Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation

Jeffrey Heer, Jock D. Mackinlay, Chris Stolte, and Maneesh Agrawala

Abstract—Interactive history tools, ranging from basic undo and redo to branching timelines of user actions, facilitate iterative forms of interaction. In this paper, we investigate the design of history mechanisms for information visualization. We present a design space analysis of both architectural and interface issues, identifying design decisions and associated trade-offs. Based on this analysis, we contribute a design study of graphical history tools for Tableau, a database visualization system. These tools record and visualize interaction histories, support data analysis and communication of findings, and contribute novel mechanisms for presenting, managing, and exporting histories. Furthermore, we have analyzed aggregated collections of history sessions to evaluate Tableau usage. We describe additional tools for analyzing users' history logs and how they have been applied to study usage patterns in Tableau.

Index Terms—Visualization, history, undo, analysis, presentation, evaluation.

1 Introduction

When investigating data with visualizations, users regularly traverse the space of views in an iterative fashion. Exploratory analysis may result in a number of hypotheses, leading to multiple rounds of question-answering. Analysts can generate unexpected questions that may be investigated immediately or revisited later. After conducting analysis, users may need to review, summarize, and communicate their findings, often in the form of reports or presentations.

By surfacing users' interaction history, we can facilitate analysis and communication. History mechanisms such as undo or "time-travel" enable revisitation in a variety of applications (e.g., [1-4, 6, 8, 12-14, 16-20, 22, 24-26]). As noted by Shneiderman [27], such history tools can play an important part in the visualization process, supporting iterative analysis by enabling users to review, retrieve, and revisit visualization states. Moreover, history tools can help users create reports or presentations, facilitating communication.

Interaction histories can also benefit research and development. History log analysis of both individual and aggregate usage can identify common usage patterns and thereby assist usability evaluation. Researchers can also study interaction patterns to better understand and model analysts' sense-making process [13].

However, the best history mechanisms for achieving these benefits are not always clear. Designers of visualization tools must consider a large design space of potential features and system architectures when designing history tools. These design decisions entail trade-offs in the types of history representations and operations that can be provided.

For example, while it is easy to log low-level input events such as key presses and mouse clicks [25], users can more readily take advantage of semantically meaningful models. Many operations might be performed on an interaction history, including editing, aggregation, bookmarking, annotation, and search. Architecture and interface design need to account for such operations. Furthermore, interaction histories can grow large quickly, and thus history mechanisms must scale accordingly. Scale concerns arise at the data level, where histories can benefit from compact description, and at the visual level, where history interfaces should be perceptually effective and space efficient.

In this paper, we explore the design of graphical history tools to support visual analysis. We first present the results of a design space analysis, enumerating design decisions for the software architecture and graphical interface of history systems. Our analysis is intended to provide an overview of important design considerations and thereby help practitioners incorporate graphical history tools into their own visualization applications.

Inspired by our design space analysis, we then present the design and implementation of graphical history tools to support analysis, communication, and evaluation in Tableau, a database visualization system [21, 28]. Although our primary contribution is a design study of history tools for visual analysis, our graphical history prototype also contributes new techniques for improving scalability, searching histories for relevant views, and generating presentations from history subsets. Furthermore, we have used our history model to support evaluation by analyzing recorded usage data. We describe our visual history analysis tools and how we have applied them to improve Tableau's user interface. We also calculate estimates of the impact of our history management techniques, finding that our techniques can reduce visualized history state spaces by over 60%.

2 Design Space Analysis of Interaction Histories

Architects of interactive history systems face a number of design decisions impacting the representations and operations available to users. To design our history tools, we first conducted a design space analysis to enumerate these decisions. We surveyed prior work spanning general history mechanisms [2, 8, 9, 23, 29] and interface designs in the areas of graphical design tools [8, 16, 17, 22, 26], web browsing [1, 5, 12, 14, 15, 30], and visualization and simulation [3, 4, 6, 10, 11, 13, 18, 19, 20, 24, 25]. In this section, we outline the design space of history tools using examples from this body of work.

2.1 History Models

2.1.1 Actions vs. States

We model interaction histories as movement through a graph of application states. Nodes in the graph represent discrete *states* of the application and edges represent the *actions* that transform one state into another. A state is defined by the settings of interface widgets and the application content (e.g., document, data, etc). At the

- *Jeffrey Heer is with the University of California at Berkeley, E-Mail: jheer@cs.berkeley.edu.*
- *Jock D. Mackinlay is with Tableau Software, Inc. E-Mail: jmackinlay@tableausoftware.com.*
- *Chris Stolte is with Tableau Software, Inc. E-Mail: cstolte@tableausoftware.com.*
- *Maneesh Agrawala is with the University of California at Berkeley, E-Mail: maneesh@cs.berkeley.edu.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

architectural level, developers must decide if their history system will maintain sequences of states, actions, or both, and how such *history items*—discrete representations of an action or historical state—will be organized.

Action logging is often referred to as the *command object* model [9]. Command objects encapsulate an interface action, typically providing both do and undo methods that apply the operation or its inverse. To traverse the history, a sequence of commands can either be done or undone in order. This approach requires that suitable inverse (undo) operations are defined for all actions.

An alternative is to log the individual states of the application. Traversing the history then involves restoring the application state to a stored configuration, removing the need to sequentially apply undo actions. However, the drawback of this approach is that the state representation can become memory inefficient.

The action and state approaches are not mutually exclusive and hybrid approaches are possible. For example, an action-based history mechanism might periodically cache the state to reduce the number of operations required by history traversal. A state model might also log metadata about the operations that were applied between states. For example, the WebQuilt web logging system [30] stores URLs (states) but also notes the index of the link clicked in the previous page, modelling web browsing at the level of individual links.

In surveying the literature, we have found that action logging is prevalent within graphic design tools, where large content models can make state models memory-inefficient. In contrast, state logging (as URLs) is common for web browsing histories. Visualization systems have utilized both approaches. As discussed later, this choice affects the range of history operations that users may perform, particularly with respect to editing and selective undo.

A common approach in visualization is to describe the visualization in terms of a chain of visual encoding operators that are applied to the data to generate the visualization state. Jankun-Kelly et al. [13] introduce a general model for visualization state as a set of parameters, and actions as transformations of these parameters. Heer et al. [11] note that identical visualization views can be reached through different parameter sets. In particular, different filtering criteria may yield the same result set. Thus, accurate analysis of revisitation may require that state models include an index of the underlying content in addition to parameter settings.

