Animated Visualization of Multiple Intersecting Hierarchies

George Robertson, Kim Cameron, Mary Czerwinski, & Daniel Robbins

Corresponding Author:

George Robertson
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
Tel: 1-425-703-1527
Fax: 1-425-936-7329
E-mail: *ggr@microsoft.com*

Running Title: Polyarchy Visualization

To appear in Journal of Information Visualization, Vol. 1, No. 1, March 2002

ABSTRACT

We describe a new information structure composed of multiple intersecting hierarchies, which we call a *Polyarchy*. Visualizing polyarchies enables use of novel views for discovery of relationships which are very difficult using existing hierarchy visualization tools. This paper will describe the visualization design and system architecture challenges as well as our current solutions. *Visual Pivot* is a novel web-based polyarchy visualization technique, supported by a "polyarchy server" implemented with a *Mid-Tier Cache* architecture. A series of five user studies guided iterative design of Visual Pivot. Finally, we discuss the effectiveness of animation in Visual Pivot.

Keywords

Information Visualization, 3D, Animation, Hierarchy, Polyarchy, User Studies

INTRODUCTION

People working in enterprises face a common problem in understanding the relationships among data from multiple databases. For example, consider the databases a typical large corporation maintains about people and resources. There are often many databases describing employees, such as those for recruiting, organization charts, managing organizational headcount, benefits, mail, access to networks and data resources, building security, telephone services, and capital assets. Each database contains information about people, with some information loosely replicated in several databases. A *metadirectory* provides a common interface to all of the databases, without replacing them. Since each database is optimized for a particular function, there is no need or desire to combine them all into a single database. In addition, a metadirectory provides synchronization between the individual databases for data that is replicated, so that changes to any data will be propagated to all databases which contain it. Metadirectory technology guarantees convergence of information across connected systems. The result is more accurate information at reduced administrative cost.

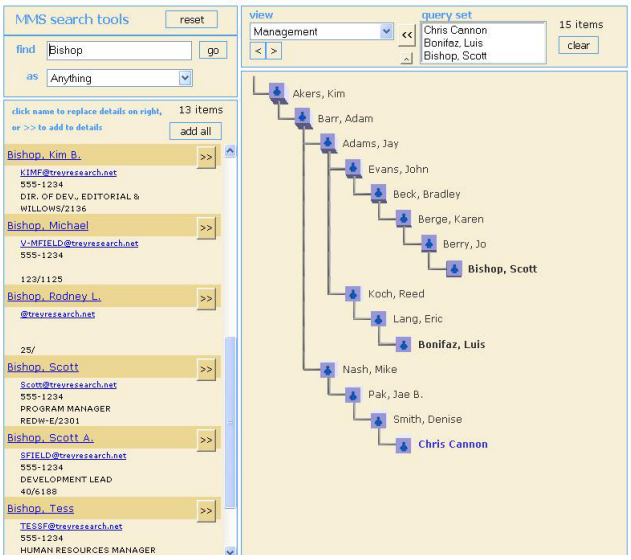

Figure 1. Polyarchy Visualization showing relationship of three people in the management hierarchy.

When a metadirectory combines information from separate databases for display, a serious visualization problem occurs: the user must make sense out of data across multiple intersecting hierarchies (i.e., hierarchies that share at least one node), which we call a *Polyarchy*. How do we show the information to the user in a way that makes sense? How can the user determine which hierarchies a particular entity (e.g., person, group, business unit, or location) belongs to? How do we display these different hierarchies in a way that helps the user understand the relationship between entities within a hierarchy as well as between the hierarchies in the context of selected entities? A *Polyarchy Visualization* must address these questions.

Solving these problems enables use of individual databases beyond their original scope. In our example, a typical employee is able to easily view relationships among people along a number of dimensions. For example, if I receive an announcement of a meeting with several people I do not know, how do I quickly find out who they are and what they do? Traditional solutions require the use of several tools and many interactions to answer such a question, if it is even possible. Such solutions require cognitive integration of the results over time and across multiple dimensions. Polyarchy visualization enables a user to get the same result with a few simple interactions. Figure 1 shows a visualization of the management relationship between three people. The user obtained this result simply by selecting the three people and the desired view.

While Figure 1 only shows one hierarchy view (Management), users need to see other hierarchy views, and we need a way to transition between them. We use a new visualization technique called *Visual Pivot,* with two hierarchies simultaneously displayed around a designated entity called the *pivot point* (the first entity in a query set). This is a visual analog to a database pivot, where a view of one dimension is replaced with a view of another dimension. An animation pivots from one hierarchy view to another, giving the user a chance to see the relationship between the hierarchies in the context of the selected pivot point, if necessary. Perhaps more importantly, the pivot animation helps the user maintain context during complex transitions.

While we have described these polyarchy problems in terms of databases about people and resources, the problems occur with any collection of databases that have entities appearing in more than one of the databases. For example, a person may want to explore several consumer oriented databases simultaneously. Products tend to be organized in multiple hierarchies; resellers may have one database organized by cost, another by manufacturer, and another by product type. Each manufacturer may have its own hierarchy in which those products appear. Bringing information from multiple resellers and manufacturers together into one polyarchy would make the shopper's experience much easier. In the following sections, we will describe how visual pivot works, briefly describe five user studies that have guided iterative design of visual pivot, discuss the effectiveness of animation in visual pivot, and discuss implementation issues for both the visualization client and polyarchy server.

RELATED WORK

Hierarchies are one of the most commonly used information structures. A typical computer user interacts with hierarchies many times each day. Over the last twenty years there has been much research on effective display and interaction with hierarchies: the Smalltalk File Browser in 1979 [19]; Fisheye Views in 1986 [9]; SemNet in 1986 [8]; Cone Trees in 1991 [18]; TreeMaps in 1991 [13]; Hyperbolic Browser in 1994 [14]; FSViz in 1995 [5]; H3 in 1997 [16]; Disk Trees in 1998 [7]; and many others. In spite of all that research, we still have not solved some basic problems, particularly with scalability (loss of context for large hierarchies) and difficulty maintaining focus on multiple selections. The polyarchy visualization technique described below specifically addresses scalability and multiple foci issues for individual hierarchies.

As the industry begins to address enterprise-wide problems (for example with metadirectories), we see uses of multiple hierarchies that current approaches do not handle. Some work has been done on multiple hierarchies. The Time Tube [7] examines a single hierarchy changing over time and highlights changes. While careful analysis of highlighted changes in Time Tube does reveal interesting patterns, it is not clear that a casual user could easily see those patterns. In addition, the user is forced to integrate these changes cognitively across time, putting a strain on short-term memory resources. A taxonomy visualization [11] examines similar hierarchies and highlights differences between them. In this visualization, all hierarchies are shown side by side and lines are drawn between common nodes in the hierarchies. This also reveals interesting patterns. However, the authors of that visualization found that the technique did not scale well and abandoned the approach. The polyarchy visualization technique does scale to very large hierarchies, as will be discussed. MultiTrees [10] are multiple hierarchies with shared subtrees.

But polyarchies are multiple *intersecting* hierarchies, sharing at least one node rather than sharing subtrees. Hence, MultiTrees are a subset of polyarchies. The added complexity requires a new approach as described in this paper.

