Graphical Histories of Information Foraging

Manuela Waldner [1] **Stefan Bruckner** [2] **Ivan Viola** [1]

[1] Vienna University of Technology, Vienna, Austria
{waldner|viola}@cg.tuwien.ac.at

[2] University of Bergen, Bergen, Norway
Stefan.Bruckner@uib.no

ABSTRACT

During information foraging, knowledge workers iteratively seek, filter, read, and extract information. When using multiple information sources and different applications for information processing, re-examination of activities for validation of previous decisions or re-discovery of previously used information sources is challenging. In this paper, we present a novel representation of cross-application histories to support recall of past operations and re-discovery of information resources. Our graphical history consists of a cross-scale visualization combining an overview node-link diagram of used desktop resources with nested (animated) snapshot sequences, based on a recording of the visual screen output during the users' desktop work. This representation makes key elements of the users' tasks visually stand out, while exploiting the power of visual memory to recover subtle details of their activities. In a preliminary study, users found our graphical history helpful to recall details of an information foraging task and commented positively on the ability to expand overview nodes into snapshot and video sequences.

Author Keywords

interaction history; graph visualization; provenance.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

INTRODUCTION

Information foraging is an integral part of knowledge workers' sensemaking processes, covering adaptive seeking, filtering, reading, and extracting of information to generate hypotheses or make decisions [28]. Often, these information foraging tasks require multiple specialized tools, auxiliary applications, and different information sources. Examples include investigative journalists synthesizing information from various sources into a coherent newspaper story, public authorities collecting and re-distributing information from and to various channels, or biologists building structural and physiological models by integrating information from online protein databases, pathways, experiments, and scientific publications.

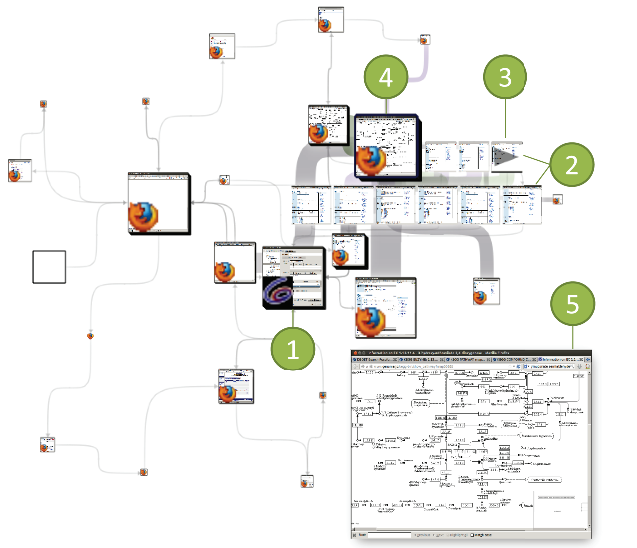


Figure 1. WindowTrails of a biochemical modeling task: information from various online sources is collected and interpreted to define a model of a biochemical reaction in a specialized simulation software (1). The history graph contains a node for each recorded view. Two nodes are expanded to activation sequences (2) and a video sequence (3), respectively. A linked view (5) shows the selected node (4) in full detail.

Whenever these knowledge workers need to re-examine previous steps of their information foraging activities or re-visit information resources, they are facing a serious challenge. While many applications provide operation or browsing histories, these individual histories cannot coherently capture an entire cross-application information foraging task. They either facilitate document re-location based on contextual cues (such as last usage times and related documents [26, 18, 13]), or provide a single-application visual operation history (such as image editors or visualization software [22, 14, 12]).

Our work *WindowTrails* is a first attempt to bridge the gap between browsing histories and operation histories of individual applications through a visual summary of cross-application information foraging. The main challenge we address in this paper is how to visually convey the structure of information flow across multiple information sources and applications, while still being able to explore fine-grained operations in full detail. Our graphical desktop history allows users to explore their task histories on multiple levels of detail in a structural and a temporal approach. We contribute a novel steerable graph-based history, with semantically zoomable node representations to aid recognition of key desktop elements and recall of exact operation details in a nested visualization (Figure 1). Thereby, we exploit the power of the user's visual memory by constructing our graphical history from a recording of visual screen changes and low-level user input. We gathered first user feedback in a preliminary user study by asking users to recall details of an information foraging task after a few days with the aid of either WindowTrails or a list-based history interface.

RELATED WORK

Our work covers aspects of graphical operation and browsing histories, as well as graph visualization. We review the related work from the two distinct subfields in the following subsections.

Graphical Histories and Provenance

Graphical operation histories have been recognized as important tools to review and summarize previous activities [14], to learn and compare workflows [20], to facilitate undo and redo actions during task operation [14, 25, 12], and for estimating task durations [32]. Most commonly, operation histories are represented as lists containing textual descriptions and small thumbnails (*e.g.*, image editors). When exploring potential visual representations of workflows, Kong *et al.* [20] found out that users expect to see snapshots of the results, but also of important intermediate steps. Therefore, several researchers generate continuous operation histories based on a video stream or representative snapshots: For instance, Klemmer *et al.* [19] create a timeline view with thumbnails for each recorded user action. In *Chronicle*, Grossman *et al.* [12] hierarchically aggregate key actions into a limited number of expandable meta-thumbnails, and Ma [22] represents the visual exploration workflow of a scientific visualization in a node-link diagram, where nodes show snapshots of the visualization and links represent operations, like transfer function adjustments. Heer *et al.* [14] also consider undo-redo activities and arrange sequential visualization states in branching timelines. While these examples are all tightly integrated into an application, Nakamura and Igarashi [25] demonstrate a method to construct compact application-independent interaction histories as storyboards. We were inspired by these rich histories and explored possibilities to translate some concepts to cross-application scenarios, where task work is fragmented across multiple resources. We also rely on a screen recording to visually summarize user operations. However, we visually embed the recorded interaction sequences of different application windows into a larger network of applications and information resources that all contributed to the user's task.