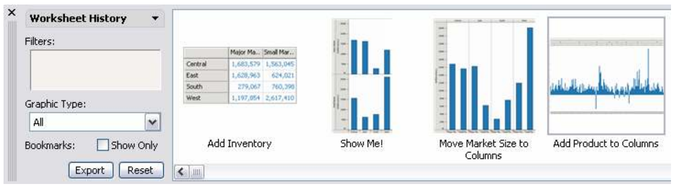
One modelling issue specific to visualization is its data-driven nature: application states are dependent on the backing data set. If a visualized data set includes streaming or editable data, a faithful history system must also take the changes to the data into account. It may be that users want historical states to update with changes to the data, thereby keeping their analysis current (a form of selective redo, discussed later). If not, data management systems that support versioning or provenance may be used; however, such systems may entail an unacceptable storage cost. As a visualization view might depend only on a subset or aggregate of the backing data, in many cases creating an extract of the data for a "snapshot" of the visualization state may be a feasible solution.

2.1.2 History Organization

History items may be organized in various ways. The *stack model* places items on both undo and redo stacks. This approach does not support branching histories, as the redo stack is cleared when new actions occur. A *timeline* model stores items in the linear order in which they occur. *Branching* models [29] store items in a tree structure, and actions performed after undo operations form a new branch of the tree. Additionally, history models may perform *content indexing* and organize history items by other metadata properties.

2.1.3 Hierarchical Command Objects

Systems may represent history items at multiple granularities. For example, one can group a sequence of low-level actions into a higher-level action through hierarchical command objects [23]. Grouped actions may provide a better semantic description of a user's intention. To construct groupings, developers can craft



	Major Ma...	Small Mar...
Central	1,683,579	1,563,045
East	1,628,963	624,021
South	279,067	760,398
West	1,197,854	2,617,410

Fig. 1. A Graphical History Interface. Thumbnails show previous visualization states and labels describe the actions performed.

"chunking" rules [17] based on the type and timing of actions. However, groupings also raise challenges for representing and navigating hierarchical history items in a user-friendly manner.

2.1.4 Local and Global History

One can organize history items by the objects on which actions are performed. For example, a spreadsheet may maintain separate histories per worksheet, while a graphics editor could maintain local histories for objects in the scene. Edwards et al. [8] propose a transactional model to support local histories in which actions may have global side-effects. In all cases, applications must support the ability to merge local histories into a global timeline.

2.2 Visual Representations of History

2.2.1 Visual Presentation

One simple presentation of a history item is a text description of the state or action, commonly found as menu text for undo and redo actions. Text descriptions should be easy to understand, and may require subtle design decisions. For example, web history systems have carefully considered different abbreviation approaches for web page titles and URLs [1, 15]. While text may be helpful for describing actions performed in a visualization, they are less well suited for the graphical nature of a visualization state.

Graphical representations of histories are also common. Some depictions involve abstract properties: for example, the color of a history item glyph might represent the type of action performed. Most common, however, are thumbnail images used to aid users' recognition of the previous interface state—an approach particularly relevant for visualizations [20]. Multiple studies have found benefits for thumbnails in web browsing [15, 31], with one study suggesting that a thumbnail size of about 120 pixels square is enough to enable 80% accurate recognition of a visited web site [15]. Other projects [16, 17, 30] enhance thumbnails to improve comprehension by highlighting changes and applying strategic callouts and cropping.

2.2.2 Spatial Organization

Depending on the underlying history model, a number of visual organizations of history items are possible. A common approach is a linear sequence of items, like a comic strip [17, 22]. Such an organization facilitates visual scanning of the history, and typically enables navigation by clicking an entry. A similar approach is to provide a continuous timeline [6, 24, 26], which shows the time duration between actions and is navigated using a slider control.

Branching histories typically use a node-link tree diagram to show history branches. Prior work has adapted both the sequence [1, 3, 4, 12, 18] and timeline [6] metaphors into branching tree displays. Klemmer et al. [16] present an inline branching design that places collapsible history branches within a linear comic strip.

Other representations are also possible. Behavior graphs [5] are an alternative representation of branching histories that we will discuss in section 4.1. Another approach is to use a content-centric representation using variables other than time. WebQuilt [30] visualizes aggregate surfing behavior in a network diagram showing traversed links between web page thumbnails. Ma's Image Graphs [20] display history states as thumbnails connected in a graph layout and depict actions between states using iconic edge representations.

2.3 Operations on History

Designers need to also consider the set of operations that their graphical history tools should support.

2.3.1 Navigation

For end-users, the fundamental operation of history systems is navigation to states in the history. Undo and redo (or back and forward) actions are common navigation operations found in many applications. Another approach is for users to click the thumbnail of a history state to directly return to that state. Some systems use a "time travel" metaphor with a timeline slider. For branching histories, a graphical history can help users differentiate branches. Content-based navigation is also possible, such as navigating to the point in time that an object was last edited [26].

2.3.2 Editable Histories

Other operations may involve editing the history, as users may wish to revise the history or replay past actions. In state-based history models, deleting a past state from the model doesn't affect any of the other states. In action-based models, editing has side-effects. Deleting a past action involves rolling back the history prior to the selected action and re-applying the subsequent actions. However, some subsequent actions may be dependent on a side-effect of the deleted action, so rules to ensure integrity are needed.

While history editing is more complicated for action-based models, it enables unique operations. Selective undo [2] allows the replay of past actions after revising the history. Similarly, selective redo [6, 17, 18] allows chains of actions to be copied and reused. Kurlander and Feiner [17] use this mechanism to support macro creation. For example, a sequence of visualization transforms might be re-applied to a new subset of data [6, 18].

2.3.3 Metadata and Annotation

Users may also wish to add metadata and annotations to history items. Bookmarking [11, 14], keyword tagging, text comments [10, 11, 16, 24], and audio annotations [10] are all potentially useful. Usage scenarios for visualization include analysts creating bookmarks for important findings [14], leaving text notes to describe a view to a collaborator [16], and recording audio annotations to aid "think-aloud" evaluation protocols [10].