POLYARCHY VISUALIZATION TECHNIQUE: VISUAL PIVOT

The goals of polyarchy visualization are: (1) to show the user how selected entities relate to each other in a hierarchy; (2) to make it easy for a user to move from one hierarchy view to another without losing context; and (3) to help the user see how various hierarchies relate to each other in the context of selected entities. We envision this latter goal to be of increasing importance as metadirectories become more common. We focus on selected entities and their paths to the root of each hierarchy, instead of general overviews of extremely large hierarchies. Traditional tree views show expanded subtrees (e.g., all siblings of any given node), making it difficult to scale to large hierarchies without losing global context, and making it difficult to focus on multiple nodes because of the distance between them. In contrast, by showing paths (rather than expanded subtrees), we are able to scale up to enormous hierarchies while supporting multiple foci.

The user interface, as shown in Figure 1, has four parts. In the *upper left*, the user specifies a search. For this particular metadirectory, the search attributes are Anything, First Name, Last Name, Full Name, Title, and Email Alias. In the example, we are searching for Scott Bishop, so we search for "Bishop" as "Anything". The *lower left* panel displays a list of the search results, including the display of key attribute values (e.g., email address, phone number, title, and location). If the user clicks on one of the search results, it will replace what is displayed to the right, so that the new query set only has that one person. There is also an "add to" button to the right of each person, which will cause the person to be added to the query set displayed to the right.

A key aspect of this system is that search results are always displayed in the context of the current hierarchy view specified in the *upper right* section, which has a menu of hierarchy views and the query set. In this example, "Management" is the selected hierarchy view and there are three people in the query set. Just to the left of the query set is a button for removing items from the query set. This panel also contains back and forward buttons for controlling the history of changes to the selected hierarchy view and query set. The *lower right* section displays the current polyarchy visualization, showing the query set in the selected hierarchy view, typically showing the paths from the selected entities up to the root.

In the lower right display, if the user clicks on an entity, it is added to the query set as the new pivot point. Hovering on an entity will bring up a tool tip, which provides additional information about the node. A right click will bring up a context menu that allows the user to pivot to any of the other hierarchy views in which the entity participates. Both the tool tip and context menu involve a query to the polyarchy server, so server performance is critical. This will be discussed further in the implementation section below.

While the example in Figure 1 only shows three people in the query set, the user can add as many people to the query set as desired. We have tested this with dozens of foci, and because we primarily show paths and hence the foci remain close to each other, it is easy to interpret the result. For example, if the user pivots to the list of email distribution lists for these three people, it is a simple operation to pivot to a view of all of the people on one of those

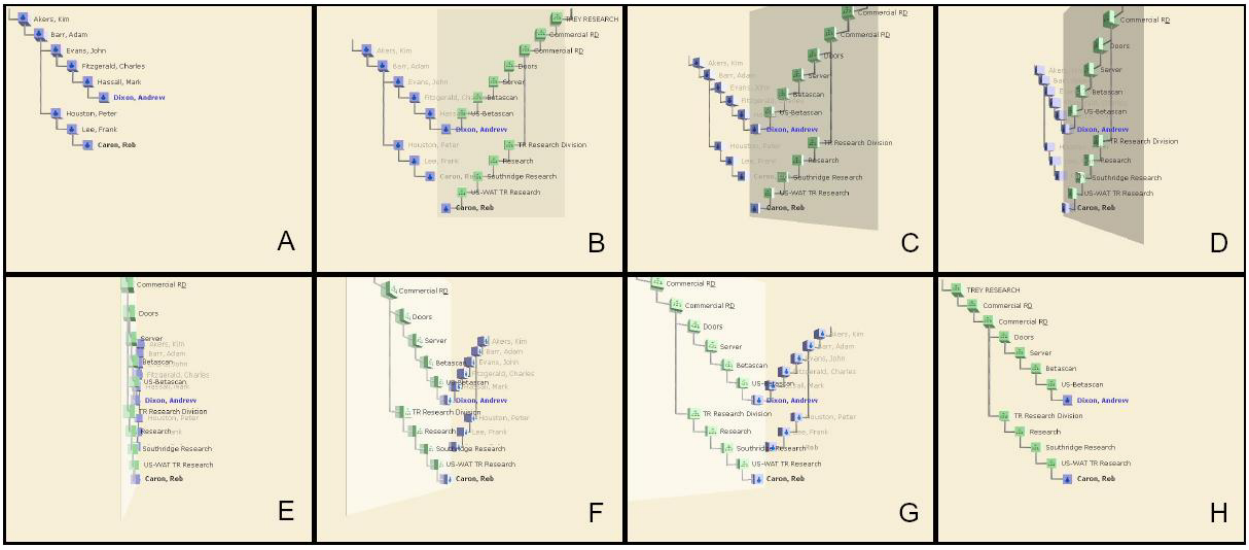

Figure 2. Visual Pivot rotation animation; a sequence of frames starting with the management view and ending with the business unit view around pivot point "Andrew Dixon".

distribution lists. This is a way to discover how many different business units in the company participate in a distribution list. In a few quick steps, the user can move between several different hierarchies and select dozens of foci.

Another key aspect of the system is how animation is used to change the display when any change to the view or query set is made. If we are simply replacing everything, an animation is used to show the old display moving off to the right as the new display moves in from the left. If we are adding an entity to the current view, part of the display is moved to make room for what is being added. If we are removing an entity from the current view, part of the display is moved to eliminate what is being removed. Each of these animations was empirically optimized to be approximately one second.

While pivoting from one hierarchy view to another, one of several visual pivot animations is used to emphasize relationships between hierarchies in the context of the pivot point and help the user maintain context. Figure 2 shows a sequence of frames using one pivot animation style (rotation about a vertical axis). Figure 3 shows the second frame of that animation (Figure 2B enlarged), with both views simultaneously visible. Nodes in the management hierarchy are all people (blue with a person icon); while most nodes in the business unit hierarchy are

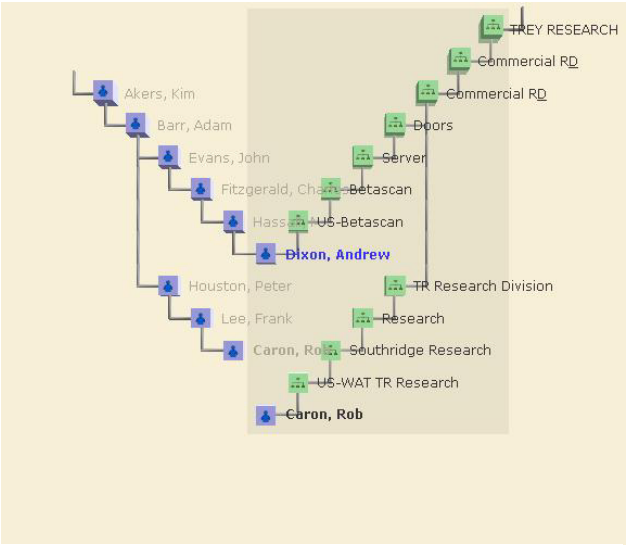

Figure 3. First stage of Visual Pivot from management to business unit hierarchy around pivot point "Andrew Dixon" (Fig. 2B).

organization units (green with an orgchart icon). The text labels of members of the query set are bold (i.e., the multiple foci of interest), with the pivot point (first entity in the query set) bold and blue. The new view is initially rotated 180 degrees around the pivot point. That is, the new view is to the right of the old view, reversed and connected to the old view at the pivot point (Andrew Dixon in this example). Additional highlighting of the pivot point may be desirable, and we are considering adding that to a future version.

