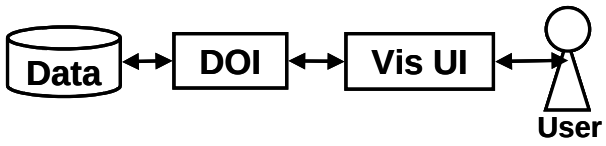


Figure 1. Attention-Reactive User Interface.

The instance of this paradigm we explore here is a method for dynamically interacting with hierarchical information in trees. Hierarchical displays are important not only because many interesting collections of information, such as organization charts or taxonomies, are hierarchical in form, but also because important collections of information, such as Websites, are *approximately* hierarchical. Whereas practical methods exist for displaying trees up to several thousand nodes, no good methods exist for displaying general graphs of this size. Hence, tree-based displays are more important than just as displays of hierarchical data.

Good visualizations of hierarchical information would (1) allow adequate space in nodes to display information, (2) allow users to understand the relationship of a node to its surrounding context, (3) allow users to find elements in the hierarchy quickly, and (4) fit into a bounded region. This last requirement is desirable in order to insure information fits on a display or that it can compose together with other display elements in an application without the need for scrolling.

For purpose of illustration, we shall use the display of organization charts as our example of hierarchical data, keeping in mind that the same techniques work for many different types of trees. Organization charts are trees (at least the ones we shall discuss first are) and have nodes that display several properties per node.

2. PREVIOUS WORK

There is a considerable literature on the use of trees for information visualization and on the layout of trees. The algorithms of tree layout from a graph drawing point of view have been summarized in Di Battista, et al [1]. Forms of trees from an information design and information visualization point of view have been surveyed in Bertin [2], Card et al [3], and Herman et al [4].

2.1 Simple Static Layouts

Much work on tree layouts assume trees will be static and concentrate on methods to layout trees that meet aesthetic criteria such as minimal line crossings, placing nodes at the same tree depth at the same level, and compactness. The simplest way to lay out trees is to lay them out *uniformly* (see Figure 2). The

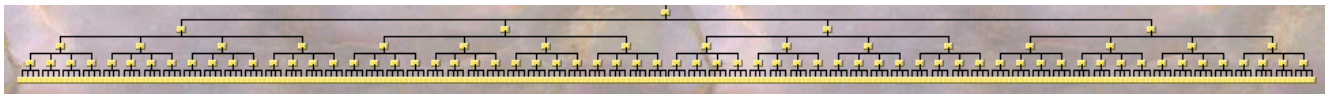


Figure 2. Small tree of 341 nodes, uniformly laid out.

number of nodes at the leaves of all the subtrees is computed and multiplied by an amount of space per node plus spacing between nodes and between subtrees. This method works for small trees, but any attempt to portray a tree of moderate size, say 1000 nodes, will start to approximate the appearance of a horizontal line, since the width increases exponentially while the height increases only linearly.

2.2 Compressed Static Layouts

More sophisticated methods of tree layouts have been developed that are spatially compact. The slide portions of deeper subtrees underneath shallow subtrees. For example, the classic *Reingold-Tilford layout* [5] and its later refinements, creates a top-down, reasonably compact tree that satisfies various aesthetic criteria, such as symmetry and the same shape of common subtrees. Tree space can also be compressed by the use of *recursive tree* layout algorithms. For example, *H-trees* lay out the first few branches as an H with later branches forming an H off those. *Ball-trees* lay-out the branches as spokes from a root with later branches as spokes from the tips of these (see Herman et al [4]).

2.3 Containment Trees

The tree layout algorithms discussed so far use node and link diagrams to represent trees. Trees have also been represented by containment, for example, as a set of nested circles. One class of particularly interesting containment trees is the *TreeMap* [6]. A TreeMap is a technique in which lower subtrees are contained within higher nodes of the tree. Space is divided, say vertically, into a number of sections equal to the number of branches at the top level. Each section is then divided horizontally according to the number of branches at the next level down in the tree. Division of the space continues vertically and horizontally until it is too small to divide. The algorithm does not allow room for the content of non-terminal nodes, but the technique can be modified so that each division has extra space for node contents. One advantage of TreeMaps is that they stay within predetermined space bounds, but there is little room for node content, especially of non-terminal nodes, and aspect ratios of the nodes vary widely, obscuring simple relationships Recent attempts to order or squarify the visualizations [7] have mitigated this effect.

2.4 Interactive Tree Layouts

The techniques discussed so far share several problems for use as components of information visualization systems. First, they do not scale sufficiently. The tree in Figure 2 has only 341 nodes, but is already nearly unreadable. Even using one of the compressed tree layouts will not adequately extend the number of nodes that can be handled. Second, much of the literature on tree layouts does not consider trees in which the nodes themselves contain significant information and require a significant amount of the layout space. Third, many of the techniques for tree layout are not bounded in space. They can therefore not be used easily as modular components of information displays. Interactive trees handle the first two of these problems by displaying only a portion of the nodes at one time. A typical interactive technique is that of the Apple Hierarchical Filing System. Each level in the tree can be expanded individually by clicking on a small triangle. Thus the user can expand portions of the tree that are to be compared on the screen, while keeping other portions of the tree compressed by eliding nodes below the compressed subtree root. The tradeoffs are that considerable manual manipulation must be performed by the user to constantly adjust views, and there is no guarantee the tree will fit, leading to more manual manipulation of scroll bars. Since the user cannot see large portions of a large tree, the user may have a difficult time navigating the tree or understanding the larger shape of the tree.

