Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies

George Robertson, Kim Cameron[†], Mary Czerwinski, & Daniel Robbins
Microsoft Research & Microsoft Corporation[†]
One Microsoft Way
Redmond, WA 98052, USA
Tel: 1-425-703-1527
E-mail: ggr; kcameron; marycz;dcr@microsoft.com

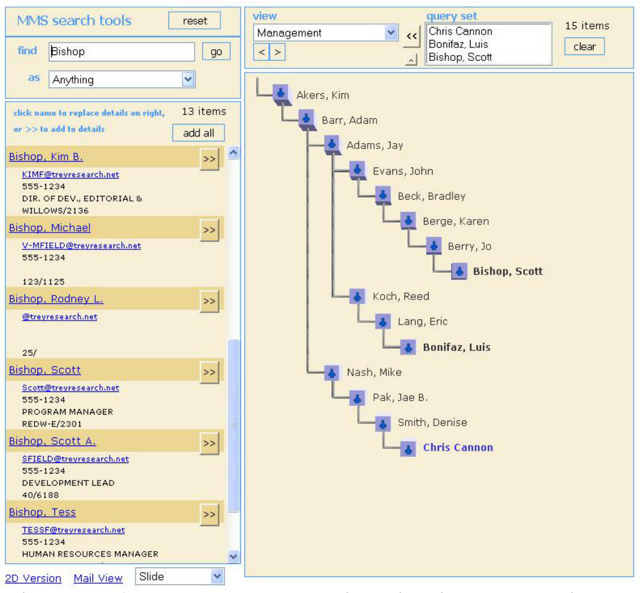
ABSTRACT
We describe a new information structure composed of multiple intersecting hierarchies, which we call *Polyarchies*. Visualizing polyarchies enables use of novel views for discovery of relationships which are very difficult using existing hierarchy visualization tools. This paper will describe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

Keywords
Information Visualization, 3D, Animation, Hierarchy, Polyarchy, Metadirectory, User Studies, Query Language

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 often exist 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 [14]. 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2002, April 20-25, 2002, Minneapolis, Minnesota, USA.
Copyright 2001 ACM 1-58113-453-3/02/0004…$5.00.



find: Bishop
as: Anything
click name to replace details on right, or >> to add to details
13 items

Bishop, Kim B.
KIMF@treyresearch.net
555-1234
DIR. OF DEV., EDITORIAL &
WILLOWS/2136

Bishop, Michael
V-MFIELD@treyresearch.net
555-1234
123/1125

Bishop, Rodney L.
@treyresearch.net
25/

Bishop, Scott
Scott@treyresearch.net
555-1234
PROGRAM MANAGER
REDW-E/2301

Bishop, Scott A.
SFIELD@treyresearch.net
555-1234
DEVELOPMENT LEAD
40/6188

Bishop, Tess
TESSF@treyresearch.net
555-1234
HUMAN RESOURCES MANAGER

Slide

view: Management
query set: Chris Cannon, Bonifaz, Luis, Bishop, Scott
15 items

```
Akers, Kim
  Barr, Adam
    Adams, Jay
      Evans, John
        Beck, Bradley
          Berge, Karen
            Berry, Jo
              Bishop, Scott
      Koch, Reed
        Lang, Eric
          Bonifaz, Luis
    Nash, Mike
      Pak, Jae B.
        Smith, Denise
          Chris Cannon
```

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

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.

When a metadirectory combines information from separate databases for display, a serious visualization problem occurs: the user must make sense out of multiple intersecting hierarchies (i.e., hierarchies that share at least one node), which we call *Polyarchies*. 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 the hierarchies as well as the relationship between entities within a hierarchy? While we have described these 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. 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 view relationships among people along a number of dimensions in a way that would have previously required using several different tools and cognitive integration of the results over time. 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 tools require many interactions to answer such a question, if it is even possible. 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 the previous example only showed 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. Perhaps more importantly, the pivot animation helps the user maintain context during complex transitions. In the following sections, we will describe how visual pivot works, briefly describe five user studies that have guided iterative design of 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 [13]; Fisheye Views in 1986 [5]; SemNet in 1986 [4]; Cone Trees in 1991 [12]; TreeMaps in 1991 [9]; Hyperbolic Browser in 1994 [10]; FSViz in 1995 [2]; H3 in 1997 [11]; Disk Trees in 1998 [3]; 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. Polyarchy visualization specifically addresses scalability and multiple focus issues.