In contrast to operation histories, browsing or file histories have been developed to re-discover resources [13], to resume a task after an interruption [26, 30], but also to assess the usability of web sites [16]. These histories typically do not try to convey what the user has done, but rather which resources have been used, which have been used in concert, and at which times. Prominent examples are conventional web browsers that represent their history as a time-ordered list of web site titles. *YouPivot* [13] augments such web histories with contextual cues, such as public calendar data or the user's twitter activities, and a stream graph visualization to highlight times of high user activities in different online resources. *WebComets* [9] uses a timeline-based approach, where individual web page visits are horizontally arranged according to their visit time and duration, and vertically clustered by browser tabs. The *Context-Aware Activity Display* [30] represents groups of related documents in bubbles, where groups are automatically defined based on the user's interaction history. None of these browsing histories assists users in recalling their actions in detail by visually summarizing what they actually have seen. In contrast, Ayers and Stasko [2] represent web histories as trees, where nodes can be expanded to thumbnail previews of the web sites. Similarly, *WebQuilt* [16] uses a zoomable network of website thumbnails to visualize user traces through web pages. While single screenshots of web pages may support recall by visual memory, we believe that a much more detailed operation recall can be facilitated if users are presented with what they actually have seen in a window (*e.g.*, where they have scrolled) and what they did in their applications (*e.g.*, what text they entered). Our work therefore differs, as we try to reconstruct multiple meaningful snapshots of the users' work process, and allow them to watch short interaction sequences on demand.

A common way to present file system and document provenance is a graph structure. In the *Open Provenance Model* [24], a directed provenance graph is specified to express causal relationships and dependencies between "things". For instance, the *VisTrails* graph [7] illustrates the dataflow to generate a scientific visualization. The document provenance graph by Jensen *et al.* [18] visualizes how information is transported through files, such as by copy-paste, save-as, or e-mail attachment operations. *Orbiter* [23] and *InProv* [5] show reads and writes of processes and files in a semantically zoomable node-link diagram and as a radial tree, respectively. *Footprints* [35] and *WebQuilt* [16] visualize users' web site traffic in node-link diagrams. A problem with these graph-based approaches is that the temporal aspect is lost. Therefore, Schmidt *et al.* [32] presented a compound graph of history elements grouped into temporal sub-sequences. Our work extends these graph-based approaches by visualizing a cross-scale graph, where users can selectively expand nodes to an animated history sequence.

Compound, Time-Dependent, and Nested Graphs

A compound (di)graph is a graph sharing its vertices with a rooted tree [33, 34]. A subgroup of compound graphs are hierarchical graphs, where nodes are repeatedly aggregated into subgraphs in meta-nodes [34]. In a graph hierarchy, only the leaf nodes of the hierarchy define the input graph, and all subgraphs must be connected [1]. Examples for graph hierarchies using node-link diagrams are *GrouseFlocks* [1] and *Orbiter* [23]. *TreeNetViz* [11] and *InProv* [5] display graph hierarchies in radial layouts. These systems are steerable, meaning that users can interactively generate cross-scale networks, visualizing nodes from different hierarchy levels at the same time [11]. All of these examples use homogeneous node representations across all scales. Our cross-scale node-link diagram is not only steerable with respect to the graph layout. It also provides the user with semantic zooming on the node representation with each hierarchy level change – from abstract application icons and thumbnails to short recorded interaction sequences.

Graph representations usually do not convey a temporal sequence of events. Card *et al.* [8] therefore use a timeline to control which time span is visualized in their *TimeTree*. Similarly, users can analyze node-link diagrams of player moves and ball passes of temporal soccer game phases in *SoccerStories* [27]. Others combine structural and temporal data into a single static compound graph, such as *TimeRadarTree* [6]. In our WindowTrails system, users are provided with different interaction techniques to explore the temporal aspect of the graph. In addition, users can expand nodes to reveal *nested* [17, 15] time-ordered sequences of recorded window states.

WINDOW TRAILS

Human memory is an adaptive system, which makes information that is no longer needed gradually less accessible [31]. Consider, for instance, a bioinformatician trying to model a biochemical reaction. She first browses various online resources, such as databases of biochemical pathways, enzymes, functional genomics experiments, and biomedical literature. As she extracts evidence from these resources, she gradually refines her model in a biochemical simulation software. A few months later, she wants to adapt her model but can no longer recall in full detail the information sources and operation steps she took that led to her model design. Another more trivial information foraging task could involve a person planning a journey for two travellers. He browses different resources to come up with an initial travel plan. As he outlines this travel plan to his co-traveller, they may disagree on a few aspects, such as the proposed flight time or the selected hotel. Being able to recall alternative options that were initially discarded, such as a cheaper hotel, could help to apply corrections to the initial travel plan. We simulated such a scenario in our preliminary user study.

Operation and browsing histories should facilitate recall of task details after a longer time, when the memory of the task has already vanished. However, the problem with exhaustive digital histories is that they make it hard to distinguish relevant from irrelevant material [3]. So how can we support users in discovering one single piece of information or a single operation in a history covering an entire task spanning over hours? We formulated three core requirements for our graphical histories reflecting these challenges:
R1: Aid recognition of key elements.
R2: Support re-tracing of activities from these key elements.
R3: Allow parts of the history to be explored in full detail.