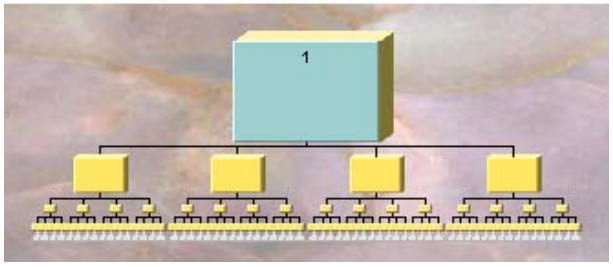


Figure 2. Display of a uniform tree of 4 levels, 4 branches each level, focus on Node 1 (number in the figure represents the node number).

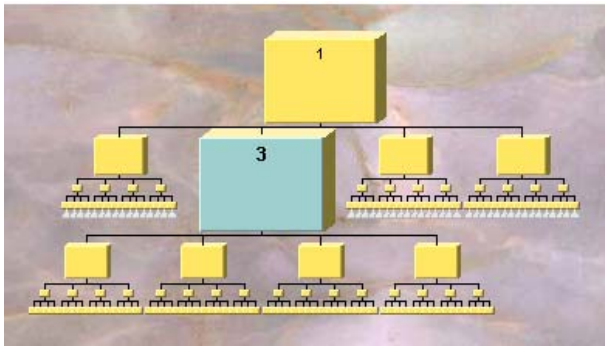


Figure 3. Same table with focus on Node 3.

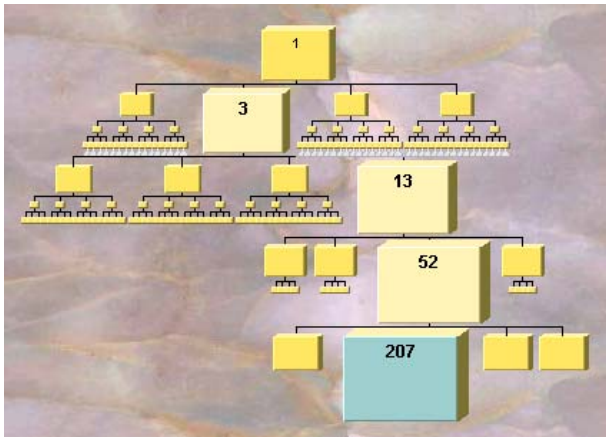


Figure 4. Same tree expanded down to a leaf node.

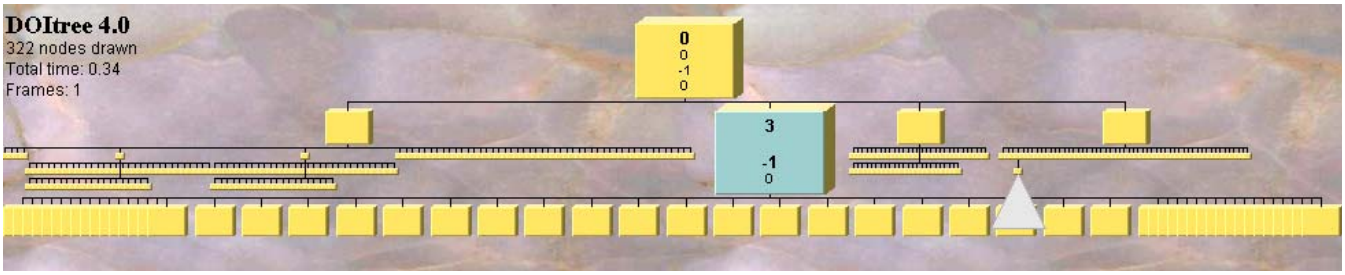


Figure 5. Tree showing overlapping of large branching factor of nodes under several nodes, when these have moderate DOI. There are 300 nodes under the triangular symbol.

2.5 Focus+Context Trees

Focus+context trees add automation for automatically choosing which portions of the tree to show at any instant. In this way, they reduce the time cost of navigating around the tree. Four main methods of expanding focus+context trees have been proposed. The first is *logical filtering* of nodes. Furnas [8] describes a class of *fisheye* techniques in which nodes are automatically displayed or elided according to the user's computed degree-of-interest (DOI) in them. The estimated

DOI of a node = Intrinsic Importance
– Distance from a focus node.

The *Intrinsic Importance* of a node is its distance from the root and the *Distance* of a node is the number of nodes that must be traversed by following parent and child links from the node of interest until reaching the subject node. Those nodes whose DOI lies below a certain threshold are not displayed. If the user indicates interest in some node, say by selecting it, this calculation is performed again and the display elides those nodes below threshold. In this way, the display of the tree follows the user's changing interest. The limitation of this technique is that there is no guarantee the displayed trees will fit in any display bounds. The technique is especially problematic when there are a large number of sibling nodes in a branch.

A second class of focus+context techniques uses *geometric distortion.* An example is the *hyperbolic tree* [9, 10]. A visual transfer function is defined that distorts space such that the area of interest is magnified at the expense of nodes of the trees some distance from this interest. Selecting a node moves it to the center (or side) of the display. Nodes further out are smaller and closer to each other. The display stays within fixed boundaries. But only a limited number of links out from the focus node can be seen. This technique is less suited for cases in which the DOI varies in discontinuous ways across the tree, although there can be a limited number of multiple foci. A variant of geometrical distortion is the *cone-tree* (Robertson et al, [11]). Cone-trees arrange the nodes in a 3-D tree. Selecting a node rotates a branch of the cone-tree, bringing related nodes into the foreground while sending other nodes into the background. This technique uses natural perspective and occlusion to achieve some of the effect of geometric distortion, but in a way that the user does not experience the geometric compression as distortion. Furnas' fisheye view technique can be combined with cone-trees, thus allowing the display of larger trees on the order of 10,000 nodes [12]. Because a cone tree is a 3-D display, some of the nodes occlude each other.