Next, the animation rotates both views 180 degrees about the vertical axis through the pivot point so that the old one is replaced on the left with the new one (Figure 2C through 2G). Figure 4 shows a schematic of this rotation animation. Finally, the new view is moved to a preferred viewing position and scale, as the old view disappears (Figure 2H). The pivot animation is done in about one second; long enough that the user is able to see that the old view is morphing to a new view with the same pivot point, but short enough that the user does not feel as though waiting for the system. While static views show the relationship between entities *within* a hierarchy, it is the visual pivot that shows the relationship *between* hierarchy views in the context of the pivot point and maintains context for the user. The animation may be stopped while in progress (with the space bar) so that the user can more clearly see the relationship between views, if necessary. Note that information about the relationship between hierarchies is limited to parts of the hierarchies in the context of the selected entities, rather than global relationships. We are concerned with hierarchies containing tens or hundreds of thousands of entities, so showing global relationships is impractical. In addition, real users of polyarchies have made it clear to us that the understanding of relationships between hierarchies that they seek is in the context of selected nodes.

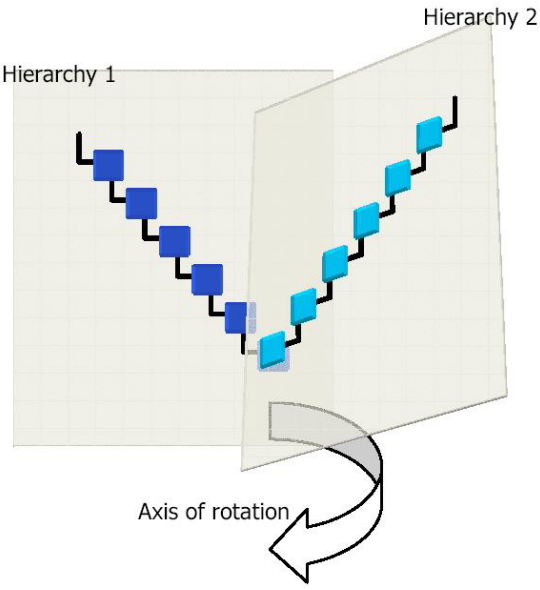

Figure 4. Schematic of visual pivot vertical rotation animation about a pivot point.

Multiple Hierarchy Visualization

The methods described above are appropriate for showing transitions between two views. To see relationships between three or more views, a user must still sequentially switch between multiple views. To simplify the case where the user needs to see two or more views simultaneously, a *Stack Linked* animation style was added. Figure 5 is an example of this style, showing three views simultaneously. Whenever the user switches to a new view, the previous views are moved back and to the right, so they recede into the

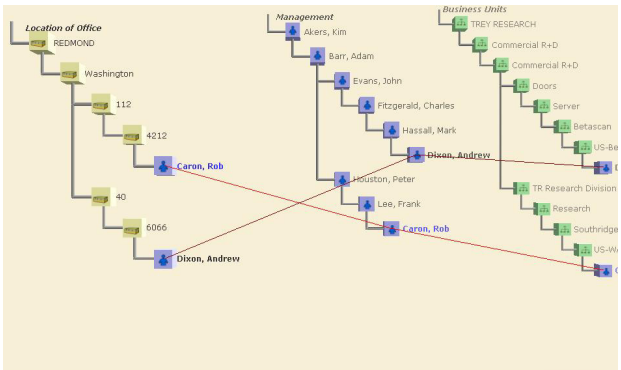

Figure 5. Stack Linked style showing three hierarchy views.

background, fading away and taking less space because of the perspective view. Optional connecting lines show where selected entities appear in each view. Once the user gets beyond three or four views, a horizontal scrollbar is needed to see the older views. While the nodes are drawn fully in perspective, the text size reduction is limited so that text never gets too small to read. When the user replaces the query set (rather than adding to it or pivoting), the old views are eliminated.

We have experimented with eight visual pivot animation styles (three rotations about vertical axes, two rotations about horizontal axes, sliding, and two kinds of stack) and a wide range of animation times. The next section discusses five user studies, including studies to select the best animation style and speed.

USER STUDIES

We conducted five user studies to guide the iterative design of polyarchy visualizations. In each study, participants were shown one or two tutorial tasks to help them learn how to use the system. Then they did two or more tasks that were timed. Each task involved retrieving information about people in an organization. Previous interviews with end users of the databases joined by the polyarchy visualization gave us a set of ecologically valid tasks for our study. Each task started with one person and built up to three people in a view. There were 12 questions for each task. Typical questions were:

1. Who is Evan's manager?
2. How many people have the same title as Evan?
3. Do Evan and Richard work in the same building?
4. What business unit do Evan, Ed, and Denise have in common?
5. Who comes from the largest set of peers, Evan, Richard, or Betty?

Study #1

The first study used a visual pivot concept mockup, implemented with Macromedia Shockwave animations. Twelve participants were given a variety of tasks to determine if the concept made sense. The sequence of selections and animations was predetermined based on static information. The goal was to gather subjective data and usability issues rather than performance data. Participants reported an average satisfaction rating of 4.84 on a 7 point Likert scale (with the highest number as most satisfied on Likert scales in each study). A number of usability issues were observed and used to guide development of a prototype. Users were observed to have no difficulty understanding the concept of visual pivot.

Study #2

The remaining studies were performed using the working prototype with live data about people working at our company. The second study compared a 2D unanimated version with a 3D animated version of visual pivot at three animation speeds (0.8, 2.0, and 4.0 sec). The pivot animation style was the rotation about a vertical axis described earlier. Nine target end users were given a series of four tasks to complete, focused on simple, hypothetical Human Resource issues. Each task was performed using one of the four conditions (2D, 3D fast, 3D medium, 3D slow). Task performance completion times were measured and subjective satisfaction data was gathered.

There was no significant effect of the different visualization techniques on task completion time. Planned comparisons showed that the 3D fast and 3D medium conditions were not significantly different than the 2D condition, but there was a trend toward the 3D slow condition having a longer task completion time ($F(1,8) = 2.15$, $p=.21$). There was a significant effect of prior experience with 3D game playing on performance. A split-half comparison, dividing participants into two groups according to their mean task completion time, revealed that the "faster" group was significantly faster than the "slower" group ($t(7) = 3.55$, $p < 0.01$). The two groups differed widely in the average amount of time that they played 3D video games weekly, with the faster group playing 9.60 hours a week on average and the slower group playing only 1.25 hours a week.

A common opinion expressed by participants was that the slow 3D condition was too slow. Six of the nine subjects thought that the slow animation was more distracting than informative. However, there were many positive comments about the 3D animation, and the average satisfaction rating was 4.42 on a 5 point Likert scale (unfortunately this was a different scale than those used on the other studies). Several participants mentioned that rotation about a vertical axis made the text difficult to read during part of the animation, because of occlusion. Text from two views overlapped at times, and was the same weight and color; hence word recognition was difficult.

Study #3

The third study was a survey of four animation styles at three faster animation speeds (0.8 sec; 1.8 sec; and 3.0 sec). There was also an instant case, which had no animation but otherwise had the same 3D appearance. Tested animation speeds were faster in response to previous observations.