2.3.4 Search and Filter

As histories grow large, users may need means beyond visual search and scrolling to find past states of interest. Search tools are one solution. Metadata such as time, action type, bookmarks, and annotations are all potential search domains. Although filtering tools have been provided for web design histories [16], most visualization histories [3, 6, 10, 18, 19, 20, 25] lack search capabilities. A notable exception is VisTrails [4], which enables querying-by-example to find related visual exploration sessions across multiple users.

2.3.5 Export

To enable communication, it is often important to export and share parts of a history. For web-based systems, one can distribute a URL [4, 11], but desktop applications are typically more cumbersome. Klemmer et al. [16] print out thumbnails and text annotations as paper reports. However, nearly all history tools are lacking more nuanced support for exporting histories into external media.

2.4 Summary

In this section, we have categorized a range of design decisions that arise when crafting an interactive history system. These decisions include how to represent and organize historical data (e.g., states, actions, or both), how to visually present histories (e.g., linear or branching layout), and what interactive operations the history should support (e.g., navigation, editing, search, and export). Still, the task remains of deciding which route to take when designing a system. The features and context of use of the underlying visual analysis tool can further inform the design process for history tools. To illustrate this process, we now apply our design space analysis to develop an interactive history system in the context of Tableau, a database visualization system.

3 Graphical History in Tableau: A Case Study

Based on the considerations raised by our design space analysis, we designed a history interface supporting analysis and communication in Tableau, a commercial visual analysis system. We now describe Tableau and present the design of our graphical history tools.

3.1 The Tableau Visual Analysis System

Tableau is a commercial system, based on Polaris [28], for visualizing the contents of databases. As shown in Fig. 2, the Tableau interface includes a list of available database fields and a workspace in which users can select fields and drag them onto shelves corresponding to visual encodings such as position, color, shape, and size. Tableau is based on a specification language called VizQL. VizQL statements are generated from the contents of the interface shelves and they specify both the data that should be visualized (as database query statements) and how the visualization should appear (as visual specification statements). This formalism supports a range of visualizations, including bar charts, time series, scatter plots, and heat maps, as well as analytic operations such as filtering, sorting, and drill-down [28].

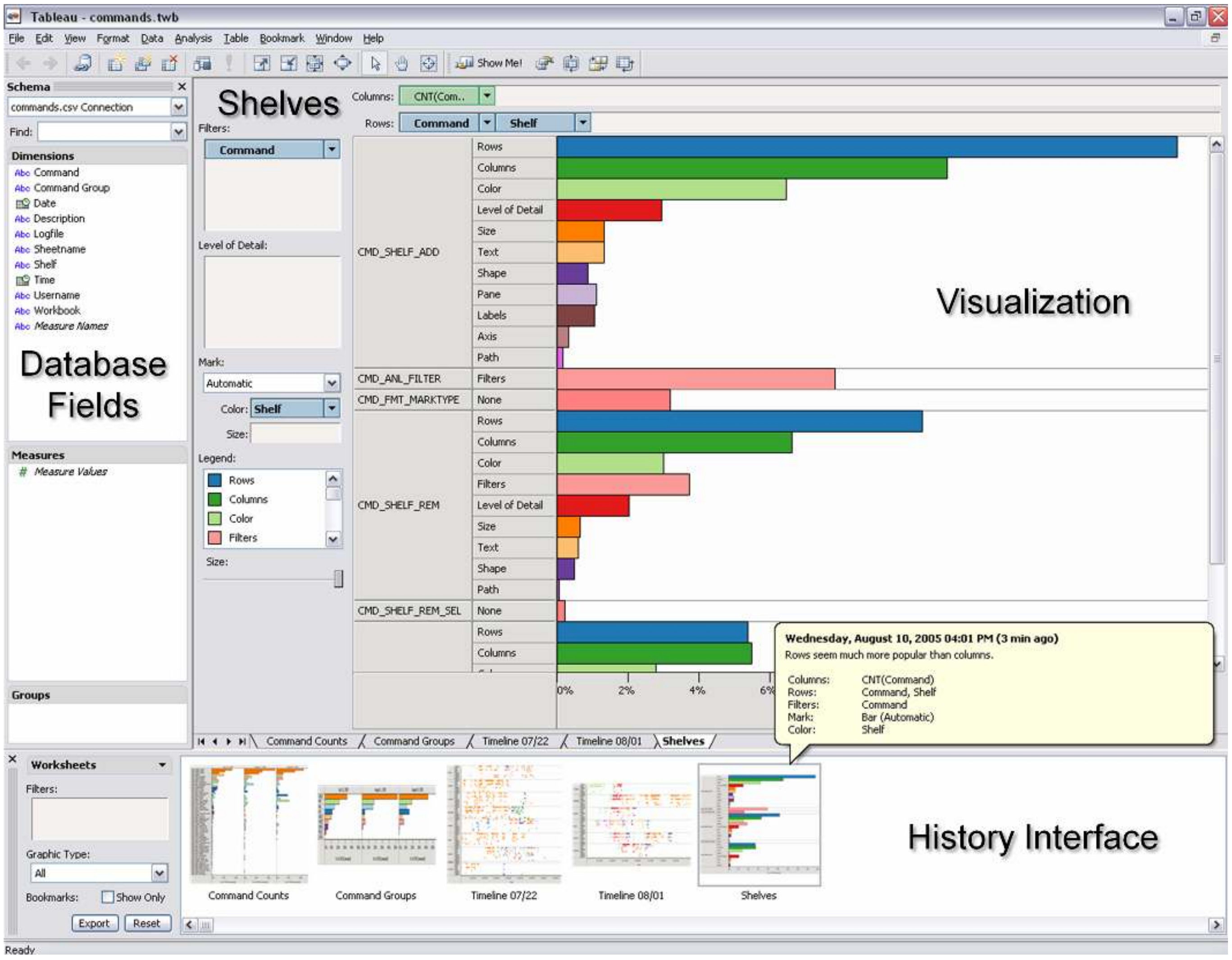
Akin to Microsoft Excel, Tableau supports multiple worksheets. Each state of a Tableau worksheet is described by a VizQL statement. Tableau's original history model used a state-based logging approach, with each worksheet organizing VizQL statements on undo and redo stacks. This model does not support branching histories, except through duplication of worksheets. Undo and redo buttons provide some support for history navigation, but the model does not provide text descriptions for undo/redo actions. In the following sub-sections, we describe a redesigned model to better support analysis and communication.