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 [3] 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 [7] examines similar hierarchies and highlights differences between them. 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 [6] 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: VISUAL PIVOT

The goal of polyarchy visualization is to show the user how various hierarchies relate to each other in the context of selected entities, and to show how those entities relate to each other. We focus on selected entities and their paths to the root of each hierarchy, instead of general overviews of extremely large hierarchies.

The 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 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. 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.

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 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 shows in which other hierarchy views the entity participates, and allows the user to pivot to any of those views. 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.

Another key aspect of the system is how the display changes 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.

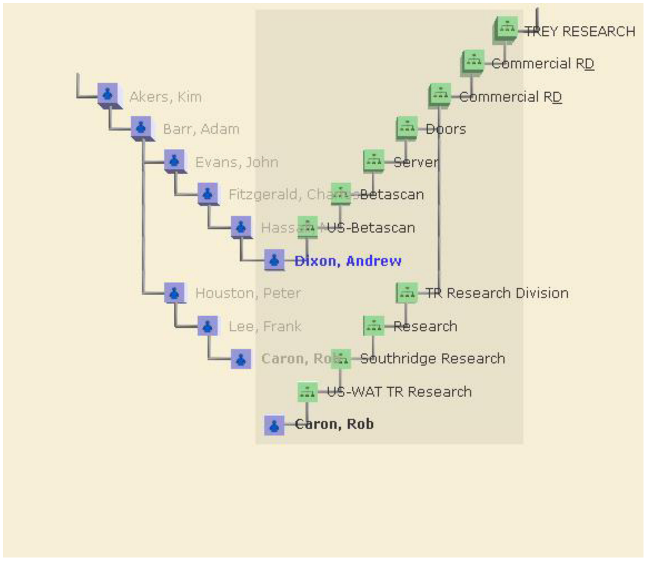


Figure 2. First stage of Visual Pivot from management to business unit hierarchy around pivot point Andrew Dixon.

While pivoting from one hierarchy view to another, one of several visual pivot animations is used to emphasize relationships between hierarchies and help the user maintain context. Figure 2 shows the beginning of one pivot animation style (vertical rotation), 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 organization units (green with an orgchart icon). The text labels of members of the query set are bold, 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).

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 3 shows an intermediate point during that animation. Finally, the new view is moved to a preferred viewing position and scale, as the old view disappears. 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 visual pivot that shows the relationship between hierarchy views and maintains context for the user. The animation may be stopped while in progress so that the user can more clearly see the relationship between views.

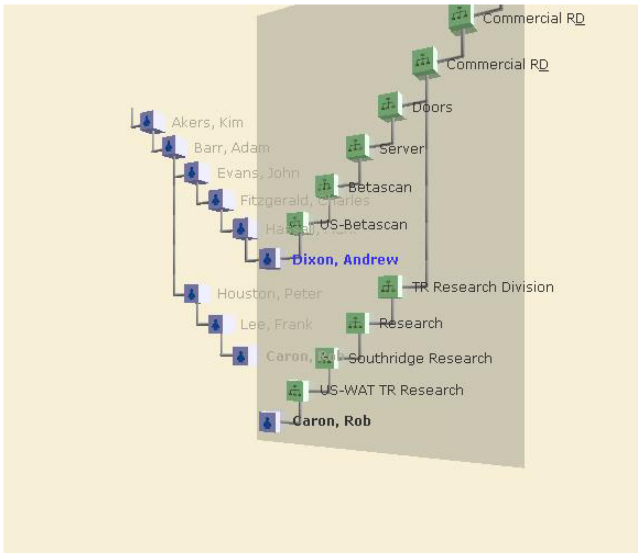


Figure 3. Visual Pivot during a pivot animation.

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 sequentially switch between multiple views. To simplify the case where the user needs to see two or more views simultaneously, a

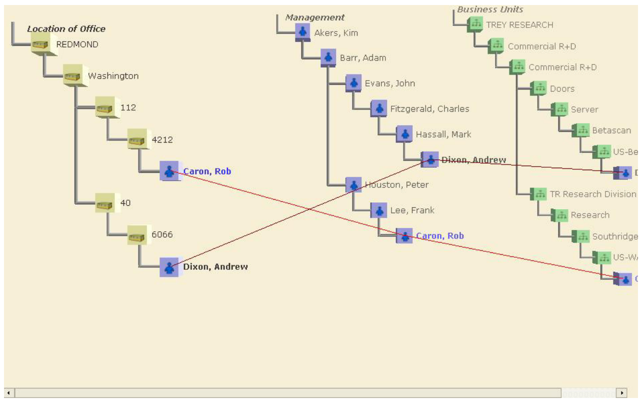


Figure 4. Stack Linked style showing two hierarchy views.