The first style was the rotation about a vertical axis used previously. Other styles attempted to fix observed problems, particularly with occlusion. The second style used rotation about a vertical axis, but only the new view moved with no text visible and a transparent panel behind it to help with visual grouping. The third style was a sliding style (see Figure 6): the new view appeared to the right of the old view and slightly lower, then moved to the

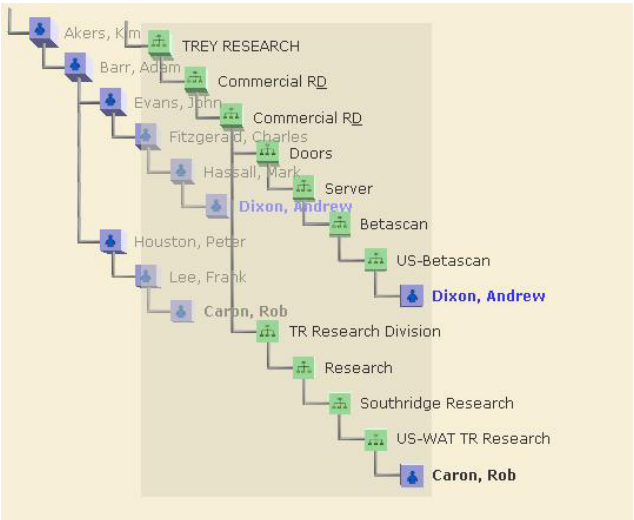

Figure 6. Sliding animation during transition from management to business unit hierarchy.

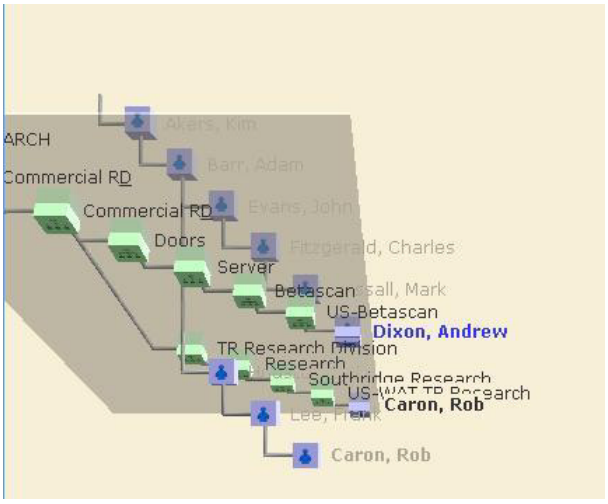

Figure 7. Horizontal rotation during transition from management to business unit hierarchy.

left and then up, while the old view's text was grayed out to help with occlusion. Graying out text in one view was intended to eliminate the word recognition problem. The fourth style (see Figure 7) pivoted about a horizontal axis through the pivot point, while the old view's text was grayed out.

Eight target participants were given one task and asked to try each of the 13 style/speed combinations. They were asked to vote for their three favorite combinations, using 10 total points divided any way they wanted. That is, if a participant liked one combination a lot more than the other two, that person could give the first an 8 and the other two one point each. Results obtained from the study revealed that the preferred style was rotation about a horizontal axis, with the sliding style placing second. The preferred speed was 0.8 seconds (by a large margin), followed by 1.8 seconds, then instant. No one voted for the slow speed.

Study #4

The fourth study was performed using three faster animation speeds (0.5 sec; 1.0 sec; and 2.0 sec), and focused on comparing the two highest rated animation styles: sliding and rotation about a horizontal axis. The animation speeds chosen for the study were faster in response to the results of study #3. Fourteen participants performed seven complex, multi-part tasks, one for each experimental condition (instant, sliding fast, sliding medium, sliding slow, horizontal rotation fast, horizontal rotation medium, and horizontal rotation slow). The order of experimental condition was determined with a Latin square design. Performance times were recorded for all subtasks that occurred within each complex task.

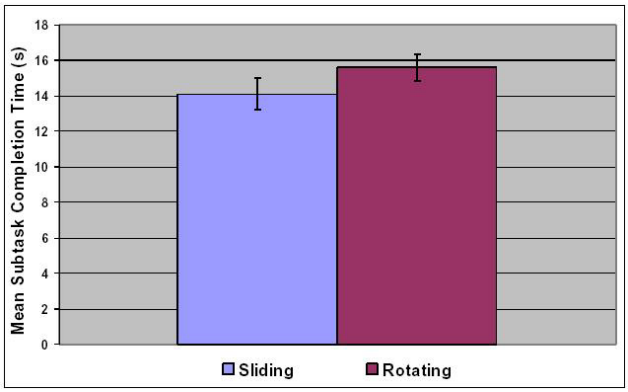

Figure 8. Study 4: Mean subtask completion time for sliding versus horizontal rotation.

An analysis of the completion time for each subtask showed a reliable advantage for the sliding animation over rotation about a horizontal axis (see Figure 8). There was also a significant effect of practice for both animations, with a reliable advantage for the sliding animation (see Figure 9). There was no reliable difference between instant, fast, and medium speeds. However, the slow speed caused a significant disadvantage (see Figure 10).

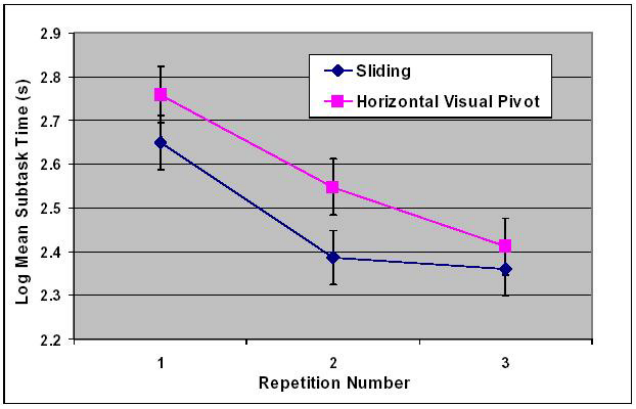

Figure 9. Study 4: Learning effects of sliding versus horizontal rotation.

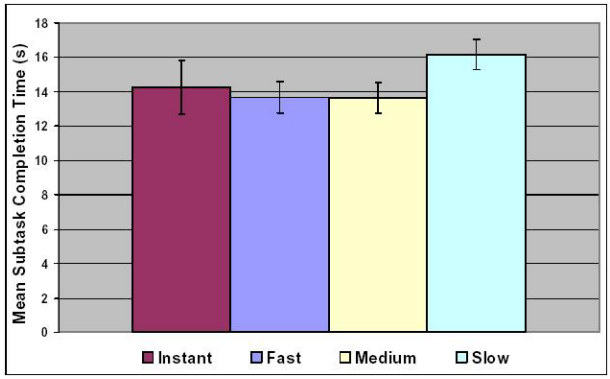

Figure 10. Study 4: Mean subtask completion time versus animation speeds.

Satisfaction results indicated a strong preference for the fast animation speed. When comparing mean satisfaction scores collected at the end of each task there was a significant difference between animation styles ($t(8) = 5.443$; $p < 0.001$), with sliding receiving higher satisfaction scores than rotation about a horizontal axis. The sliding animation averaged 5.9 on a 7 point Likert scale, while horizontal rotation averaged 5.6. When asked which visualization technique they preferred at the end of the experiment, participants were split evenly (seven preferred sliding and seven preferred horizontal rotation).