Another class of technique for expanding focus+context displays is *semantic zooming*. As the display is zoomed in and nodes are expanded past a certain threshold their content changes. Fox and Perlin [13] used this technique in their Pad system to expand a calendar, which can be seen as a sort of containment tree. Mackinlay et al [14] also used semantic zooming in calendars and other *spiral trees*. The tree is arranged in a 3D spiral. One node from the tree is expanded and its content semantically zoomed to reveal additional structure. The higher-level nodes are spiraled toward the center and made more distant from the user (using perspective to reduce their size). This tree layout has the virtue that it stays confined in fixed bounds, but the user only sees a limited subset of the nodes in the tree, essentially a view looking upward in the tree toward the root. Graham and Kennedy [15] used both semantic zooming and geometrical distortion to display biological taxonomies.

A final focus+context technique is to cluster nodes far from the point of focus. Those nodes near the point of interest are expanded into their constituent parts; those more distant are kept in an aggregate form. This technique was used in SemNet [16] and could be applied to trees.

3. DEGREE-OF-INTEREST TREE (DOI TREE) SOLUTION

Our Degree-of-Interest Trees (DOI Trees) solution combines all

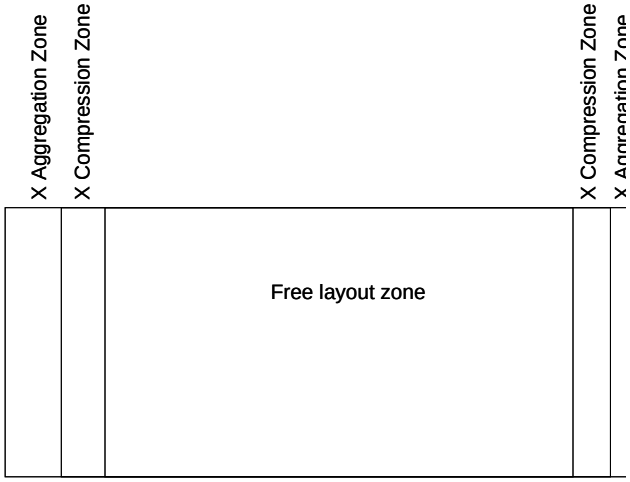


Figure 6. Compression and aggregation zones.

four focus+context techniques with a method to ensure that the tree stays within a predefined bounding box. DOI Trees combine (a) an expanded computation of users' DOI estimates with (b) logical filtering to elide nodes of low DOI, (c) geometric scaling of node size according to DOI so as to be able to hold different levels of information, (d) semantic scaling of the contents of the nodes with node size, (e) clustered representation of large unexpanded branches of the tree, and (f) animated transitions, designed to speed the user's rapid understanding of changes in the tree. New techniques are developed for many of these parts.

3.1 Degree-of-Interest Computation

The degree of interest calculation is expanded beyond that used by Furnas. Whereas Furnas's calculation assigns all siblings the same distance from the focus node and hence the same DOI value, our calculation treats the children of a parent node as ordered and assigns fractional DOI offsets to the children based on order distance from the focus node. The farther the sibling from the focus node, the more the fractional decrement in its DOI (but the decrement is always less than 1). This allows the visualization part of the program to decide which sibling nodes to compress and how to compress them. Whenever the user clicks on a tree node, that node becomes the focus node, DOI values are recomputed for each node of the tree, the tree is laid out again, and an animated transition moves to the next layout. Multiple-foci can be determined by values of the data or hits in a search.

3.2 Visualization of Tree

There are a small number of possible node sizes (we currently use three main sizes). The largest size is sufficient to display the entire full content of the node. For an organization chart, this would include a person's name, picture, organization, title, extension, room number, web page, and possible other information. A middle size node still displays enough information to identify a person including a few facts about him. A small node just displays the fact that a node exists in that position and hence shows tree structure (and a mouse target for tree expansion). In addition, nodes have multiple faces to allow the storing of additional information. A table maps DOI values into node sizes. Figure 3 shows the display of the uniform tree in the Figure 2 (with a branching factor of 4 at each node) when the focus is at the root. The larger node has automatically been selected for the focus node and its color changed. Smaller node sizes have been automatically selected by the algorithm for nodes with lower DOI. Optionally, a small "fade value" is assigned to cause nodes farther away that would be the same size to be a little smaller. This is equivalent to increasing the weighting on the DOI distance function.

In Figure 4, one of the nodes (node 3) on the next level down has been selected, changing the DOI calculation for the nodes. Now when the tree is displayed, node 1 is reduced in size, node 3 is increased, and nodes below the focus tree are increased in size, according to the computed DOIs for the nodes. The transition proceeds by a smooth animation from one state to another to keep the user oriented and unconfused.

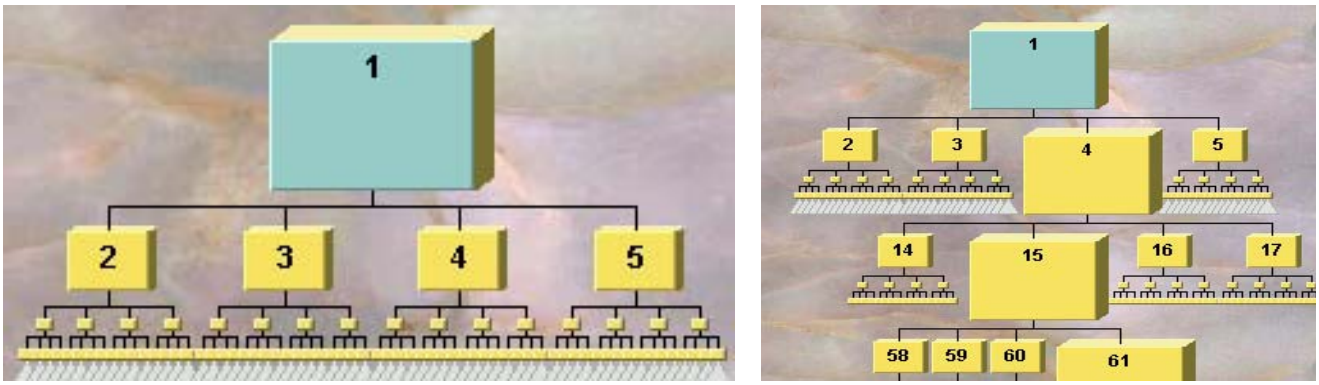
In Figure 5, one of the lowest nodes has been selected, either by selecting directly, or by selecting nearby nodes, causing the target node to get larger and be more easily selectable.