We designed a couple of early draft visualizations for histories of desktop resource usage, from simple time-dependent lists to complex information flow visualizations similar to Sankey diagrams. The well-known node-link diagram thereby was the most compact and at the same time the most easily understandable representation, according to informal user feedback. We therefore chose a node-link diagram as initial history representation to provide an overview of history elements and their relations. Node-link diagrams are known to be well suited for exploration of relatively small graphs and for path-finding tasks [10]. They are a ubiquitous choice to describe complex time-dependent systems, such as state transition diagrams [29], scene transition graphs for videos [36], sports activities [27], provenance [24, 7, 18, 22], UML activity diagrams, or biochemical pathways. In a node-link diagram of an information foraging task, relevant "information sinks", where collected information was processed and synthesized, should be easily detectable, while related "information sources" should be visually connected to these sinks.

However, classic node-link diagrams do not convey any temporal information and also do not summarize what the user did or has seen. Therefore, we nest linear snapshot sequences into these nodes that can be expanded on demand. We provide nodes in multiple levels of detail, so users can gradually drill down from an abstract node representation to short history sequences in full temporal resolution.

Our WindowTrails system is composed of two core components, as described in more detail in the following subsections: 1) a recording component, capturing the visual desktop history of the user and 2) the steerable visualization of this information foraging history with three levels of detail.

Data Collection

Our graphical information foraging history is based on a comprehensive recording of low-level input events and visual screen changes during the user's information foraging work. The recording can be invoked on demand whenever the user performs some knowledge-intensive work. The following data is recorded:

We keep track of all windows that have been in use during the user's task. As single windows are often used to show different documents or information sources, we furthermore distinguish between different *views* of a window. We define a *view* as a distinct content (*e.g.*, document or web page) shown in a window. This content-centric approach is a common granularity used for distinguishing distinct history items in existing file and browsing histories, such as web browsers. In our data structure (Figure 2 middle), views represent Level 1 nodes. In the history visualization, these nodes are the initial collapsed nodes of the node-link diagram at start-up (Figure 3 left).

Whenever the user performs a focus switch (*i.e.*, a new window receives the input focus) or the active document within a window is switched, we save a focus switch event. A focus switch event is described by the following tuple: (v_{prev}, v_{curr}, t), where v_{prev} and v_{curr} denote the previously and currently active view, respectively, and t is a time stamp. In the graph, focus switch events represent the edges between nodes (Figure 2 right), and each tuple also corresponds to one Level 2 node of v_{curr} (Figure 2 middle).

We furthermore store all visual updates on the screen as the following tuple: $(v, type, rect, img, t)$, where v is the currently active view, $type$ describes the type of the update (full

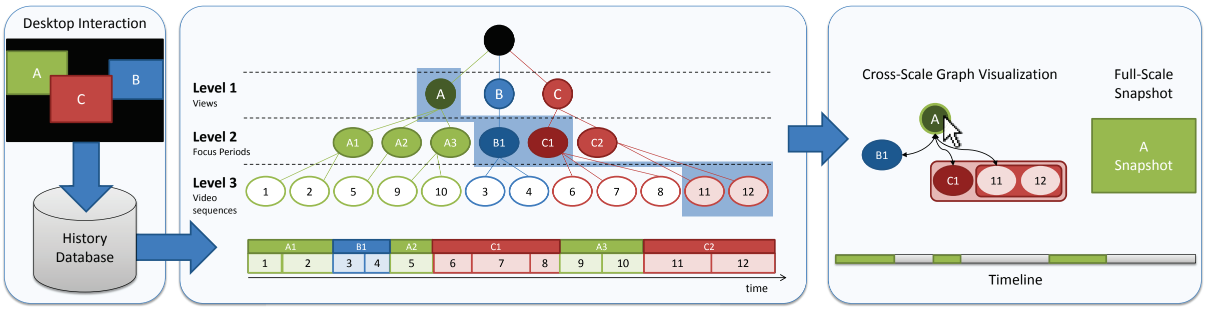


Figure 2. System overview, illustrated with three example windows and an exemplary user selection in the resulting cross-scale graph: (left) desktop interaction with three application windows (A, B, C), recorded into the history database; (middle) the temporal sequence of the recorded events (numbers in timeline at the bottom), the hierarchical data structure (top) with exemplary user-defined cut through three hierarchy levels (blue background), and (right) the resulting cross-scale graph with a linked full-scale view and timeline. In this example, the user expanded the nodes B, C, and C2.

window, key-frame, or partial update), $rect$ is the visual update's position and extents, img is the link to the actual image data, and t is the event's time stamp. Key-frames are partial updates covering a fraction of the active window's size larger than a pre-defined threshold. Each key frame is a Level 3 node (Figure 2 middle) and is represented as a fully expanded animated node in the history visualization (Figure 3 right).

Finally, we capture low-level input events as $(v, type, t)$, with v denoting the active view, $type$ distinguishing between a mouse button press, a key press, and a scroll wheel input, and t as time stamp. This information is used to determine the nodes' relative sizes in the node-link diagram, as described in the next subsection.

Cross-Scale Graph Visualization

Based on the recorded data, we construct a tree structure (*cf.*, Figure 2 middle). While the second and third hierarchy levels represent temporal sequences, the first level corresponds to a network of views. We present the collected data in a compound directed graph, which is described in more detail below.