While study results suggest that the fast sliding style is sufficient, it is quite easy to support multiple animation styles and speeds, thus giving users a choice.

Study #5

The fifth study focused on comparing the sliding animation style with the stack style (Figure 5 without connecting lines shown), which was added to provide a way for users to see multiple views simultaneously. Since some task questions could be answered using information from previous views, we hypothesized that performance would be faster with the stack style. Six participants took part in this study.

The performance results indicate no reliable difference between sliding and stack styles. Since the stack style can become complex, it is possible that its potential advantage was overcome by complexity. Satisfaction results show that the sliding style was preferred by every participant. Sliding averaged 5.96 on a 7 point Likert scale, while stack averaged 4.67. Their comments clearly indicate that stack style complexity limited satisfaction.

These 5 studies enabled iterative refinement of the pivot style and animation speed to users' satisfaction. The sliding animation at a speed of about one second is now the default pivot and animation parameters. We believe the fact that the sliding pivot fared best over the studies supports the hypothesis that animation helps users maintain context, since that animation had the minimum occlusion of all the styles tested. Additional speeds and styles are also supported since alternative animations also received high satisfaction scores.

Comparison Study

In a pilot test of study 5, we attempted to compare the polyarchy visualization to the Outlook Address Book, an existing technique for accessing information about people. During the pilot study, we found that many of the tasks simply could not be done using Address Book, and others were so onerous that the participants refused to complete them after an initial attempt. Headtrax is another existing tool (using a traditional hierarchical tree control user interface) that allows users to access information about people. It also cannot be used for many of the tasks, and is also onerous for others. Because of these problems, and the initial reaction from the pilot participants, it became

difficult to do a direct comparison study of polyarchy visualization and existing tools. To address this problem, we have performed a keystroke-level analysis [6] of 15 tasks for each of the three systems (polyarchy visualization, Address Book, and Headtrax).

Figure 11 shows the results of the keystroke-level analysis, which indicate clear benefits to using the polyarchy visualization with sliding animated pivots. The 15 tasks were the subtasks from one of our earlier study tasks, and are shown in increasing order of number of foci (single focus for the first five, two foci for the next five, and three foci for the final five). We counted the number of behaviors needed to complete the task in each of the three systems. By behaviors we specifically mean mouse clicks, mouse movements, visual scans, reading, scrolling, counting, item storage in short-term memory (STM) or comparisons with items in STM. Outlook Address Book could not be used to complete four of the tasks without paper (external memory), and Headtrax could not be used to perform eight of the tasks. Step counts greater than zero show how many extra steps are required in Outlook (blue) or Headtrax (red). If the application could not be used to complete the task, a hashed bar is shown at a fixed maximum height. One step count is shown that is less than zero, and this indicates the one task for which the polyarchy visualization required 3 extra steps over Headtrax, because the task information was already displayed in the Headtrax user interface from the previous task results. In all other cases, many more steps are required in Address Book and Headtrax to complete the tasks, or the tasks could not be completed. Tasks 1, 6, and 11 are simple steps to add an additional person to the focus. The three foci tasks (12 through 15) are considerably harder than two foci tasks for Address Book and Headtrax because they cannot easily be done with short term memory (either long term or external memory is needed for those tasks), resulting in a greater risk of forgetting, interruption, and error. In contrast, three foci tasks are only slightly harder than two foci tasks for the polyarchy visualization.

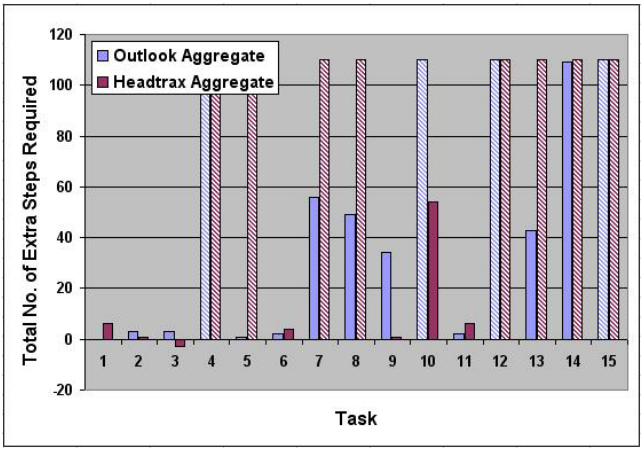

Figure 11. Extra behavioral steps required for each of 15 tasks. The crosshatched bars are cases where the task could not be done (4 tasks for Outlook and 8 for Headtrax).

DISCUSSION: ANIMATION EFFECTIVENESS

We believe that animation is a critical component in Visual Pivot, helping the user maintain context and potentially see relationships between different views around a pivot point. Our user studies confirm that there is a strong user preference for animation and no task performance penalty as long as the animation is keep to one second or less. A review of relevant literature on animation suggests why we obtained those results.

Robertson, Card, and Mackinlay [18] reported early results on the use of interactive animation in an information visualization. In the Cone Tree, animation was used during transitions to newly selected nodes. The transformations were complex, possibly involving several simultaneous rotations of sub-cones (some clockwise and some counter-

clockwise), bringing the selected node and the path to the root to the front of the display. It was observed that without animation users took extra time to re-assimilate the information structure with which they were interacting. With a one second animation, the perceptual system tracked the transformation, and users spent much less time in re-assimilation. The speculation was that the visual perceptual tracking system was at work, allowing the user to pre-attentively perceive that it was the same structure undergoing transformation. While no user studies were done to verify that speculation, a simple demonstration comparing the transformation with and without animation was very compelling. We believe the same mechanism is at work in the Visual Pivot animations.

Along similar lines, Bartram [1] argues that current visualization techniques try to offload some cognitive processing to the perceptual senses via the use of mentally economical graphical codes, such as shape, size, color, and position. It is argued that the cognitive economy comes from the rapid and efficient processing of these codes by the pre-attentive visual system, as opposed to requiring attentive scrutiny. Bartram argues that the use of these codes is limited by human perceptual capacity and visual acuity (e.g., the foveated area is limited to around 2 degrees in size or less). In order to support synthesis and awareness across large information displays, Bartram argues for the use of animation to improve information bandwidth. Based on a series of studies, Bartram concludes that one strong argument for using animation comes from its ability to evoke an emergent property of grouping when multiple, similar motions occur across a dense data display. This allows the user to immediately recognize associated elements which may be widely dispersed. In Visual Pivot, the nodes in each view are grouped with the aid of animation in just the way Bartram describes. Bartram also demonstrates that combining movements of separate elements can elicit an immediate understanding as to how they are related. These findings support the claim that Visual Pivot helps the user see the relationships between views and maintain context.

Bederson and Boltman [2] added a one second transition animation to a hierarchical family tree organization and compared this to a no animation version of the identical tree on navigation, memory, tree reconstruction and satisfaction measures. The additional one second animation did not slow down participant response times, and significantly reduced the number of errors and time taken in the reconstruction of the tree from memory. In addition, 12 out of 20 participants thought that the animated version of the tree provided both an improved understanding of relationships and the ability to better see an overall view. The authors believe that the animation directly aided the learning of spatial relationships through spatial memory, which helped in the navigation and reconstruction tasks. A preliminary design guideline offered by the authors is that if a task requires users to know something about objects' spatial positioning, and the viewpoint is changed, then animating that change in viewpoint appears to benefit users in the task. They also made the claim that their guideline would likely generalize from 2D to 1D and 3D visualizations.