4. UTILIZATION OF THE SPACE RESOURCE

Space on the display is a resource. Making the tree stay within its resource requires methods for monitoring and making adjustments to the tree visualization. Often, to stay within its bounding box, the tree visualization must be compressed. But if the space is being under-utilized, the tree might also be profitably expanded to take advantage of the additional space. Both compression and expansion are controlled by the users' estimated DOI for each node.

4.1 Compress to Fit

The fact that the basic DOI-based algorithm very greatly reduces the pressure on the space resource sets up the condition for algorithms that enforce space boundaries to be successful. There are two cases to consider: the tree not fitting in the X direction and the tree not fitting in the Y direction.



(a) **(b)**

Figure 7. (a) Tree from Figure 1 without Expand-to-Fit turned on. (b) Same tree with Expand-to-Fit expanding the node with the most subnodes.

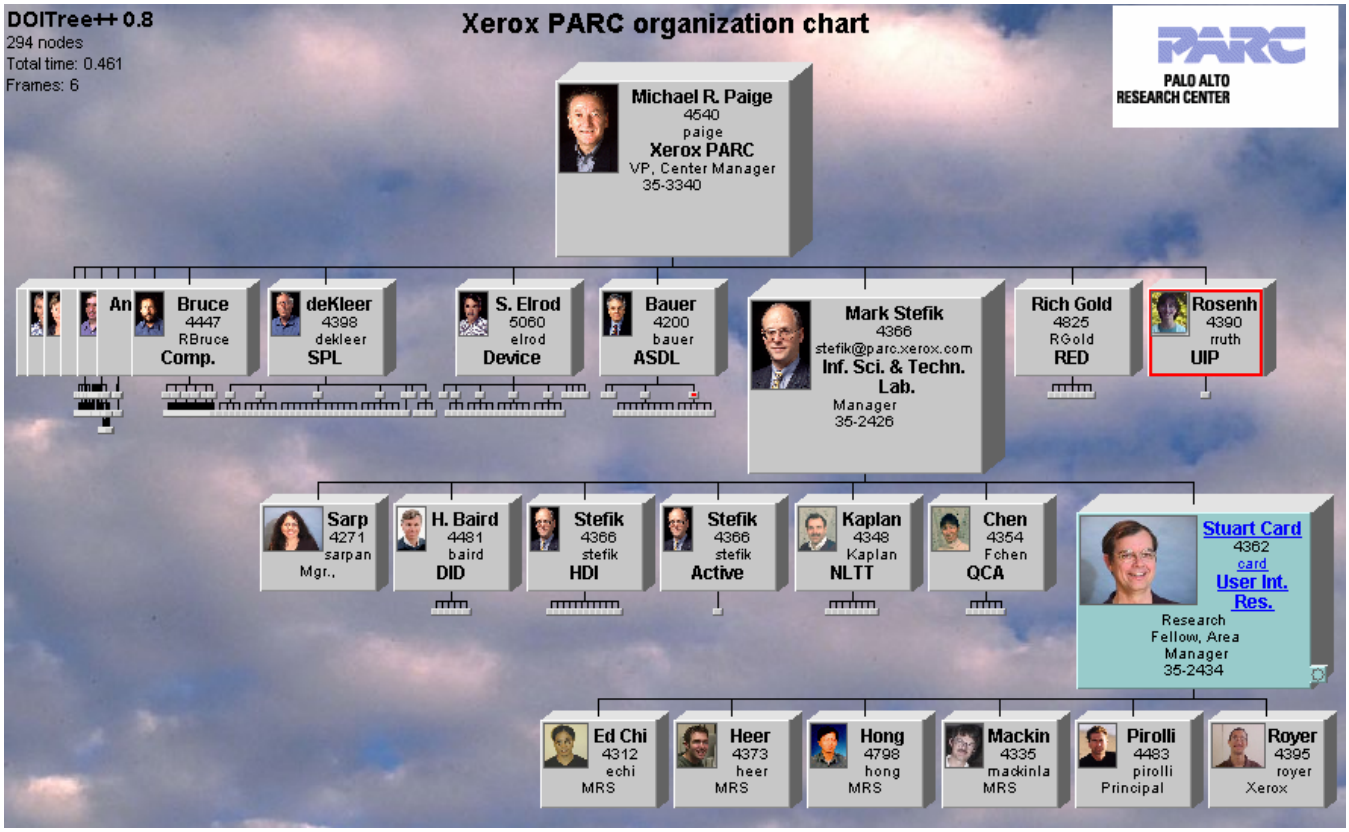


Figure 8. Organization chart with over 400 nodes accessible over WWW through Web browser.