Level 1: View Network — The first level of the tree defines the base network, with which the visualization is initialized when the user loads a history. It is visualized as node-link diagram of views, with distinct views as nodes and focus switches between views as directed links (Figure 3 left). Each link is associated with a focus switch time. A node's incoming and outgoing links with the smallest and largest time values, respectively, define the node's time interval $[t_s, t_e]$. The opacity of the links encodes the number of the switches between two adjacent views.

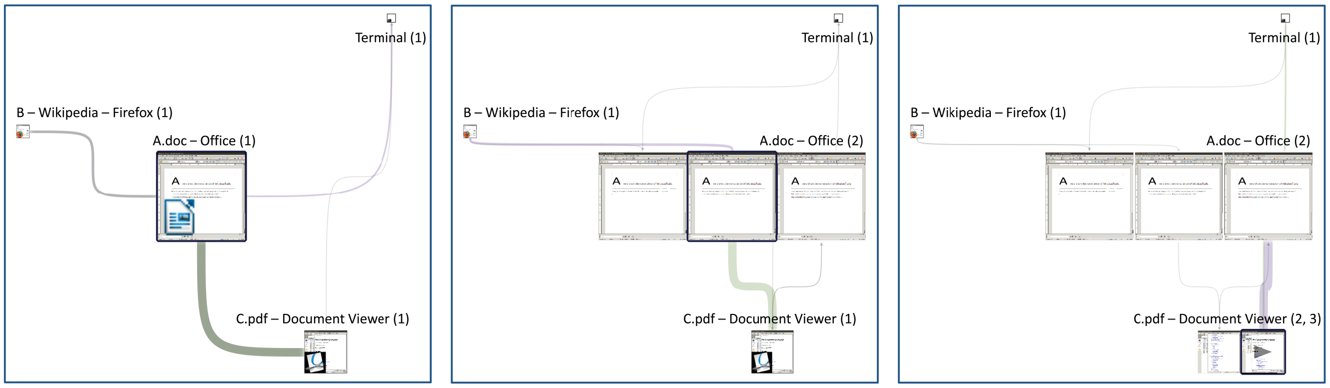
To make intensively-used information sinks visually distinctive, nodes are scaled according to the amount of user activity associated. For calculating the node size, we count the number of recorded low-level input events associated with the node's view and weigh them according to their event type. We weigh scroll wheel events lower than mouse clicks and key presses, so views where information was actively processed (*e.g.*, by typing text) clearly stand out. A force-directed graph layout assures that frequently visited views are located centrally, and that views being used together are placed in close proximity.

In an early version of the visualization, Level 1 nodes were represented as application icons. However, collected data from real scenarios and feedback from users (see Preliminary User Study) suggested that information collection tasks often lead to a large number of views associated with the same application and application icon, respectively. This was primarily the case for the web browser, where each loaded web site was shown as a separate node with the same icon in the node-link diagram. We therefore changed this representation for large views in the final version of our visualization: Above a given node size threshold, Level 1 nodes are represented by a combination of icon and thumbnail. The thumbnail thereby shows the last recorded snapshot of the view in the history.

Consider the simplistic example in Figure 3 left: While the web browser window "B" and the PDF viewer "C" were used to collect information, the user entered the retrieved information into the word processing document "A". As the user switched back and forth from the word document to the two information sources, node "A" is placed in the center, with links to the related elements. The increased activity in the word processor leads to a clearly distinct node size of "A". While this network fulfills our first requirement **R1** to guide the user's attention to the most actively used and closely related history items, it is not possible to determine which activities have been performed in more detail.

Level 2: Storyboard of Application States — Level 1 nodes can be expanded to a linear sequence of window snapshots to show a temporal sequence of application states. As application-independent histories cannot access any semantic task information to segment the history into meaningful portions, we provide one snapshot for each time a view was in focus (Figure 3 middle). Each snapshot node has one incoming and one outgoing link to a different view, defined by the recorded focus switch events. The two links define the node's focus time interval $[t_s, t_e]$. The snapshot sequence substitutes the icon+thumbnail representation in the initial graph layout. Neighboring Level 1 nodes can be minimally shifted to prevent collisions, and will return to their original locations when the node is collapsed again.

Node thumbnails show the last recorded visual state of the view in the node's focus time interval. Thus, an expanded



Terminal (1)
B – Wikipedia – Firefox (1)
A.doc – Office (1)
C.pdf – Document Viewer (1)
Terminal (1)
B – Wikipedia – Firefox (1)
A.doc – Office (2)
C.pdf – Document Viewer (1)
Terminal (1)
B – Wikipedia – Firefox (1)
A.doc – Office (2)
C.pdf – Document Viewer (2, 3)

Figure 3. Cross-scale graph visualization for four example views A, B, C, and a terminal window: (left) the initial Level 1 network; (middle) A expanded to Level 2; (right) C2 expanded to Level 3. Labels were added for illustration purposes, with the current node level in brackets.

snapshot sequence of a view represents a visual summary of discrete operation steps in an application – *i.e.*, of what the user has seen while performing the task. The snapshot sequence should help the user identify key actions from which related items can be re-traced by following incoming and outgoing links ($\rightarrow$ **R2**). In the simple example of Figure 3 middle, the user can observe the changes in document "A" for each focus period, and can also follow the links to the information sources visited before and after the selected focus period.