Bertamini and Proffitt [3] describe two issues surrounding motion organization. The first issue is how the type of motion involved might interact with the establishment of a frame of reference for that motion. The second issue involves how the structure of the organization (not the motion information per se) is used to constrain the motion organization. The authors compare three types of motion (translation, divergence, and rotation) in testing whether they can each support the perception of common motion. They found that translational motions in space can serve

as a moving frame of reference for the perception of relative motions, but that rotation cannot without a fixed pivot point. The authors define structural invariants (SI's) as constant geometrical properties of an event that are revealed when the configuration moves. Rotating around a fixed pivot point is an example of an SI. The authors discuss how it is crucial that the pivot point be clearly demarcated in rotations around a point. They further make the point that the SI's cannot be extracted from any single static frame of a display; their constraints are revealed by motion. Their status is structural, not dynamic. The Visual Pivot rotation animations are clear examples of this phenomenon.

In addition, Bertamini and Proffitt performed experiments that suggest that translation is the easiest motion to recognize (simpler than divergence and rotation). They conclude by stating translation and divergence are approximately equal in their abilities to support moving frames of reference, but rotation needs the presence of SI's. From an ecological point of view, translation and divergence signify the displacement of an object while rotation does not. SI information is important for determining what motion we perceive, indicating that motion and spatial structure are not really separable visually. These results confirm what we saw in Visual Pivot user studies. The sliding animations are translations, hence were easiest to recognize and were preferred. Most rotation animations had easily recognized pivot points, hence were structurally invariant and also worked well. In a few cases, a pivot point was not visible from one view to the next, thus losing the SI property. That may partially account for rotating being preferred less often than sliding pivots.

Finally, Proffitt and Kaiser [17] argue that motion information is a "minimally sufficient" condition for perceiving numerous environmental properties, and that its use eliminates the need to use most conventions typically employed in static displays, such as pictographic cues. They also argue that animated displays can elicit more accurate perceptions than their static counterparts in certain contexts. The authors conclude that users analyze animation into relative (rotation) and common (translation) motion components. Designers of animation displays need to recognize that moving object configurations interact with their displacement perception and secondly, rotation and translation motions have different perceptual significance. Rotations define 3D form, while translations define observer-relative displacements. This analysis suggests that the Visual Pivot sliding animation may be perceived as observer-relative while the rotating animations may be perceived as defining form, which is possibly less important for the tasks we used in our studies.

Morrison and Tversky [15] found no evidence that animation has any benefit over static graphics for learning. However, the animations they studied were neither interactive nor under user control. Also, they acknowledge that it is possible that animation may be effective for maintaining user attention and comprehension. We have demonstrated that there is a benefit to using Visual Pivot; the animations are interactive (brief and initiated by user action), can be controlled by the user, and are intended to maintain user attention and comprehension, hence maintain context.

In polyarchy visualization, animation is used to maintain context in several ways. In addition to the context around the pivot point during a visual pivot, any addition to (or deletion from) an existing view is animated so that the appropriate parts of the structure move (e.g., to make room for added material). Animations also help maintain

context by conveying the nature of changes (i.e., one animation for replacing the selection set; one for pivoting to another hierarchy view; one for adding material; and one for deleting material). Without animation, the user would just see a new display in each case and tend to lose track of what changes are taking place.

Overall, the evidence supporting the usefulness of animation is mounting, though more studies should be done. Our own future studies will examine how different levels of complexity (number of items in a complex query) influence optimal animation style and timing.

IMPLEMENTATION ISSUES

Building effective polyarchy visualizations requires solving a number of difficult problems. The goal is to enable a user to easily explore data from several existing databases, and understand the relationships between multiple intersecting hierarchies and between entities in those hierarchies.

Defining a Polyarchy

The first step in providing a polyarchy visualization is identifying which hierarchies should be exposed to the user. Some hierarchies are explicit, others are implicit. The metadirectory designer identifies which databases to include and which hierarchies to expose.

Many of these hierarchies are straightforward. For example, management, business units, and location are all simple hierarchies. However, the designer may want more complex views of those hierarchies. For example, in our experimental polyarchy, "Related People" is a relationship between people that uses the management hierarchy to show a person's direct reports, their manager, their manager's direct reports, and the path up to the root. This represents the set of people the selected person is most likely to interact with. The designer can define this view by specifying how many levels up and down to display, and whether or not to include siblings. For "Related People", the designer specifies "up=*, down=1, siblings=true". Contrast this with the "Management" view shown in Figure 1, where the designer specifies "up=*, down=0, siblings=false". Choosing these views is not something that can easily be done algorithmically, as it requires thought about the most useful relationships to expose. While this may be a difficult task for the designer, it need only be done once for a particular metadirectory. The visualization must be designed to allow the designer to specify such views, and the user to easily select them.

The second step is to identify search attributes, to help the user select entities for display. Although analysis can suggest candidate search attributes, the metadirectory designer must identify those that are most appropriate.

Once a polyarchy is defined, three fundamental problems must be addressed to provide an effective visualization: scale, flexibility, and responsiveness. Scale is a key problem in two ways. First, the amount of data being explored is enormous, ranging from tens of thousands to millions of nodes. Second, the number of users accessing this data is enormous; recall that our goal is to make enterprise data useful for most of the employees in the enterprise. Flexibility is a key problem because users want to explore each hierarchy fully, with no performance penalty for any hierarchy. Responsiveness is critical because of the exploratory nature of user interactions, requiring much iterative refinement of queries. Responsiveness is important in any interactive system, but takes on a special character in this

system because of the flexibility and scale problems. For example, to scale in number of users, the data passed between client and server must be kept to a minimum, and processing required by the server must be minimized. This leads to delaying the request for data until it is actually needed. Context menu and tool tip contents are not requested until the user clicks the mouse button or dwells on a node. To remain responsive requires the data for these basic operations must be obtained extremely quickly.

In this section, we will discuss a variety of issues that were encountered while solving the key problems of scale, flexibility, and responsiveness.

Web Service Architecture

Because of scale issues in database size and number of users, it was clear from the beginning that a web service architecture with a web-based client was required. A web service architecture supports many different client interfaces and back-end servers, and hence is more flexible than traditional client/server architectures. In this paper, we describe one particular client and server. The details of the server will be discussed later. Here we describe the nature of the client and communication between client and server.

The client is a web page that supports the user interactions described earlier (search for people, selection of people, managing the query set, choosing the hierarchy view, context menus, and tool tips) as well as the visual pivot sliding and rotation animations. The client communicates with a Polyarchy Query Server (PQS) using a Polyarchy Query Language (PQL) expressed as XML (eXtensible Markup Language) [12] request and reply forms. These XML forms are packaged as SOAP (Simple Object Access Protocol) [4] remote procedure calls, which allows handling of faults in a uniform way.