Compression in X. The tree not fitting in the X direction is common and occurs either because of a large branching factor below one node or else because there is an accumulation of widths across several subtrees. In the latter case, the nodes below each box are laid out in the horizontal space available for each box. This pattern is visible in Figure 6. The nodes with the highest DOIs in a row have the largest node size. This node size establishes the Y height of a region in which to lay out the immediate descendents of a node. If there are too many Y descendents horizontally, they are folded into multiple rows according to a common organization chart convention (with a vertical line joining the rows, see Figure 11). If the branching factor of individual subtrees is large, then the nodes are overlapped as in Figure 6. If there are elided nodes below a threshold DOI value, then a triangular symbol proportionate to the log of the number of nodes is used.

To handle trees too wide for the bounding box, the box is divided horizontally into three regions (Figure 7): The regular free layout zone, a compression zone, and an aggregation zone. Typically, 70% of the screen is in the free layout zone, with 30% in the combined compression and aggregation zones. If necessary, the horizontal layout will be compressed (as in row 3 of Figure 6) for some of the nodes by overlapping them. As the mouse is moved over these nodes, they spring to the front, overlapping their neighbors, thereby allowing the user to peruse them. Space in the compression and aggregation zones is allocated according to the fractional DOI value of each node. By default, the value gets smaller as the node gets closer to the edge of the display. With large numbers of nodes, multiple nodes may occupy the same display location. Only one of the nodes will be displayed. If that node is selected as the focus node, it will be shifted to the free layout zone and surrounding nodes will then be visible. The use of DOI to do selective expansion and the use of folding rows greatly increases the size of tree that can be horizontally laid out. The use of compression and aggregation zones ensures that all trees can be fit within the space.

Compression in Y. It can also happen that a tree would be too deep vertically to fit within its space. Normally the tree is then scaled to fit within the Y dimension. If the scaling would result in nodes that are too small to display their contents, nodes are either elided lower in the tree or at the top, depending on the DOI and the position of the node of interest. First, if the nodes for a tree are less than a threshold DOI, then they are elided and replaced by an elision graphic, essentially representing the nodes as a cluster. Since nodes decrease intrinsic importance with distance from the root and in distance importance with distance from the focus node(s), the chain of nodes from the focus node through successive parent nodes to the root will have the same DOI value except that the fade feature will gradually make the nodes smaller as they approach the root node. Since the tree is scaled to fit within the window, this could cause all of the nodes to become very small for very deep trees. To deal with this problem a top portion of the tree is removed and replaced with an elision graphic.

4.2 Expand to Fit

Figure 12. Multi-focal result of a search. Besides showing the search hits, the tree context for the hits is also shown.

Sometimes the algorithms described so far might leave extra blank space on the display below the focus node. Therefore, an alternative version of the algorithm, switch selectable by the user, expands part of the tree into this space. The way it does this is that if there is still vertical space available on the display, the "most interesting node" is expanded.

This could (1) be the node with the highest DOI on the row, or it could (2) be the node with the largest number of descendents, or (3) it could be the node with the highest "information scent" as determined by key words typed by the user or (4) the words of greatest frequency in the nodes selected or (5) by other means. Figure 8(b) shows the tree of Figure 8(a) expanded according to the subtree with the most nodes.

4.3 Within-Node Compress to Fit—Semantic Zooming

Each node also represents an assigned space resource within which there are items to display. In our example organization chart application, these fields include attributes such as post name, post reported to, name, title, office extension, email, picture file URL, and home page URL. Just as we used compress-to-fit techniques for the layout, we can also use semantic zooming compress-to-fit techniques within the node:

1. *Data deletion.* Smaller nodes only display some of the data items.

2. *Word abbreviation.* Words and phrases are abbreviated if there is not room on the line where they are displayed. For example, Vice President becomes V.P. The system uses a text file of abbreviations plus some heuristics to generate abbreviations. Figure 9 shows the DOI Tree of an organization chart with the green node selected. The effect of data deletion and word abbreviations on the middle-sized nodes can be seen.

3. *Node rotation.* The normal view of nodes shows them as 3D boxes. The 3D property is meant to suggest that the boxes have alternative faces. Stroking a box (by dragging the mouse horizontally over a box) makes the box appear to rotate such that another side of the box faces forward. This allows more data items to be associated with a node that are quickly accessible. Figure 10 shows the rotation of the nodes. Figure 10(a) shows a frame in mid-rotation; Figure 10(b) shows the completed rotation. In the figure, the picture has been expanded to fill the whole node side,

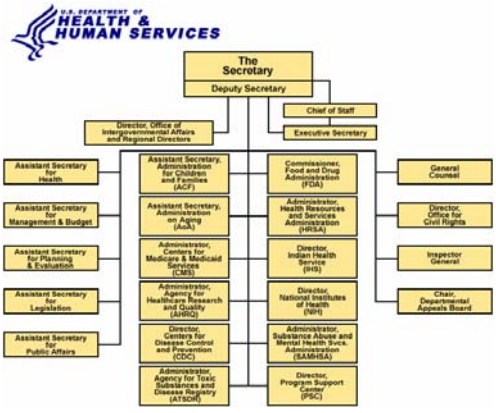


Figure 10. Organization chart showing folding convention. This tree layout is more difficult for producing understandable animated transitions for the user.

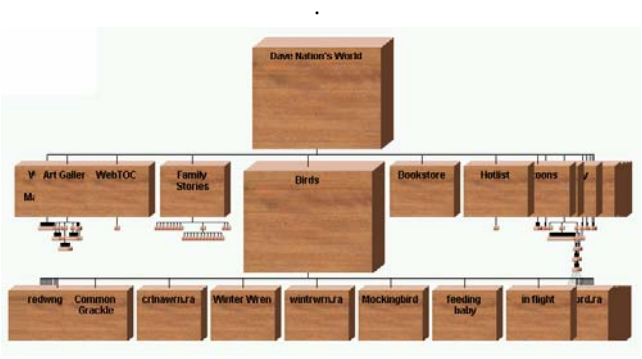


Figure 11. Visualization of Dave Nation's World website. Nodes expand and contract as user browses or searches website

to serve as a better cue—another form of semantic zooming. There are optimally three faces to a node—front, left, and right (more would be confusing).