3.2 A Re-designed History Model

In crafting a history model for Tableau, we wanted to maintain the existing, clean approach of declaratively modeling state as VizQL statements. However, VizQL statements alone are not enough, as we also wanted to record historical data that enables us to provide high-level descriptions of user actions, both to provide a more informative user interface and to support usage evaluation. As a result, our improved history model uses a hybrid state/action approach as identified in our design space analysis.

History items in Tableau still record states as VizQL statements, but we also introduced aspects of action-based logging. We created a classification scheme for each action supported by the interface. When an action occurs, its unique identifier and any arguments are passed to the history system, which stores the command description and the current VizQL statement as a history item. Having a record of actions allows us to create text descriptions, improving the cues for undo and redo within the interface. Our classification scheme groups actions into five top-level categories: *shelf* (add, remove, replace), *data* (bin, derive field), *analysis* (filter, sort), *worksheet* (add, delete, duplicate), and *formatting* (resize, style) commands.

In addition to basic history items, our model supports composites of grouped sub-items, similar to hierarchical command objects [23]. All history items support data fields such as a timestamp, bookmark status, and text annotations. We organize items in a branching structure for each worksheet, replacing the prior stack model. When a user visits a past state and performs an action, a new analysis branch is added to the model (see Fig. 5). Our history abstraction also supports merged histories, implemented as a composite history view of worksheet histories. By default, the state model does not include the database contents and changes to the database will cause historical states to update to reflect the current data. However, users can create data extracts if desired, ensuring a static data set.



Wednesday, August 10, 2005 04:01 PM (3 min ago)
Rows seem much more popular than columns.

Columns: CNT(Command)
Rows: Command, Shelf
Filters: Command
Mark: Bar (Automatic)
Color: Shelf

Fig. 2. The Tableau Visual Analysis Tool, visualizing data collected from aggregated history usage logs. The panel on the left provides a list of database fields. Fields can be dragged onto visual encoding shelves in the workspace on the right to create visualizations. Multiple worksheets are supported, indicated by the named tabs underneath the visualization. The panel along the bottom shows an analysis history viewer, currently providing an overview of the current state of each worksheet. A tooltip provides details-on-demand for the selected history item.

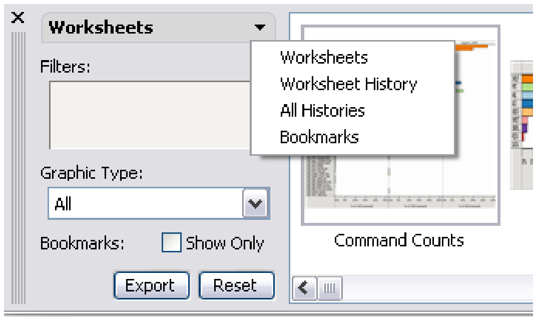


Fig. 3. History Interface. A drop-down menu determines which history items are shown. History can be filtered by data fields (via drag-and-drop), chart type, and bookmarks (§3.5.2).

3.3 Design of the Graphical History Interface

We designed our history representation with the understanding that graphical history should aid analysis in an unobtrusive fashion. The visualization should serve as the primary focus of attention and the history as an auxiliary display. We wanted to ensure that the graphical history "pays for" its screen real estate, using only the space needed for effective presentation and navigation of history items.

As a result, the history model is depicted using a sequential, comic-strip display (Fig. 1, Fig. 2 bottom), including a thumbnail image and text description for displayed history items. As tree diagrams can require a lot of screen space, we present branching histories inline: branch contents are listed sequentially, with sibling branches sorted by the timestamp of the first item. We position the history viewer along the bottom of the interface. Users can optionally hide the viewer to make more space for the visualization. Hovering the mouse pointer over a history item reveals a tooltip with details-on-demand (Fig. 2). The tooltip lists the time the state was first visited in absolute and relative ("3 min ago") time. The tooltip includes text annotations added to the item and a summary of visual encodings: which data fields are placed on which shelves.

To maximize the usefulness of the history display, the interface provides four modes, accessible via a drop-down menu (Fig. 3):

- *Worksheets* mode presents an overview of the current state of all worksheets, with a thumbnail and name for each (Fig. 2).
- *Worksheet History* mode presents the history of a worksheet. Thumbnails are captioned with action descriptions (Fig. 1, 6).
- *All Histories* mode is similar to Worksheet History mode but depicts the merged global history across all worksheets.
- *Bookmarks* mode shows all views that have been bookmarked. Captions include the source worksheet name and a timestamp.

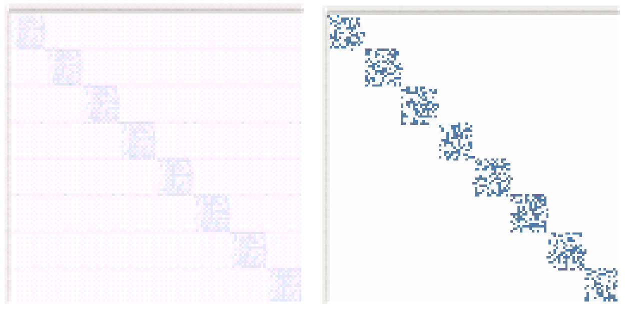


Fig. 4. Adjusting Thumbnail Contrast. The image on the left is an overview thumbnail generated by down-sampling that suffers from "wash out". On the right, high-frequency elements such as gridlines have been removed and pixel values are adjusted such that the data color in the image matches the color encoding palette.

3.3.1 Thumbnail Image Generation

The history viewer provides thumbnails of visualization states to aid recognition. Thumbnail size introduces a trade-off between screen usage and recognizability. Based on the experimental results in [15], we chose 120 pixels square. As noted by our design space analysis, some graphics editors [16, 17] provide enhanced thumbnails that highlight differences between history states and perform selective cropping. As changes in Tableau regularly involve complete updates of the visualization, this approach did not seem appropriate. Views in Tableau often require scrolling, so we reasoned that a thumbnail that provides an overview of the display as well as historical data would be the most useful for facilitating analysis.

To generate the overview images we render the visualization at its native resolution and then scale the resulting image. We place limits on the image buffer size, cropping the image as needed to constrain memory usage. We also avoid extreme aspect ratios by non-uniformly scaling the image when needed. In some cases, down-sampling a large overview can result in a "washed out" image. For large images, our thumbnail generation routine first modifies the visualization, removing high-frequency visual elements such as gridlines and element borders. In the resulting thumbnail, pixels with brightness over a threshold value are then scaled towards the nearest color in the color encoding palette (see Fig. 4).