Stack Linked animation style was added. Figure 4 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 background, fading away and taking less space because of perspective view. Optional links show where selected entities appear in each view. In practice, three or four views can be shown before the display becomes too complicated.

We have experimented with eight animation styles (three vertical rotations, two horizontal rotations, 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 (intermediate to advanced PC users between the ages of 20 and 60) 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, starting with one person and building 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. Where is Evan's office?
4. What manager do Evan and Ed have in common?
5. Do Evan and Richard work in the same building?

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 of visual pivot with a 3D animated version at three animation speeds (0.8 sec; 2.0 sec and 4.0 sec). The animation style was the vertical rotation pivot 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 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 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.

Study #3

The third study was a survey of four animation styles at three 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.

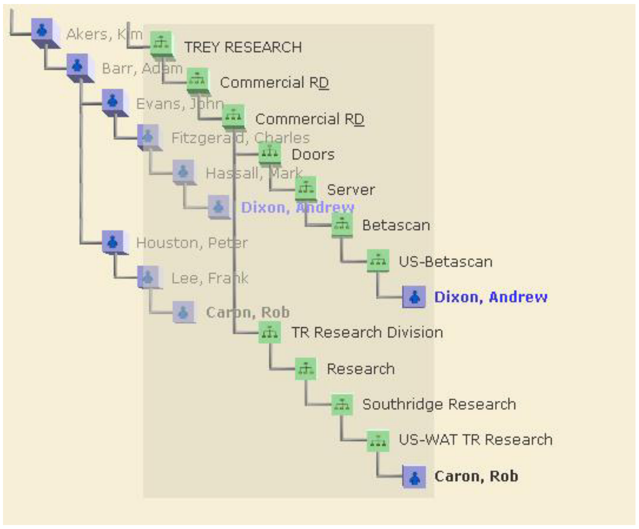


Figure 5. Sliding animation during transition from management to business unit hierarchy.

The first style was the vertical rotation used previously. Other styles attempted to fix observed problems, particularly with occlusion. The second style used vertical rotation, 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 5): the new view appeared to the right of the old view and slightly lower, then moved to the left and then up, while the old view's text was grayed out to help with occlusion. The fourth style (see Figure 6) pivoted around a horizontal axis through the pivot point, while the old view's text was grayed out.

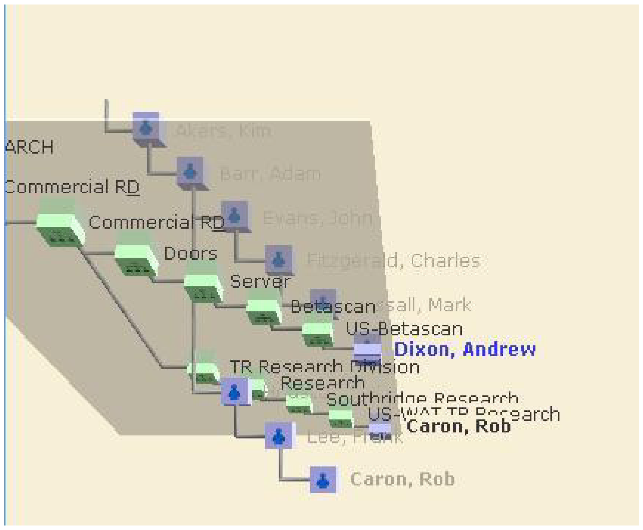


Figure 6. Horizontal rotation during transition from management to business unit hierarchy.

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 fast (by a large margin), followed by medium, then instant. No one voted for the slow speed.

Study #4

The fourth study was performed using three animation speeds (0.5 sec; 1.0 sec; and 2.0 sec), and focused on comparing the two highest rated animation styles: sliding and horizontal rotation. 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 visualization type was determined with a Latin square design. Performance times were recorded for the fifteen subtasks that occurred within each complex task.

An analysis of the completion time for each subtask showed a reliable advantage for the sliding animation over horizontal rotation (see Figure 7). There was also a significant effect of practice for both animations, with an advantage for the sliding animation (see Figure 7). There was no reliable difference between instant, fast, and medium speeds. However, the slow speed caused a significant disadvantage (see Figure 8).