4.4 Tree Transitions

User orientation in the tree is preserved by making the views of the tree animate into each other. The animation time is set at a desirable level, usually in the range of (0.5~1.0 sec) (see [17]). The average draw time per frame for recent frames is used to set the number of animation frames that can be drawn in this time. The in-between frames show a linear change in each node's size and position. It is important that the choreography of animated transitions be understandable by the user. In fact, the necessity for simple-to-understand transitions was found to limit tree layouts. For example, Figure 11 employs a folding convention often used in organization charts. Implementation of similar layouts convinced us that it is difficult to make tree transition animations with such layouts that are understandable to the user, except in the case where the stacked rows are of the smallest size nodes (e.g., as in Figure 6). Rapid-to-understand transitions are very important, because focus+context displays use system dynamics to substitute for being able to show the entire tree. The user must have the experience that there is a single tree that is being stretched and pulled as it is being explored.

5. SEARCH AND MULTIFOCAL DOI TREES

DOI calculations can be done based on computations other than the user's point of attention used above. An example is a user searching over a collection. This often results in multiple initial focal points from which to start the DOI calculation. The interest in a given node may be composed of DOI generated from multiple sources. In Figure 13, the user has searched for the name Stefik in the organization chart. The result reveals that Stefik appears three times and reports to himself twice. The chart shows these nodes, but it also shows them in context to the rest of the organization chart. The visualization makes it easy to find answers to questions like "to whom does Stefik report?" or "who is the lone person reporting to Stefik in one of his capacities?" or "who are all the research fellows and how are they distributed across laboratories?"

6. USES OF DOI TREES

DOI Trees have many uses. Here we list just a few:

Information Browser. Items in the tree could be linked to arbitrary URL pages or to programs, such as an email program. Hence, the tree could act as a browser across pages of WWW data (Figure 12). For some applications, such as the organization chart, the tree as a browser operates more quickly than a conventional WWW browser page. This is because a group of the pages can be on the screen together in their relationship.

Organization Chart. This is the application we have used as an example. In addition to displaying the organization chart and its use in finding people in the organization, the URL links on the nodes of the tree also serve as gateways to supporting data (Figure 9). This chart has over 400 nodes, is accessible over the Web, and combines all the information contained in ten separate organization charts (each of which fills a page). We also maintain a larger organization chart several times its size. By searching for a name or by browsing the chart, the details of the individual organizations are revealed. Furthermore, the chart serves as a gateway into the organizational home pages of the different organizations (accessed by clicking the appropriate link within the node) or personal home pages. It also could be used to access email to any of the individuals whose email is given on the chart by simply clicking the link.

Web site visualization. Another use is for views of Web sites, which have been coerced into tree form. Thumbnail miniatures of pages could be displayed in the nodes. Full size displays of the pages could be displayed beside the browser.

Web site statistics. The DOI of individual pages in the web site could be set to a function of the number of hits that page has received in the last month or week or hour or other time period. Thus, site sponsors could watch the activity of their web sites.

Databases. Databases that are expressible by trees could be displayed and searched. For example, the 7000-node taxonomic database used for competitive tests at CHI is shown in Figure 14. By following the higher-level groupings, the user has found the node "Ebola Virus."

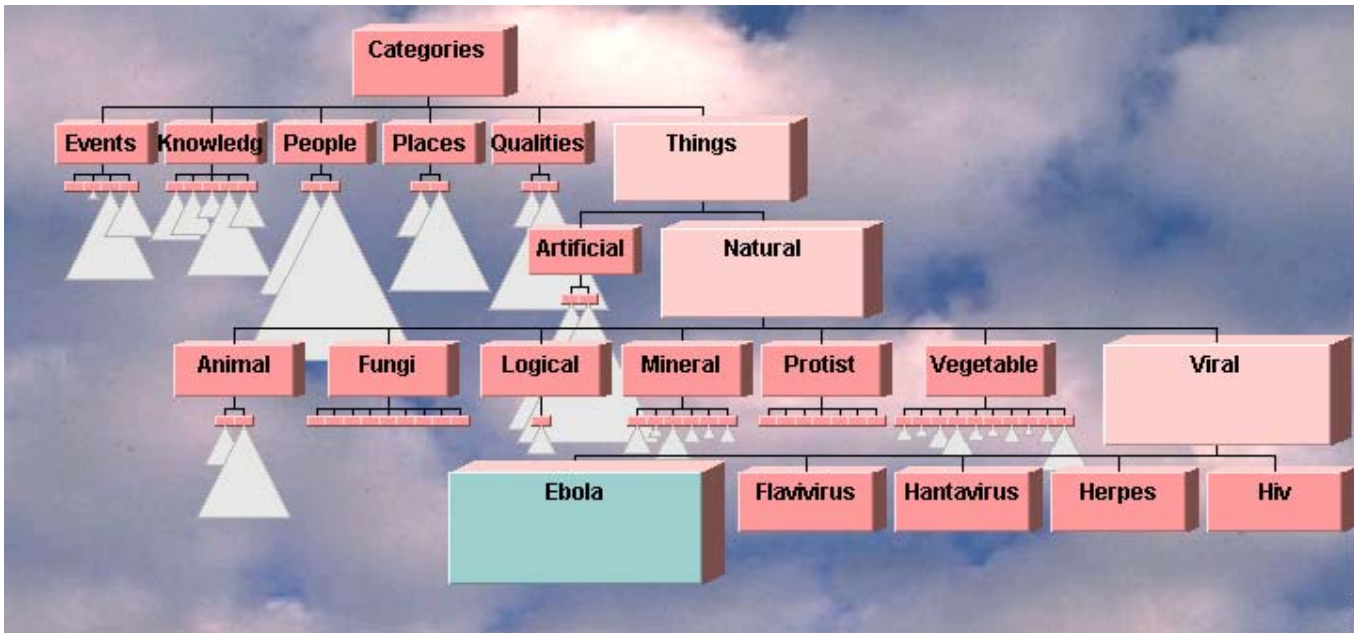


Figure 13. 7000-node taxonomic database. User has found the item Ebola.