3.4 Navigating and Managing History

By visualizing past analysis states, our graphical history display facilitates revisitation. Users can click a thumbnail to skip back to a prior state. If a user performs analysis operations while visiting a prior state, a new analysis branch is created and depicted in the graphical history. However, as these histories can quickly become unwieldy, we have implemented additional techniques to reduce the complexity of the display and filter unneeded views. Figure 5 depicts our model and how it is mapped into a visual display.

3.4.1 Manual Editing

We support manual editing so that users can delete unwanted states from the history. As we use state-based logging, deleted states are simply removed from their history branch and do not impose side effects on other history items. However, one caveat is that deletion can result in an incomplete timeline in Worksheet History mode, in which text descriptions of prior actions are provided.

3.4.2 Chunking

When a group of related actions are performed in sequence, they may be better represented as a single higher-level event. For example, in a word processor the keystrokes [c][h][u][n][k] might be represented as the word [chunk]. To support such chunking our system provides hierarchical history items. In the spirit of Kurlander and Feiner [17], we have hand-crafted a set of "chunking" rules to coalesce actions into a grouped history item. As new states are added to the history, the rules evaluate if the new state should be chunked with the previous state. A set of predicates expressing the chunking conditions are applied and if any evaluates to true (and no exception rules do) the new state is chunked with the previous state.

We have implemented three chunking rules based on empirical usage data, along with some exception cases. In an analysis of user activity (sec. 4.3.2), we found that rapid sequences of formatting actions are common and could benefit from aggregation. Accordingly, we include a rule that chunks history items if the most recent state was the result of a formatting operation. Similarly, a quick succession of sort or filter actions (less than 30 sec. apart) likely indicate a multi-step configuration of the view and are chunked together. A rapid series shelf actions to build up (or take down) a view are also common, and so we chunk them when separated by less than 5 sec. We also support exception cases: large time durations—possibly indicating a break between sessions—prevents any chunking, as does bookmarking or annotating a state.

When our rules determine that two actions should be chunked, the thumbnail in the history view updates in-place and no new thumbnails are added to the view. Users can click in the history view to skip to a state prior to the chunked sequence. However, undo events will step back through each of the chunked actions individually. We believe that this interaction is less complicated than an interface providing explicit level-of-detail controls (e.g., [17]).

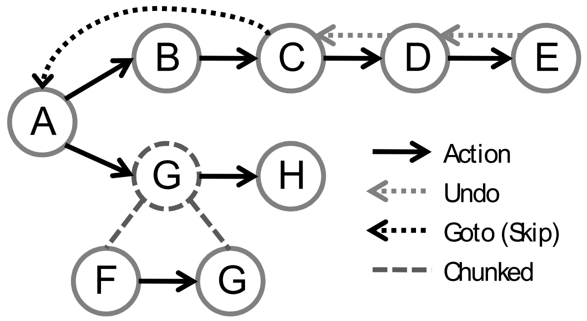


Fig. 5a. History Management. A user performs actions to go from state A to state E, performs two undo actions, and then skips back to state A. The user performs new actions to go to states F, G, and H. Chunking rules determine that states F and G should be coalesced.

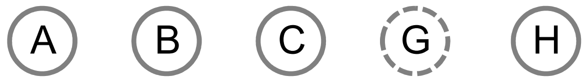


Fig. 5b. Visual Presentation of History Model. The states in Fig. 5a are presented in a linear sequence. States D and E are culled by undo-as-delete (§3.4.3), and states F and G are coalesced due to chunking rules (§3.4.2). Branches (starting at states B and G) are listed inline.

3.4.3 Undo-as-Delete

Undo actions may be the result of varied intentions. For example, an undo may be viewed as a navigation action, moving to a previous state with the intention of later rolling forward again. Alternatively, an undo may serve as a "delete" operation to recover from a mistake or an undesirable action. Similarly, Shipman and Hsieh [26] have argued that a quickly-made undo is probably a delete. In our effort to improve the scalability of graphical histories, we hypothesized that most uses of undo fall within the latter category, such that "undone" states are rarely revisited. As described in section 4.3.1, we empirically tested this idea, finding that undo actions were over 12 times more common than redo actions.

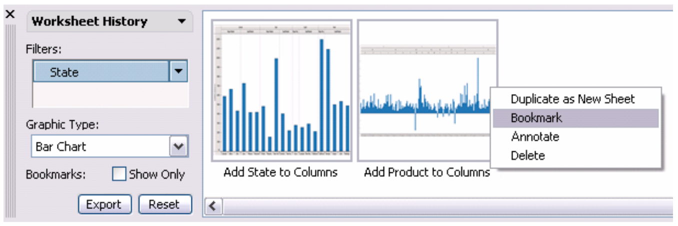


Fig. 6. Filtered History showing all Bar Charts that include the data field “State”. A context menu provides operations on history items.

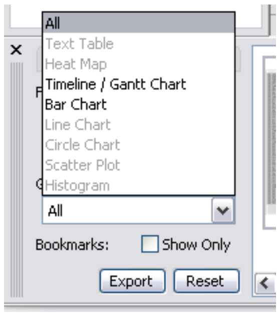


Fig. 7. Filter by Chart Type. A selection menu highlights the chart types currently available in the interaction history.

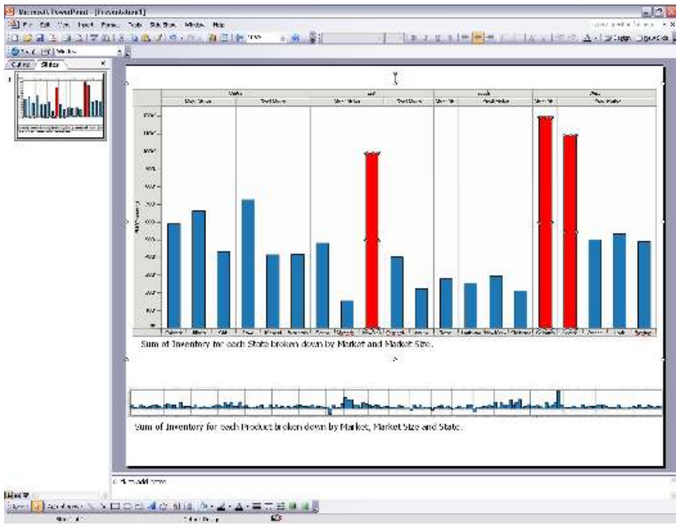

Sum of Inventory for each State broken down by Market and Market Size.
Sum of Inventory for each Product broken down by Market, Market Size and State.