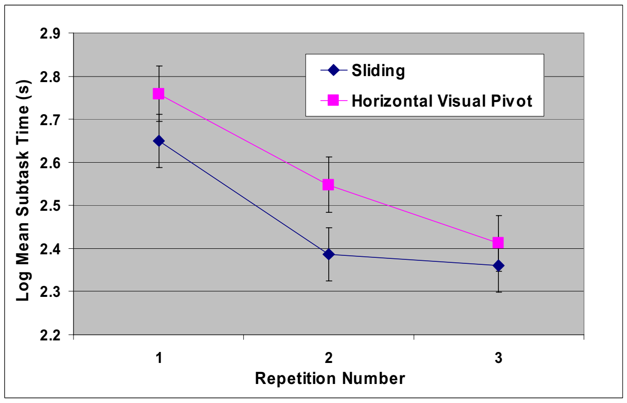


Figure 7. Study 4: Learning effects of sliding versus horizontal rotation.

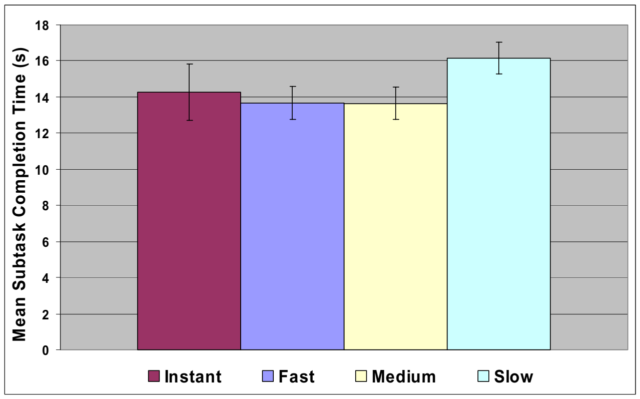


Figure 8. Study 4: Mean subtask completion time versus animation speeds.

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 horizontal rotation. The fast sliding animation averaged 6.0 on a 7 point Likert scale, while horizontal rotation averaged 5.64. 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 6 without links 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 indicated 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 showed that the sliding style was preferred by every participant. Sliding averaged 5.96 on a 7 point Likert scale, while stack averaged 4.67. Participants' comments clearly indicated 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 was ranked highest supports the hypothesis that animation helps users maintain context, since that animation had the least occlusion of all the styles tested. Additional speeds and styles are also supported since they also received high satisfaction scores.

IMPLEMENTATION ISSUES

The goal of polyarchy visualizations 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. To effectively achieve the goal, three fundamental problems must be addressed: 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 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 that obtaining the data for these basic operations must be extremely fast.

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 (as opposed to client/server) architecture supports many different client interfaces and back-end servers. 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 scripted 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 polyarchy visualization 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) [8] request and reply forms. These XML forms are packaged as SOAP (Simple Object Access Protocol) [1] remote procedure calls, which allows handling of faults in a uniform way.

Data-Driven Service

Polyarchy visualizations must be flexible not only in terms of how easily a user can explore, but also in terms of how easy it is 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.

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 UI challenge is to build an interface that is intuitive and hides 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

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 traditional 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. The 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.

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. But, the designer may want more complex views of those hierarchies. For example, "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". 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.

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 (e.g., see Figure 4). 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 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 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. Relationships between multiple selections are visualized in static views, while visual pivot sliding and rotation animations show how the hierarchies are related to each other. The animations also help the user maintain context. 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. 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 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.

REFERENCES
1. 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.
2. Carrière, J. & Kazman, R. Interacting with hugh hierarchies: beyond Cone Trees. In *Proceedings of Information Visualization '95*, IEEE, 74-81.
3. 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.
4. 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.
5. Furnas, G.W. Generalized fisheye views. In *Proceedings of CHI'86* (Boston, MA), ACM, 16-23.
6. Furnas, G.W. and Zacks, J. Multitrees: Enriching and reusing hierarchical structure. In *Proceedings of CHI'94* (Boston, MA), ACM, 330–336.
7. 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.
8. Homer, A., *XML IE5: Programmer's Reference*. Wrox Press Ltd., Birmingham, UK, 1999.
9. Johnson, B. & Shnedierman, B. Tree-maps: A space-filling approach to the visualization of hierarchical information. In *Visualization 1991*, IEEE, 284-291.
10. Lamping, J. & Rao, R., Laying out and visualizing large trees using a hyperbolic space. In Proceedings of UIST'94, ACM, 13-14.
11. Munzner, T., H3: Laying out large directed graphs in 3D hyperbolic space. In *Proceedings of Information Visualization '97*, IEEE, 2-10.
12. 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.
13. Tesler, L. The Smalltalk Environment. In *Byte*, August 1981, p. 90.
14. The Burton Group. MetaDirectory FAQ. Catalyst '99 Conference, July 1999. See http://www.netapps.org/Events/HTMLDocs/workshopmetadirectoryfaq.htm.