Level 3: Video Sequences — If the user was producing a long piece of work in a single view without consulting other information sources, a single Level 2 snapshot may represent a long operation period. Therefore, users can expand individual snapshots one level further, splitting the period into shorter sequences for a fully detailed exploration. We aggregate all visual update events associated with the view's focus period into multiple sequences with a pre-defined maximum length and at least a single key-frame within the period. Changes between two subsequent nodes are depicted by animated snapshots. Animated Level 3 nodes are overlaid by a transparent arrow and a small progress bar below the node to clearly indicate that the full operation history of a particular time period is replayed.

In the example of Figure 3 right, the user expanded the second Level 2 node of PDF document "C" to determine which parts of the document had been read. Since the recorded interaction sequence is very short, the Level 3 sub-graph consists only of a single node in this example. This fulfills our final requirement **R3**: Once identified, items of interest can be explored in full detail, by watching short video sequences of recorded user operations.

Interactive History Exploration

WindowTrails are not meant to be static. In contrast, interactively drilling down from the overview network to real-time interaction sequences is a core concept of our graphical history. The compound graph representation thereby provides structural access to history items. The detailed exploration is happening through multiple linked views. In addition, we provide interaction techniques to explore the history temporally.

Semantic zooming is the core concept of our interactive history visualization. Starting from the Level 1 network, users interactively create cross-sections through the graph hierarchy by left-clicking on nodes, as depicted in Figure 3. To collapse expanded nodes, they right-click on any child node.

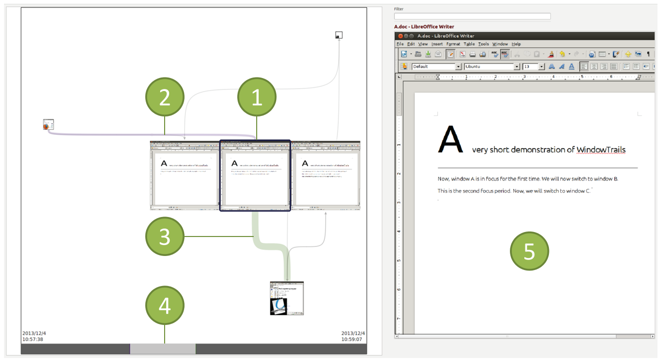
In addition, users can **filter** the graph representation by selecting a focus node with the middle mouse button. The transparency α_n of a node n is then defined by its shortest distance d_n (*i.e.*, number of links in the shortest connecting path) to any focus node in the graph:

$$\alpha_n = \alpha_0 + \frac{(1-\alpha_0)}{d_{max}}(d_{max} - d_n + 1),$$

where α_0 is the minimum node opacity, d_n is the node's distance to the nearest focus node, and d_{max} is the maximum distance of a node to a focus node to be displayed. In practice, we chose $\alpha_0 = 0.2$ (with $\alpha_0 \in [0,1]$) and $d_{max} = 1$. This means that all nodes that do not directly link to a focus node will be shown with minimum opacity. Multiple nodes can be put into focus to emphasize certain operations. Focus nodes are rendered with red borders.

Users can explore **details-on-demand in linked views** by hovering over a node with the mouse. The incoming and outgoing edges of a hovered node are highlighted in semi-transparent purple and green, respectively, and bidirectional connections merge to gray. At the bottom, a timeline indicates the focus periods of the selected node. On the right, a linked view shows a full-scale snapshot associated with the node. Figure 4 is a screenshot of our linked views, showing details of node A2.

Finally, we allow users to **trace interaction trails** in the history in their temporal sequence. Consider Figure 5, where it is impossible to tell which views were visited between the two focus periods of the PDF document C without expanding node A. To trace the visited views after the first visit of C, the user hovers the left-most node of C to select it. After this selection, we set the current time step t of the exploration to the node's start time t_s. When the user presses the right arrow key, we query the selected node's outgoing link with time t_l, which results in the minimum $\Delta t > 0$, where $\Delta t = t_l - t$. The new highlighted node is the outgoing link's adjacent node


1
2
3
4
5
2013/12/4 10:57:38
2013/12/4 10:59:07
Filter
A.doc - LibreOffice Writer
File Edit View Insert Format Table Tools Window Help
Default
Ubuntu
13

A very short demonstration of WindowTrails

Now, window A is in focus for the first time. We will now switch to window B.

This is the second focus period. Now, we will switch to window C.

Figure 4. Linked views to inspect details on demand: (1) the highlighted node, (2) the incoming edge in purple, (3) the outgoing edge in green, (4) the focus period in the timeline, and (5) the full-scale snapshot view.

(for instance, A in Figure 5 left), and t is set to t_l. Similarly, when stepping in reverse direction (by pressing the left arrow key), we query the selected node's incoming link with minimum $\Delta t = t - t_l$, where t is initially set to the selected node's end time t_e.

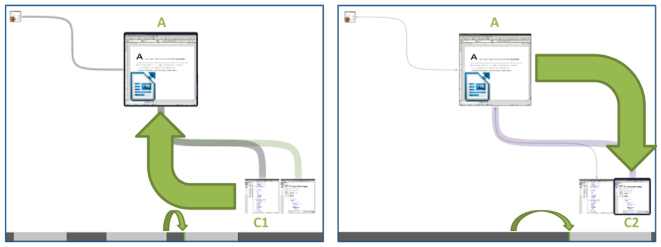


Figure 5. Re-tracing steps from C1 to C2 via the collapsed node A (large green arrows). In the timeline, the time steps are shown (small green arrows).