The client is written in a combination of JavaScript, Dynamic HTML, and XSLT (eXtensible Stylesheet Language: Transformations) templates. The search results displayed in the lower left panel of Figure 1 are constructed by sending search requests to the PQS server in a PQL query, then transforming the PQL reply with an XSLT template. A 2D version of the lower right panel (Figure 1) is constructed in a similar way using XSLT. However, XSLT and Dynamic HTML cannot easily be used for 2D animations and cannot be used for 3D animations. So, there is a 3D version of the lower right panel implemented by walking the Document Object Model of the PQL reply and constructing a 3D scene using WildTangent's Web Driver, a web-based 3D rendering engine (see http://www.wildtangent.com).

Data-Driven Service

Polyarchy visualizations must be flexible to make exploration by the user easy, and also to make it easy for the metadirectory designer to describe the databases to include, the search attributes, the hierarchy views to expose to the user, and how information is displayed to the user. To address the responsiveness problem, this information must be in a form that allows the server to optimize caches and indexes for most responsive replies to client queries. These issues are addressed in our web service architecture by providing an XML data description which drives both client and server.

Elements of Interest. The data description includes information about elements (or attributes) of interest, identifying: whether the element is a search attribute; the display name of the search attribute in various languages; and how the element is indexed on the server. The server can index an element as "Selecting", "Distinguishing", or "Classifying". A "Selecting" element is an attribute with few entities having a particular value, but the values are not unique; hence this is a good search attribute. A "Distinguishing" element has substantially unique values (e.g., email address). A "Classifying" element is a good attribute for a filter (e.g., job title). The assignment of index type for each element can be made by the metadirectory designer. It can also be derived algorithmically with a statistical analysis that evaluates how widely distributed an attribute is over the set of objects, and the uniqueness of the values of that attribute. In our sample polyarchy, there are twenty elements of interest, six of which were identified as search attributes.

Data Dimensions and Views. The data description includes information about data dimensions (hierarchies) and their views. The data dimension description drives the server construction of polyarchies. The view description drives the client in the construction of PQL queries and interpretation of PQL replies. It also drives client construction of the view pull-down menu and context menus. There may be many views for each dimension.

The view specifies how to construct the query and display the reply. The reply can be a list, a hierarchy, or a hierarchy constrained to a subset of the database. The search is further specified by the number of requested levels up, the number of levels down, and whether or not to return the node's siblings. In our sample polyarchy, there are five dimensions and ten views.

Display Formatting. Display of nodes in the polyarchy visualization and tool tip content are also data-driven. For each element type, there is a description of how to display the node in the search results panel and the query results panel, including what icon to use, and how to display tool tips. Both display lines and tool tip content lines are designed to allow a mixture of text and any number of attribute values.

Making tool tip content data-driven is important for two reasons. From a flexibility point of view, it is required to ensure that these techniques can be used for any collection of databases. From scale and responsiveness points of view, it is important to delay retrieval of tool tip content until needed, to minimize the amount of data transmitted by the server. This can only be done if the client has a description of tool tip content in order to generate those delayed queries.

Source Databases. Finally, the data description describes the source databases. This description specifies a source XML document and anchoring element, or unique global identifier defined for each entity in the database. Each element or attribute is defined in the database description. In our sample polyarchy, there are approximately 126,000 entities, 59,000 of which are people.

PQL: Polyarchy Query Language

The Polyarchy Query Language (PQL) is similar to SQL, but has been designed for queries and responses needed for polyarchy visualization, focusing on the problems of scale, flexibility, and responsiveness. In particular, PQL

was designed to make it qualitatively easier than SQL to perform hierarchical queries and vastly reduces network traffic for such queries.

PQL is a rich query language, allowing enormous flexibility for exploration. The polyarchy visualization is designed to be intuitive and hide much of the complexity. For example, it is well known that having users specify Boolean queries is problematic. We hide that complexity by giving the user a way to manage a query set of people, implicitly performing union searches on that set. We also hide complexity by defining useful views that specify the various parameters of a search. This has worked in all cases we have encountered so far, but only because PQL is rich enough to support a wide range of queries.

PQS: Polyarchy Query Server

Our architecture includes a server component for several reasons. The main reason is that there are several different ways in which hierarchical relationships emerge from data elements held in existing stores such as relational databases. Currently, we have formalized three. The first is what we call "data-based upnodes", such as when one entity's manager attribute references another entity's employee ID attribute. This is a recursive relationship. The second is through "explicit arcs", which are actually packed into data elements, as is the case in business unit information in our example database. The third is through inferences derived from ordered lists of attributes present in each tupple of a database record, an approach used in building database and spreadsheet pivot tables. We expect that more will emerge.

Navigating these hierarchies requires completely different types of queries, schema knowledge and logic. The essence of polyarchy is to abstract these differences such that a client need not understand the specific mechanisms used in accessing a given hierarchy, but rather has "normalized" operations effective on hierarchies of any kind. This normalization, both in describing and operating on hierarchies, vastly simplifies conceptualizing and building polyarchy clients such as the one described here. Abstracting polyarchy as a set of operations from the particular mechanism used in a given hierarchy is the role of the polyarchy server. In addition, there is a need for a language to express polyarchy queries at an abstract level. This language must be interpreted by a polyarchy server.

Further, conventional database technology has none of the optimizations needed for efficient hierarchical query. These must currently be hand-crafted for each specific instance of each type of hierarchy. So the second role of the polyarchy server is to supply the optimizations necessary for polyarchy clients to achieve user satisfaction. For example, determining the intersection of two large recursive hierarchies from the top of two trees requires a separate query for each element in the data set when using conventional SQL, but only a single query in PQL. These are capabilities which one day will be present in general purpose database servers but which today are absent.

The Polyarchy Query Server (PQS) has gone through three implementations. The first version used Microsoft Metadirectory Services (MMS), a commercial product available to enterprises to solve the metadirectory database problems described in the introduction. The first version of the client used MMS as its server, using the existing MMS query language instead of PQL. That implementation had problems in flexibility and responsiveness, since MMS was not optimized as a query server.

Our second version of PQS was an experimental version of MMS, written on top of SQL-Server instead of MMS's own data store. This allowed mixture of MMS queries with SQL queries, solving some of our flexibility problems. However, there were still problems with responsiveness because many round trips to the server are required to evaluate polyarchy queries with a relational database server. It became clear that current relational database servers cannot provide the combination of responsiveness and flexibility required for this visualization.

To maximize flexibility, scalability, and responsiveness, a third version of PQS was implemented. By developing PQL, PQS, and the client visualization together, we have created a flexible environment in which to develop the model for polyarchy data representation, while eliminating assumptions through which a server tends to limit a client (latency, performance, and scalability).

PQS was designed as a *Mid-Tier Cache,* loosely coupled to the underlying data stores through replication of data. A mid-tier cache takes snapshots of all information sources and holds them in RAM, optimized for the accesses specified by the data description shared with the client, then responds to PQL queries from clients. The cache can be replicated on any number of server machines. This makes it possible to easily scale to an arbitrary number of users.

Designing PQS as a cache benefits from the rapidly decreasing cost of RAM and multi-processor hardware, allowing a reconsideration of traditional strategies for database optimization. For example, it is not hard (or expensive) to build a server with enough RAM that most (if not all) of the cached data can be locked in memory, avoiding paging performance issues altogether. This is particularly important given the flexible nature of polyarchy queries, which tend to access diverse parts of the database frequently, with a need for minimal latency.