Fig. 8. Tableau visualization exported into PowerPoint. The history “Export” feature can be used to seed presentations with captioned, editable versions of Tableau visualizations.

As a result, we developed a new history management technique we call *undo-as-delete*. As a user performs actions, new items are added to the graphical history. When a user clicks the back button to perform an undo, the last state is removed from the graphical history. The underlying model still maintains the state, and thus subsequent redo actions work as expected, with the history item returned to the graphical history. However, if the user never executes a redo, the undone analysis branch is discarded as in a stack-based organization.

This approach enables unneeded history to be automatically culled, reducing the complexity of the history model. Branching histories can still be created by navigating to past states using the graphical history rather than the undo button. New actions will then result in a new branch without deleting the previous branch, thus preserving the analysis trail. Similarly, we automatically disable undo-as-delete if user interaction suggests a view is important. Bookmarking or annotating the current state exempts it and previous states on the same branch from deletion. In future research we plan to see if other indicators such as selections might prove desirable.

3.4.4 History Navigation and Management Example

Assume a user performs four actions in a row, as shown in Fig. 5, to move through states A→B→C→D→E. The result is a linear list of states. The user then undoes the previous two actions, moving back to state C. Our undo-as-delete rules automatically hide states D and E to reduce the complexity of the history. The user next explores an alternative analysis, first skipping back to state A by clicking its thumbnail in the graphical history view, and then performing new operations to move through states A→F→G→H. When the user performs a new operation to go to state F, the undo-as-delete rules delete states D and E from the underlying model. Furthermore, when chunking rules determine that states F and G are similar, they are coalesced into the single entry G in the graphical display. As shown in Fig. 5b, the abbreviated branch G→H is presented in sequence after branch B→C, as sibling branches are sorted by the timestamp of the first entry in the branch. Similarly, if the user were to skip back to state B and perform new actions, the new branch would be placed in the list sequence directly after state C.

3.5 Operating on History

As discussed in our design space analysis, operations on history models can support sensemaking, search, and communication. Guided by these concerns, we have incorporated operations for affixing metadata to history states, dynamic querying of the history interface, and exporting histories to support sharing and presentation.

3.5.1 Metadata: Bookmarks and Annotations

By right-clicking an item, users are presented a context menu with which they can bookmark that state or add a text annotation (Fig. 6). Bookmarked views are then available in the *Bookmarks* mode of the history viewer. We have also considered adding a keyword tagging feature; this could be achieved by generalizing the bookmarking feature to include user-provided text labels. Text annotations are available in tooltips when the mouse hovers over an item (Fig. 3).

3.5.2 Search and Filter

Even with mechanisms for combating scale (sec. 3.4), histories can grow large. To help retrieve states of interest, we have introduced multiple search features (Fig. 6). We hypothesized that the type of visualization and the data fields used are salient aspects with which users might recall past states. Our history viewer supports filtering by data field by reusing the shelf metaphor for visual encodings. Users can simply drop a data field into the history filter shelf to limit the view to only those states that include the data field. A combo box allows users to further limit the history view to specific chart types (Fig. 7). These filtering operations are implemented by indexing the VizQL expressions stored with each history item. Users can also use a checkbox to limit the view to bookmarked history items.

3.5.3 History Export and Sharing

Finally, the history viewer provides export features to share and communicate findings. By clicking the “Export” button, users can view a menu of export options. Selected history states can be output as a saved Tableau file, allowing reloading of the states as a set of worksheets. Visualization views for selected history states can also be exported as either bitmap or vector images, and can be embedded in reports and presentations. By exporting Tableau visualizations in the Windows Metafile format, we can export a set of history states directly into PowerPoint slides as editable graphics. Analysts can automatically generate a slide deck from a set of selected history states and then annotate and edit exported visualizations in PowerPoint directly, as in Fig. 8.

4 Using History to Evaluate Visualization Design

While the previous section focuses on graphical histories to support end-users, we also used histories to evaluate Tableau. By exporting and aggregating history logs, we can analyze user behavior. Here we discuss two analysis approaches and present some of our findings.

4.1 Analyzing Individual Usage with Behavior Graphs

We have explored tools for analyzing individual usage sessions. One technique we have found useful is behavior graphs, which we model after Card et al.'s web behavior graphs [5]. Figure 9 shows a Tableau session visualized in a behavior graph. The graph is read in a snake-like fashion. Actions are listed left-to-right except for Undo events, which are placed right-to-left on a new row. Subsequent actions resume left-to-right ordering on a new row. Vertical columns often contain the same state, making revisitation patterns clear. Color is used to indicate the types of actions performed by users. We have found these visualizations particularly useful for understanding patterns of branching and revisitation.

4.2 Analyzing Aggregate Usage

Analysis of aggregate usage is also important for determining usage patterns. For these and other history analysis tasks, we have used Tableau itself. First, we map each history log into a tabular format. Columns in these tables include timestamps, session ids, user ids, worksheet names, and actions performed. We store the resulting logs in a database which we then visualize in Tableau. Our taxonomy of commands (sec. 3.2) enables us to analyze command usage at multiple levels of granularity. Figure 2 shows Tableau being used to analyze the results of collected history logs: the primary display shows a histogram of command usage, while the graphical history display contains thumbnails for other analyses. An analysis of aggregated usage timelines is shown in Figure 10.

4.3 Findings

By analyzing user histories, we have made a number of findings to improve the design of Tableau's interface and estimate the impact of our history management techniques. Here we describe four such examples. All usage data has been collected using a version of Tableau that includes our augmented history model, but without a graphical history interface. Usage data has been collected from 9 Tableau employees and 27 customers willing to share their data. The data consists of 20,192 actions from 36 users, with a median of 350 actions per user. Of these, 17,401 actions result in visual history items, as non-visual actions—such as opening a workbook or adding a derived field—are not included in the history interface.