IMPLEMENTATION

Our WindowTrails implementation consists of two pieces of software: a prototypical recording tool on the window manager level and the interactive graphical history. Our proof-of-concept history recorder is implemented as a plug-in to a wide-spread OpenGL-based window manager for the *X Window System* (*Compiz*). When activated, the plug-in listens to window mapping and unmapping events, focus switches (*i.e.*, when a window receives the input focus and is restacked to the top), and window manipulation events (moving and resizing). For each newly created window, we store the application icon, the application name, and the command line argument that initiated the window's process. The latter is not used in our current implementation, but could be utilized to re-initiate the window's process from the history interface. As it is not possible to directly infer the windows' loaded documents or URLs on the X Window System level, we additionally keep track of the window titles to distinguish between distinct views of a window. User activity is captured through the X Window System's *XInput* extension.

We utilize the X Window System's *XDamage* extension to be notified about "damaged" rectangular window regions that require repaint. These regions are read back from the framebuffer through *glReadPixels*. Only if a window is mapped, moved, or resized, we make a full window snapshot.

From these recorded damage regions, a snapshot of a view v associated with a window w at a particular time t can be reconstructed as follows: We first retrieve the last full visual update of w before the requested time t, and load its associated image to a texture tex_{base}. Then, we query the time associated with the last key-frame update t_{kf} of v before t. Finally, we gather all visual updates associated with v between t_{kf} and t and copy them into tex_{base}. In some occasions, this leads to visual artifacts, when the key-frame at t_{kf} did not properly cover all previous window content. In our examples, we chose a key-frame threshold of 0.5 – meaning if more than half of the window's size is repainted, it will be treated as a key-frame. The lower this threshold, the fewer images have to be accumulated into tex_{base} because more recorded updates will be labeled as key frames, but the more likely it is that snapshots will be assembled incorrectly.

For animated Level 3 nodes, with a given start time t_s and end time t_e, we create one static texture for the node at t_s, as described above. Then, for each time step $t_i \leq t_e$, we create a transparent texture tex_i with the dimensions of tex_{base} and copy all visual updates between t_i and t_{i+1} into tex_i, where i is the number of seconds passed since t_s. We create a Level 3 node for every 10 seconds containing at least one key-frame.

The recorder writes all events to an SQLite database. Recorded update regions are stored as PNG images to the hard drive in their original dimensions.

This proof-of-concept recorder is sufficient to generate data to demonstrate our interactive history visualization. However, for long-term employment, several technical tweaks are recommended to improve the performance and applicability. In particular, our recording tool generates an extensive amount of image data by capturing window update regions in full resolution (*e.g.*, around 1 GB per 30 minutes in our user study). Down-scaling these images and image-based comparisons to previously captured frames could significantly decrease the amount of captured image data. Of course, with every down-scaling, it will be harder for the user to identify detailed information in the reconstructed snapshots. Another useful extension could be the incorporation of *inotify* to additionally observe file system changes, so nodes in the visualization can actually link to their associated files and processes, respectively. A complementary approach to restore entire system states could be a combination with *DejaView* [21].

The graphical history itself is implemented as a stand-alone *Qt*-application. It generates the compound graph recursively, by first calculating the force-directed layout of all distinct views in the history (using *vtk*), and then adding linear snapshot sequences of expanded nodes. If snapshot sequences exceed the visualization width, they are scaled down. We use the *Box2D* physics library to resolve occlusions between expanded nodes and collapsed network nodes in a post-processing step. Alternatively, a constraint-based graph layout could be used to preserve the initial graph topology.

USAGE SCENARIOS

To illustrate the capabilities of WindowTrails, we will revisit the scenario of the bioinformatician attempting to re-trace her biochemical modeling steps:

The bioinformatician loads the history she recorded while designing the model and quickly spots the icon of her main simulation software popping out from the visualization (Figure 1-1). In addition, she clearly sees the most frequently used tabs of her web browser, where initially those with direct connections to her simulation software are most relevant for her. She expands the simulation software node and hovers over the snapshots to quickly review her operation steps. She identifies a snapshot created when she was entering an important reaction formula and selects it as focus element to filter the items visited directly before and after (Figure 6-1). From this snapshot, she follows the incoming trail to a browser tab that was frequently accessed in her modeling task, showing information on the reaction she had entered. To determine where she found this reaction information, she expands the node and selects the first snapshot in the sequence as another focus element (Figure 6-2). Now, the origin browser tab (an online pathway visualization) pops into focus as well (Figure 6-3), because it was visited directly before. Since this tab was activated many times and links to a lot of different nodes, she traces her steps back from the reaction formula using the arrow key until she reaches a node showing a pathway search interface in the third re-tracing step. In the linked detail preview, she can easily identify the search term she used to come up with the reaction she was looking for. Since the timeline indicates that the search window tab has been activated twice, she expands the node, focuses on the second snapshot (Figure 6-4), and follows the outgoing link from the other snapshot of the search interface. Surprisingly, the search result leads to the same resulting pathway. With these findings, she can more precisely recall the reactions and pathways she browsed and studied in detail to come up with her initial model design.

As illustrated in this scenario, we envision our graphical histories to be primarily an aid for knowledge workers to re-examine their information foraging tasks. However, there are also other application areas where WindowTrails could be beneficial. For instance, it could be used to create interactively explorable cross-application tutorials, in contrast to the wide-spread video tutorials published on the web. Similarly, intelligence analysts could use WindowTrails to share and discuss their cross-application workflows with collaborators. Human-computer interaction designers could evaluate desktop field studies with WindowTrails, and evaluate how the operation of a single application is embedded into a user's normal desktop routine.