Multilinked databases. Many databases that are not trees can also be displayed as DOI Trees. An example of such a database is given in Figure 15. The database is coerced into a tree. Additional, non-tree links are revealed as blue lines (as in the figure) when the mouse is moved over them. Because of the mapping into the tree, some nodes may be duplicated in the structure (these are colored pink in the figure). By using these techniques, complex structures that would be difficult to plot as generalized graphs are plotted as trees, but the other linkages can still be investigated.

Email streams visualization. Email streams could be represented as trees. The DOI for these streams could be generated based on the content similarity and tree closeness.

6.1 Data Source and Authoring

The data for DOI Trees can be derived from a database. They can also be read from tab-delimited files. Users can thus prepare and edit trees for DOI Tree display by using normal spreadsheets without any programming. The present embodiment of DOI Trees enables users to place arbitrary bitmaps as backgrounds to the tree and to the nodes. This allows the display of these trees to be readily adapted to presentation requirements of an organization.

7. DISCUSSION

The evaluation of dynamic visualizations such as DOI Trees is a subtle undertaking. Philosophically, we do not believe it is sufficient to do a simple system A vs. system B human factors evaluation on DOI Trees, because the results may depend on the task, the contents of the nodes, and perceptual properties of the visualization design, all of which need to be teased out. Our previous studies on the hyperbolic tree required lengthy studies using eye-movements and scales of semantic ambiguity (measured as information scent) in order to tease out issues of visualization design [18, 19] and visual attention [20]. We plan similar studies for DOI Trees, but they are beyond the scope of this paper. What we can say is that the current prototype is in use for organization charts at PARC, in use for other databases in the government, and under consideration for a web analytics product component. Users seem to be able to use DOI Trees easily. Several government agencies in the health and statistical services area have indicated interest. When we contacted the Xerox licensing office for licensing discussions on this system, we were amazed that they had already discovered it on the internal corporate Web and were already using it for their own applications. Thus, we believe it will not be hard to use the system for practical applications.

DOI Trees use predictions of the user's dynamically changing interest to change the display: If the user indicates interest by selecting one nodes, then the system predicts the user's relative interest in the other nodes through the DOI calculation. If the user indicates interest by a search for some term, then the system predicts interest in the other nodes through the DOI calculations from the nodes that hit. We are implementing other ways for the user to indicate a base interest, such as frequency of access. The purpose of having separate DOI and visual rendering calculations is so that new methods of indicating interest may be devised without having to redo the visual algorithms.

Nodes of most interest are filtered for visualization, geometrically enlarged, semantically zoomed, and shown in tree context of relevantly selected other nodes. Nodes predicted to be distant from the user's interest are shrunk, aggregated, or elided. Nodes in the display become the access portal to related information on the web or applications like email. DOI trees force the display to be contained within a constrained space, but they also choose extra nodes to display in order to fill that display.

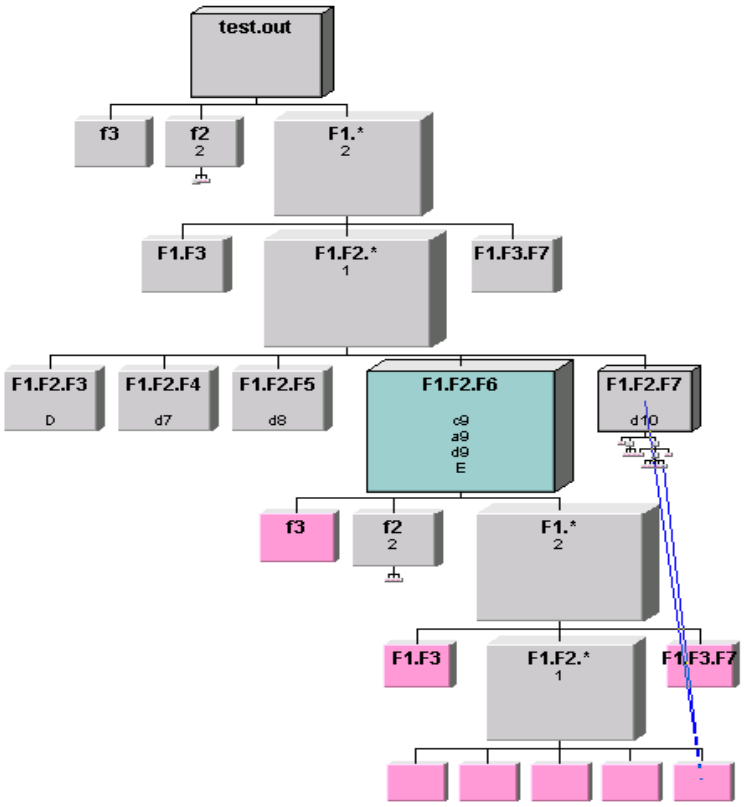


Figure 14. Tree derived from database with multiple links per node. One link type is used to form the tree. Others are visible when the mouse is placed atop them. (Node faces have had proprietary data removed).