4.3.1 The Undo / Redo Ratio

As we designed our history interface, we wanted information about how users used the existing undo and redo features. Looking at the usage logs, we found a total of 1,023 undo events and 82 redo events: undo was ~12.5 times more common than redo. Thus, most undone actions were never revisited, a finding that supports our undo-as-delete model for managing histories (sec. 3.4.3).

4.3.2 The Prevalence of Formatting

When analyzing command usage, we found that formatting actions, in which users adjust size and styling, accounted for 23.8% of all actions. Furthermore, they were performed in succession: 73.6% of all formatting actions were followed by an additional formatting action. In response, we crafted chunking rules (sec. 3.4.2) that coalesce all formatting events. As runs of consecutive resize events were common, subsequent development effort has also focused on improving Tableau's automatic sizing routines.

4.3.3 Use of Automated Presentation Tools

Tableau's automated presentation features (named "ShowMe") [21] help users create more perceptually effective visualizations. We used history data to evaluate usage of these features by end-users. For example, we found a relatively low rate of mark type adjustment (560 mark changes among 8,248 shelf changes, for a 6.8% error rate), suggesting that the automatic selection of mark types was helpful. We also discovered that analysts used ShowMe features throughout usage sessions, suggesting that ShowMe commands had become a regular part of their visual analysis.

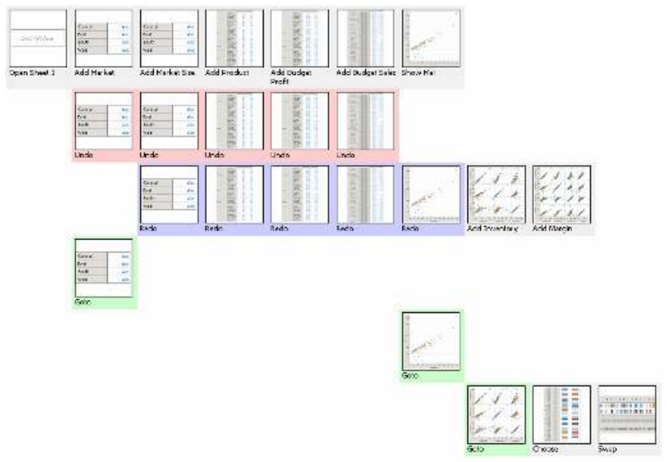


Fig. 9. Tableau Behavior Graphs depict user behavior in an analysis session. Actions except undo and goto are placed sequentially in left-to-right order. Undo actions (red) move right-to-left on a new row. Goto actions (green) indicate navigation actions made in the history viewer.

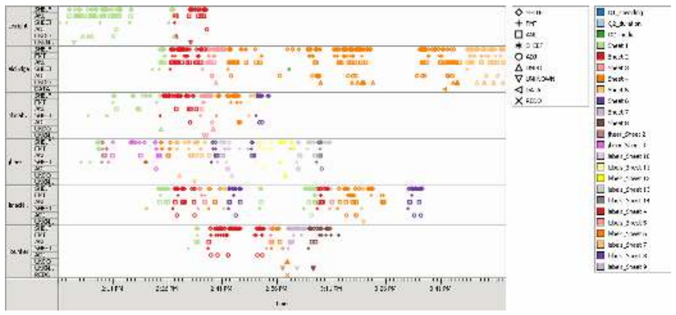


Fig. 10. Aggregate Analysis of Tableau Usage. Each row shows the timeline for a different user. Shapes indicate command types; color indicates worksheet usage. The color patterns indicate different worksheet usage and revisitation patterns across users.

Management Technique	Items culled	% culled
Undo-as-Delete	941	5.4%
Chunking Formatting Actions	4,139	23.8%
Chunking Filter & Sort Actions ($\Delta t \leq 30s$)	1,432	8.2%
Chunking Shelf Actions ($\Delta t \leq 5s$)	4,228	24.3%
Total Items Culled (out of 17,401)	*10,740*	*61.7%*

Table 1. Estimated Reductions from History Management.

4.3.4 Estimated Impact of History Management Techniques

Finally, we have used collected history data to estimate the savings provided by our chunking rules and undo-as-delete. Table 1 shows the number of states culled when applying our techniques to the collected history data, showing that 61.7% of states are either removed or chunked. Thus, we might expect presented histories to be as little as 40% the size they would be without our techniques. We note, however, that this is an estimate from recorded data and as such does not include manual deletion of history items or the effects of bookmarking and annotation.

5 Summary and Future Work

In this paper, we have introduced a design space analysis of history systems and used it to develop a prototype history interface for the Tableau visualization system. Our analysis served as a useful guide for navigating the design decisions we faced while architecting history interfaces to support visual analysis and communication. Our resulting history model integrates history management and undo/redo functionality and provides an editable, graphical history that supports branching analysis histories within worksheets and merged global histories across worksheets. Our graphical history interface allows revisitation of previous views and is designed to complement Tableau's visual analysis features by providing overview displays of visualization states both within and across worksheets.

Our history tools introduce a suite of novel features. Our undo-as-delete feature provides an empirically-motivated mechanism for helping improve the scalability of history displays, while preserving the capability for branching histories. Our search, filter, and annotation features enable users to retrieve previous visualization views based on the data fields involved, the type of chart, and bookmarked status. Selected history items can then be exported in multiple formats, including as presentations in which Tableau visualizations can be edited as native vector graphics.

We have also applied our history model to support evaluation of the Tableau system. Our visual analysis of history logs has inspired multiple improvements to Tableau's user interface and has informed the design of our graphical history tools.

In future work, additional mechanisms for managing history may be of help. For example, our chunking rules are hand-crafted and highly specific to Tableau. Could a more general characterization of analytic tasks be applied to reuse design knowledge across visual analysis tools? Another potentially useful feature would be automated estimates of the saliency (or "importance") of visited views. Such estimates could inform semantic zooming of history displays, chunking, and more automated forms of presentation generation. How should features such as timing, revisitation, and interaction influence such a model?

Future work might further facilitate the creation of presentations from analysis histories. Our current approach enables manual selection of history views (in conjunction with search and filtering) and export of those views into external media. Other visual analysis tools [4, 7, 11] have explored explicit sharing and story-telling features. Novel tools that make use of recorded user histories for structuring presentations or story-telling may prove beneficial.