PRELIMINARY USER STUDY

We conducted a preliminary user study with an early version of WindowTrails to explore whether and to which extent visual histories help users recalling their information foraging activities, and to detect potential room for improvement. For that purpose, we invited seven paid regular computer and internet users (aged 26 to 36, 2 females, 4 computer scientists

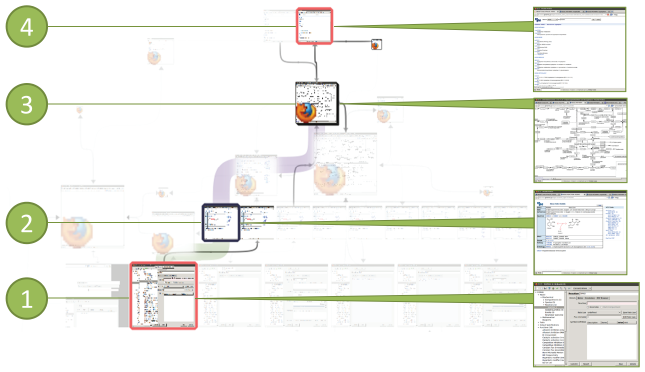


Figure 6. Snapshot of the bioinformatician's exploration steps with enlarged thumbnails on the right: (1) the key activity spotted in the simulation software, (2) the first instantiation of the web site explaining on the key reaction, (3) the pathway visualization containing a link to the reaction, and (4) the search performed to come up with this pathway.

and 3 technical support workers) to perform a study consisting of two parts:

In the first part, users were asked to plan a realistic journey for themselves and a second person fulfilling a few given constraints. They could use any online resources and a few supportive applications. The travel plan describing all transports, *etc.* had to be entered into a text document. We stored all user activities with our recording tool and the experimenter also noted the users' steps manually. We restricted the maximum task time to 30 minutes. One user had to be dropped due to a system failure during recording.

The second part of the study was scheduled three to six days after the first part. In this second part, users had to answer two types of questions: 1) corner stones of their final travel plan (*e.g.*, the airline company of the flight) and 2) details about discarded alternatives (*e.g.*, the name of the bus company of an airport shuttle they had considered but discarded). The second set of questions was highly dependent on the first study part and was prepared individually for each user. We also interviewed users for the reasons why certain options had been chosen or dropped (*e.g.*, reasons why an alternative flight connection they had obtained was finally discarded). They were provided with the print-out of the travel plan created in the first study part, as well as a graphical history.

We employed a between-subjects design for this second part, assigning participants to one of two groups. For the first group, we used an early version of WindowTrails to represent their interaction history. The control group was presented with their recorded interaction history as a temporally sorted, scrollable list of window titles that were color-coded by application type and linked to a timeline on the bottom. Since there is no directly comparable visual history that supports in-depth cross-application task exploration, we chose this simplistic baseline condition to get a first impression of the added value of WindowTrails compared to a classic list-based browsing history. Our hypothesis was that a recorded history is of little help to recall task details when presented as a simple list. With WindowTrails, however, we expected to see that users would be able to recall many forgotten details.

We recorded the user activities by screen capturing and audio recording and issued a questionnaire after the experiment (Table 1). Both study parts were conducted on a workstation with a 27-inch monitor, a quad-core 1.90GHz CPU, and 8GB RAM.

Q1	The provided history was very helpful to remind me about details.
Q2	I consulted the history very often when I could not remember details.
Q3	The layout of history elements was very easy to understand.
Q4	In the history, I could find very quickly what I was looking for.

Table 1. Experiment questions (5-point Likert scale).

Results

We report results on the frequency of history usage to answer questions that could not be fully recalled, and qualitative feedback on the usability of WindowTrails.

History Usage. All except two users could recall all details of their final travel plan, as well as their considerations when making the final decisions, without any digital support. However, users were unable to recall many details of discarded alternatives without any history support. In the control group, the users rarely consulted the history list when being unsure about details. In contrast, users of the WindowTrails group always utilized the visualization when in doubt (usage frequencies are visualized in Figure 7 right, and a resulting visualization by one user is shown in Figure 8). This is also reflected in the questionnaire results: List-users found their history representation less helpful than the WindowTrails-users and also consulted it less frequently according to their subjective judgment (*cf.*, Figure 7 left, Q1 and Q2). After consulting the histories, 75% (WindowTrails group) and 66.7% (control group), respectively, of the final answers were correct. Feedback of WindowTrails users indicates that the snapshots of the views (*"when you see the pictures again, when you look at it: oh yes, there was this hotel again, and that hotel..."*), as well as the compact representation of the entire history in the node-link diagram (*"here you have everything in one place!"*) lead to the increased helpfulness, compared to the list.

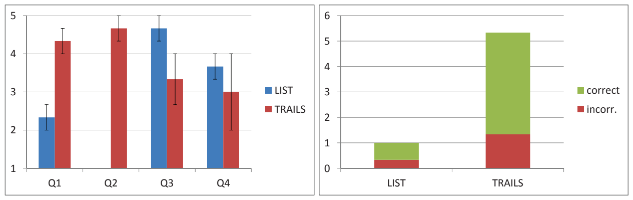


Figure 7. Left: questionnaire results (blue=list, red=WindowTrails) for the questions in Table 1. Right: Average number of times the list (left bar) or WindowTrails (right bar) has been invoked, and whether the resulting answer was correct (green) or incorrect (red).