These design features are in the service of higher-level goals. The principle goal is to reduce the average cost-structure of information to the user of a large information set. The DOI calculation and its subsequent use in visualization attempt to reduce the cost structure of the task using the information. For example, in Figure 12 the cost in time for finding who Stefik reports to or who reports to him in his many capacities is low. This is partly done by making user access to many of the other reporting relationships in the tree a little slower. That is to say, the DOI calculation is used to *bias* the interface in a way that accelerates likely actions. This differential cost of accessing information is what we mean by a cost structure and the paradigm of an attention-reactive user interface seeks to dynamically change these biases according to where the user's attention is (or should be—this paradigm can also be used to direct the user's attention to areas the system thinks are important). Although there is an access bias, notice the fact that DOI Trees still maintain an overview of the entire field of information, maintaining context and at least a minimum access to all pieces of information.

The second goal is to increase modularity between the DOI and the visual components by limiting the information exchanged between the DOI calculation and the visualization to a narrow interface. It should be possible to change either the DOI or visual component more or less independently Hence DOI Trees should make a good system building block.

The third goal is spatial modularity. By making the DOI Tree stay within its space bounds (which could be dynamically increased or decreased), it is easy to compose this display with other displays (for example, a panel showing detailed information about the current focus node). Thus, it would be easy to use this focus+context display as the overview part of a larger overview + detail display. Whereas a technique like TreeMaps stays within a bounded space, they have a more difficult time showing the contents of interest for nodes in large trees.

The last two goals make DOI Trees a modular system component to use in the construction of attention-reactive user interfaces for systems involving access to or sensemaking of large collections of information. The DOI Trees presented are particularly simple instantiations of the attention-reactive user interface idea. More complex dynamic calculations are possible that handle other sources of context or that take over automatically handling other overhead tasks for the user as the user's attention progresses.

8. ACKNOWLEDGMENTS

Jeff Heer from Xerox PARC and Debbie Roberts from NSA contributed code to the algorithms.

9. REFERENCES

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs. Upper Saddle River, NJ: Prentice Hall, 1999.

[2] J. Bertin, Semiology of Graphics: Diagrams, Networks, Maps. Madison, WI: University of Wisconsin Press, 1967/1983.
[3] S. K. Card, J. D. Mackinlay, and B. Shneiderman, Readings in Information Visualization: Using Vision to Think. San Francisco, California: Morgan-Kaufmann, 1999.
[4] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," IEEE Transactions on Visualization and Computer Graphics, vol. 6, pp. 24-43, 2000.
[5] E. M. Reingold and J. S. Tilford, "Tidier drawings of trees.," IEEE Transactions of Software Engineering, vol. SE-7, pp. 21-28, 1981.
[6] B. Johnson and B. Shneiderman, "Space-filling approach to the visualization of hierarchical information structures.," in Proceedings of IEEE Visualization '91, 1991, pp. 284–291.
[7] B. Shneiderman and M. Wattenberg, "Ordered TreeMap Layouts," presented at IEEE Symposium on Information Visualization, 2001.
[8] G. W. Furnas, "The FISHEYE view: a new look at structured files," in Readings in Information Visualization: Using Vision to Think, S. K. Card, J. D. Mackinlay, and B. Shneiderman, Eds. San Francisco: Morgan Kaufmann Publishers, Inc., 1981, pp. 312-330.
[9] J. Lamping and R. Rao, "Laying out and Visualizing Large Trees Using a Hyperbolic Space," presented at Proceedings of UIST'94, ACM Symposium on User Interface Software and Technology, 1994.
[10] T. Munzner and P. Burchard, "Visualizing the structure of the World Wide Web in 3D hyperbolic space," presented at Proceedings of VRML '95, 1995.
[11] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone trees: Animated 3D visualizations of Hierarchical Information," presented at Proceedings of CHI'91, ACM Conference on Human Factors in Computing Systems, New York, 1991.
[12] S. K. Card, J. D. Mackinlay, and B. Shneiderman, Readings in Information Visualization: Using Vision to Think. San Francisco, California: Morgan-Kaufmann, 1999.
[13] K. Perlin and D. Fox, "Pad: An Alternative Approach to the Computer Interface," presented at Proceedings of SIGGRAPH'93, ACM Conference on Computer Graphics, 1993.
[14] J. D. Mackinlay, G. G. Robertson, and R. DeLine, "Developing Calendar Visualizers for the Information Visualizer," presented at Proceedings of UIST'94, ACM Symposium on User Interface Software and Technology, Marina del Rey, Ca, 1994.
[15] M. Graham and J. Kennedy, "Combining linking & focusing techniques for a multiple hierarchy visualisation," presented at 5th International conference on Information Visualisation, London.
[16] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas, "SemNet: Three-dimensional representations of large knowledge bases," in Cognitive Science and Its Applications for Human-Computer Interaction, R. Guindon, Ed. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1988, pp. 201–233.
[17] S. K. Card, T. P. Moran, and A. Newell, "The Model Human Processor: An engineering model of human performance," in Handbook of Perception and Human Performance, K. K. L. Boff and J. Thomas, Eds. New York, New York: John Wiley and Sons, 1986, pp. Chapter 45, 1- 35.
[18] P. Pirolli, S. K. Card, and M. M. Van Der Wege, "Visual information foraging in a focus+context visualization," presented at CHI 2001, Seattle, 2001.
[19] P. Pirolli, S. K. Card, and M. M. Van Der Wege, "Effects of information scent and information density on the hyperbolic tree browser," in review.
[20] P. Pirolli, S. K. Card, and M. M. Van Der Wege, "The effect of information scent on searching information visualizations of large tree structures," presented at AVI '2000, Palermo, Italy, 2000.