There are three advantages to a mid-tier cache architecture. First, the cache can be optimized based on the data description described earlier. Second, the cache can be replicated, so the architecture scales to an arbitrary number of users. Third, it can work with many different underlying stores simultaneously. As long as the store can produce an XML snapshot, the mid-tier cache can incorporate that snapshot.

Unresolved Issues

There are two interesting unresolved issues with the visualization: hierarchy ordering and text rendering.

Hierarchy Ordering. When the server returns a hierarchy, sibling nodes at a particular level of the hierarchy are in an undetermined order. The client currently sorts siblings alphabetically so that the user sees a consistent result. This works for a single hierarchy. However, if multiple entities are selected and the user pivots to another hierarchy view, the order of two selections may be reversed. Without animation, this can be quite confusing. Animation helps, since the user sees the reversal take place. The problem might be solved by modifying the sort for the new view to reflect the order of the old view. Unfortunately, the new view will not be in alphabetical order. A user could be just as confused by un-alphabetized results. There appears to be no way to solve both problems simultaneously; hence this seems to be a fundamental problem with polyarchy visualization.

Text Rendering. The polyarchy display is a node-link diagram with a text label for each node. There are several ways to display text in the 3D scene so that it moves appropriately during 3D animation. Text rendered in 3D would provide ideal depth cues (e.g., change in size depending on distance from the viewer). However current 3D rendering techniques for text produce poor quality and readability because current texture filtering is inadequate. Some readability problems can be addressed by bill-boarding the text in 3D (showing it always in the plane of the screen but rendered at the proper depth). This still requires texture filtering, resulting in poor quality. An alternative is to render text in an overlay plane, adjusting its position on each frame so that it appears to be in the correct place. This produces the highest quality and readability, but incorrect depth cues. While this approach is not ideal, it is the approach used in our prototype.

CONCLUSION

Polyarchies, or multiple intersecting hierarchies, are a new kind of information structure encountered in work on enterprise-wide databases. Metadirectories are a database solution for providing a common front end to a collection of databases. Metadirectories solve the synchronization and update problems with these databases, but do not solve the problem of how to effectively visualize polyarchies. The user must be able to see the relationship between hierarchies in the context of selected nodes as well as the relationships between multiple entities within a hierarchy.

Polyarchy visualization addresses the polyarchy problem as well as more generic fundamental problems of hierarchy visualizations: both the ability to scale to extremely large hierarchies and the ability to focus on multiple selections without loss of context. It does so by constraining what is viewed to the minimal set of information that answers particular questions. These solutions are packaged in views that can be easily configured by the metadirectory designer. Users can easily determine in which hierarchies an entity participates. Relationships between multiple selections are visualized in static views, while visual pivot sliding and rotation animations help the user maintain context and show how pairs of hierarchies are related to each other in the context of selected entities. Iterative design and user testing has addressed a number of usability issues and demonstrated that polyarchy visualization and visual pivot are easy to understand and use, and have high satisfaction ratings. In actual use, the polyarchy visualization allows typical office workers to easily (e.g., in a few interactions) discover information about others (e.g., people in a meeting) that would otherwise require the use of several different tools and dozens or hundreds of interactions. Polyarchy visualization is a significant step forward, allowing users to use novel views to easily discover relationships that were very difficult using standard hierarchy visualization tools.

Polyarchy visualization uses a web service architecture based on a mid-tier cache to address issues of scalability, flexibility, and responsiveness. Designing the visualization client, polyarchy server, and query language together has made it possible to address these issues. The architecture is scalable in terms of the size of the database and the number of users. The architecture is flexible by enabling the end user to easily explore multiple intersecting hierarchies while maintaining multiple foci of interest and by enabling the designer to easily specify search attributes and hierarchy views. The architecture is responsive by designing queries and responses to minimize the amount of data exchanged between client and server, and by designing the server so that minimal processing and paging is required to process each query. The architecture is also responsive because both client and server are

driven by a common database description; the server can optimize for exactly the kind of accesses the client will make.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of Brent Field for running two user studies, and George Furnas for helping distinguish polyarchies and MultiTrees. Discussions with Barbara Tversky about the benefits of interactive animation have also been greatly appreciated.

REFERENCES

1. Bartram, L., Enhancing visualizations with motion**,** *Hot Topics:Information Visualization 1998***,** North Carolina, USA, 1998**.**

2. Bederson, B. & Boltman, A. Does animation help users build mental maps of spatial information? *University of Maryland HCIL Technical Report No. 98-11*, September, 1998;

3. Bertamini, M. & Proffitt, D. Hierarchical motion organization in random dot configurations. *In Journal of Experimental Psychology: Human Perception and Performance, 26(4)*, 2000, pp. 1371-1386.

4. Box, D., Ehnebuske, D, Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., and Winer, D., SOAP: Simple Object Access Protocol. In *MSDN Library*, January 2001, Microsoft.

5. Carrière, J. & Kazman, R. Interacting with hugh hierarchies: beyond Cone Trees. In *Proceedings of Information Visualization '95*, IEEE, 74-81.

6. Card, S., Moran, T., and Newell, A., *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum, New Jersey, 1983.

7. Chi, E., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R., & Card, S. Visualizing the evolution of web ecologies. In *Proceedings of CHI'98*, ACM, 400-407.

8. Fairchild, K.M., Poltrock, S.E., & Furnas, G.W. SemNet: Three-dimensional graphic representation of large knowledge bases. *Cognitive Science and its Application for Human-Computer Interface.* R. Guindon (Ed). Lawrence Erlbaum, New Jersey, 1988.

9. Furnas, G.W. Generalized fisheye views. In *Proceedings of CHI'86* (Boston, MA), ACM, 16-23.

10. Furnas, G.W. and Zacks, J. Multitrees: Enriching and reusing hierarchical structure. In *Proceedings of CHI'94* (Boston, MA), ACM, 330–336.

11. Graham, M., Kennedy, J., & Hand, C. A comparison of set-based and graph-based visualizations of overlapping classification hierarchies. In *Proceedings of AVI 2000* (Palermo, Italy), ACM, 41-50.

12. Homer, A., *XML IE5: Programmer's Reference*. Wrox Press Ltd., Birmingham, UK, 1999.

13. Johnson, B. & Shnedierman, B. Tree-maps: A space-filling approach to the visualization of hierarchical information. In *Visualization 1991*, IEEE, 284-291.

14. Lamping, J. & Rao, R., Laying out and visualizing large trees using a hyperbolic space. In Proceedings of UIST'94, ACM, 13-14.

15. Morrison, J. & Tversky, B., The (in)effectiveness of animation in instruction. In *Extended Abstracts of CHI 2001* (Seattle, WA), ACM, 377-378.

16. Munzner, T., H3: Laying out large directed graphs in 3D hyperbolic space. In *Proceedings of Information Visualization '97*, IEEE, 2-10.

17. Proffitt, D. & Kaiser, M. Perceiving environmental properties from motion information: minimal conditions. In Ellis, S.R. (ed.), *Pictorial Communication in Real and Virtual Environments*. London: Taylor & Francis; 1993.

18. Robertson, G., Mackinlay, J., & Card, S. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of CHI'91* (New Orleans, LA), ACM, 189-194.

19. Tesler, L. The Smalltalk Environment. In *Byte*, August 1981, p. 90.