Not only might exported histories communicate findings, they may help teach analysis by example. Along these lines, analysis histories may also contribute to our understanding of common analysis patterns. Jankun-Kelly et al. developed their parameter-set model of visualization state [13] with this goal in mind. History logs have proven useful for evaluating Tableau usage. A larger-scale analysis of history patterns may better characterize sense-making processes at an operational level and suggest enhanced interface designs for supporting visual analysis.

ACKNOWLEDGEMENTS

We thank our colleagues at Tableau Software, especially Andrew Beers, Matthew Eldridge, Iain Heath, and Pat Hanrahan. Maneesh Agrawala was partially supported by an Alfred P. Sloan Foundation fellowship and NSF grant CCF 0643552.

REFERENCES

[1] Ayers E.Z., Stasko J.T. Using Graphic History in Browsing the World Wide Web. *Proc. World Wide Web*, 1995.

[2] Berlage T.A. Selective Undo Mechanism for Graphical User Interfaces based on Command Objects. *ACM Transactions on Computer-Human Interaction*, **1**(3): 269-294, 1994.

[3] Brodlie K., Brankin L., Poon A., Banecki G., Wright H., Gay A. GRASPARC: a problem solving environment integrating computation and visualization. *Proc. IEEE Visualization*: 102-109, 1993.

[4] Callahan S.P., Freire J., Santos E., Scheidegger C.E., Silva C.T., Vo H.T. Managing the Evolution of Dataflows with VisTrails. *Proc. IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, 2006.

[5] Card S.K., Pirolli P., Van Der Wege M., Morrison J.B., Reeder R.W., Schraedley P.K., Boshart J. Information Scent as a Driver of Web Behavior Graphs: Results of a Protocol Analysis Method for Web Usability. *Proc. ACM CHI*: 498-505, 2001.

[6] Derthick M., Roth S.F. Enhancing Data Exploration with a Branching History of User Operations. *Knowledge Based Systems*, **14**(1-2): 65-74, March 2001.

[7] Eccles R., Kapler T., Harper R., Wright W. Stories in GeoTime. *Proc. IEEE VAST*: 19-26 2007.

[8] Edwards W.K., Igarashi T., LaMarca A., Mynatt E.D. A Temporal Model for Multi-Level Undo and Redo. *Proc. ACM UIST*: 31-40, 2000.

[9] Gamma E., Helm R., Johnson R., Vlissides J. Command, in *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley: 233-242, 1995.

[10] Goodell H., Chiang C., Kelleher C., Baumann A., Grinstein G. Collecting and Harnessing Rich Session Histories. *Proc. Int. Conf. on Information Visualization*: 117-123, 2006.

[11] Heer J., Viégas F.B., Wattenberg M. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. *Proc. ACM CHI*: 1029-1038, 2007.

[12] Hightower R., Ring L., Helfman J., Bederson B., Hollan J.D. Graphical Multiscale Web Histories: A Study of PadPrints, *Proc. ACM Hypertext and Hypermedia*: 58-65, 1998.

[13] Jankun-Kelly T.J., Ma K.-L., Gertz M. A Model and Framework for Visualization Exploration. *IEEE Trans. on Visualization and Comp. Graphics*, **13**(2): 357-369, 2007.

[14] Kaasten S., Greenberg S. Integrating Back, History and Bookmarks in Web Browsers. *Extended Abstracts ACM CHI*: 379-380, 2001.

[15] Kaasten S., Greenberg S., Edwards C. How People Recognize Previously Seen Web Pages from Titles, URLs, and Thumbnails. *Tech Report 2001-692-15*, Department of Computer Science, University of Calgary, Alberta, Canada, 2001.

[16] Klemmer S.R., Thomsen M., Phelps-Goodman E., Lee R., Landay J.A. Where Do Web Sites Come From? Capturing and Interacting with Design History. *Proc. ACM CHI*: 1-8, 2002.

[17] Kurlander D., Feiner S. Editable Graphical Histories. *Proc. IEEE Workshop on Visual Language*: 127-134, 1988.

[18] Kreuseler M., Nocke T., Schumann H. A History Mechanism for Visual Data Mining. *Proc. IEEE InfoVis*: 49-56, 2004.

[19] Lee J.P., Grinstein G.G. An Architecture for Retaining and Analyzing Visual Explorations of Databases. *Proc. IEEE Vis*: 101-108, 1995.

[20] Ma K.-L. Image Graphs: A Novel Interface for Visual Data Exploration. *Proc. IEEE Visualization*: 81-88, 1999.

[21] Mackinlay J.D., Hanrahan P., Stolte C. Show Me: Automatic Presentation for Visual Analysis. *IEEE Trans. on Visualization and Comp. Graphics*, **13**(6): 1137-1144, 2007.

[22] Meng C., Yasue M., Imamiya A., Mao X. Visualizing Histories for Selective Undo and Redo. *Proc. 3rd Asian Pacific Computer and Human Interaction*: 459, 1998.

[23] Myers B.A., Kosbie D.S. Reusable Hierarchical Command Objects. *Proc. ACM CHI*: 260-267, 1996.

[24] Plaisant C., Rose A., Rubloff G., Salter R., Shneiderman B. The Design of History Mechanisms and their Use in Collaborative Educational Simulations. *Proc. CSCL*: 348-359, 1999.

[25] Robinson A.C., Weaver C. Re-Visualization: Interactive Visualization of the Process of Visual Analysis. *Proc. GIScience Workshop on Visual Analytics & Spatial Decision Support*, 2006.

[26] Shipman F.M., Hsieh H. Navigable History: A Reader's View of Writer's Time. *The New Review of Hypermedia and Multimedia*. **6**(1): 147-167, 2000.

[27] Shneiderman, B. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proc. IEEE Visual Languages*: 336-343, 1996

[28] Stolte C., Tang D., Hanrahan P. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Trans. on Visualization and Comp. Graphics*, **8**(1): 52-65, 2002.

[29] Vitter J.S. US&R: A New Framework for Redoing. *IEEE Software*, **1**(4): 39-52, 1984.

[30] Waterson S., Hong J.I., Sohn T., Heer J., Matthews T., Landay J.A. What Did They Do? Understanding Clickstreams with the WebQuilt Visualization System. *Proc. AVI*: 94-102, 2002.

[31] Woodruff A., Faulring A., Rosenholtz R., Morrsion J., Pirolli P. Using Thumbnails to Search the Web. *Proc. ACM CHI*: 198-205, 2001