History Usability. The list-based representation achieved higher scores for understandability and the ease of finding specific items in the questionnaire (*cf.*, Figure 7 left, Q3 and Q4). To better understand these differences, we report selected qualitative feedback. Two WindowTrails users expressed subjective difficulties understanding the initial graph layout. One user explained that he had difficulties finding specific items because *"you see so much... too much information at the same time"*. The other one complained that *"I don't know why the things are where they are"*. Mind, however, that the version of WindowTrails used for the evaluation employed solely application icons to show nodes in the initial Level 1 layout. According to their feedback, these two users expected a more time-centric approach to explore their history with a more powerful timeline to visit the recorded views in their temporal sequence. The third user, however, explicitly appreciated the graph layout because *"[a] linear [layout] would not be good, because you would constantly have to scroll back and forth."*.

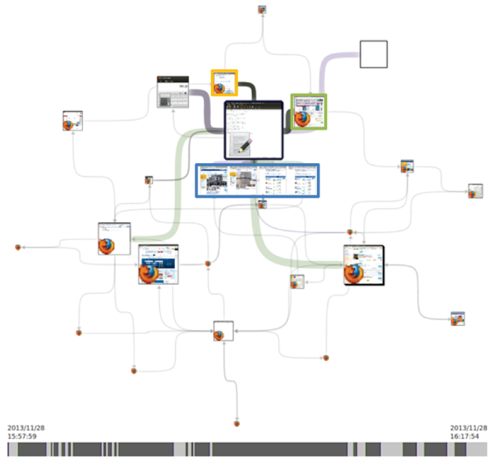


Figure 8. Resulting visualization of one WindowTrails user in our study. The nodes of the websites where she discovered her final flights, hotels, and train connections between the two given destinations are highlighted in green, blue, and yellow. The text document containing the final travel plan is selected, and the hotel website node is expanded.

All three WindowTrails users commented positively on the fact that nodes could be expanded into a linear sequence. One user formulated it as *"because then you have a documentation... you have a linear history, which maps back into this nested history"*. One user utilized animated Level 3 snapshots to determine detailed information and also appreciated the concept: *"This is good, that I don't have to watch the entire video [of the whole recorded history]"*.

Discussion and Improvements

This preliminary study suggests that the proposed WindowTrails history can be more valuable than a simple list of visited desktop resources. This is indicated by our observations that WindowTrails users always consulted the visualization when they were unsure about an answer, and also rated the helpfulness of the history higher than users in the list group.

However, the study also revealed a usability problem of WindowTrails: Two of three users reported that they found the initial graph layout hard to understand. This was surprising for us, since graph layouts are widely used for describing complex time-dependent systems and the node-link diagram was also the most easily understandable representation in early draft stages of our system, according to informal user feedback. From the two users' feedback recorded during the experiment, it seems that they both expected a more time-oriented approach to explore their personal history. On the other hand, the node-link diagram was also praised for its compactness, which might have contributed to the fact that it was used more often than the list. Also, our proposed concept of expanding nodes into linear snapshot sequences was immediately understood by all three users and well received.

In our final design of WindowTrails, we therefore applied two changes to alleviate the raised usability problem without eliminating the initial graph layout: First, we used a combination of window snapshot and application icon for large nodes. While this increases the criticized information density of the initial graph layout even more, it makes it easier to recognize previously visited items on first glance. Second, we added a standard focus+context technique to temporarily filter the visualization based on selected nodes (*cf.*, Figure 6). In an informal follow-up test, the focus+context technique was a frequently utilized feature. Also, the adapted Level 1 representation was found advantageous for identifying key elements, compared to the icon-based node-link diagram in the user study. In the future, another useful extension could be bidirectional brushing and linking between the timeline and the graph visualization. While this is a well-known standard approach (*cf.*, for instance, [8, 13]), it could be a valuable add-on for those users who rather want to explore their interaction history – or parts of it – in a linear, time-based fashion.

The recorded histories for our usage examples and the preliminary user study are shorter than one hour. While two study users even found that this short history sequence leads to a cluttered visualization, longer interaction histories will result in an even larger number of views and nodes, respectively. For longer histories, we will therefore investigate an additional simplification level to maintain a compact representation, for instance by a task-based window clustering as proposed by Oliver *et al.* [26], in the future.

Our study has also shown that the amount of image data recorded by our prototype is currently quite high. As outlined in the implementation section, down-scaling of recorded images, recording only smaller sub-regions containing visual changes, and capturing updates in a lower temporal resolution, can decrease the required storage. Also, discarding the animated Level 3 video sequences and instead merging visual updates into a single static snapshot for each key frame, such as proposed by Bezerianos *et al.* [4] or Nakamura and Igarashi [25], further reduces the amount of stored images.

CONCLUSIONS AND FUTURE WORK

We presented WindowTrails to bridge the gap between single-application operation histories and browsing histories in an integrated cross-application desktop history. It utilizes a steerable compound graph to provide a compact base representation of used desktop resources with nested snapshot and video sequences to re-trace discrete operation steps and to link individual operation histories across applications. User feedback from a preliminary study suggests that both, recall through an image-based representation of history elements and the compact representation of the entire task history in a single diagram, are useful when recalling details of an information foraging task. Since some users found the initial graph view unclear and expected a more time-centric exploration, we added a focus+context technique to temporally filter the node-link diagram by selecting nodes of interest. In the future, classic bidirectional brushing and linking between the timeline and the graph visualization may furthermore support a time-based exploration.

The design of WindowTrails is intended for graphical histories of information foraging tasks, typically ranging in time spans of hours. The next step will be to integrate such a system into the user's real desktop environment. With constant recording, it will be crucial to reduce the amount of recorded image data and to apply visual clustering on the graph visualization to ensure compact representations even for days, months, and years of recorded histories.

ACKNOWLEDGMENTS

This research has been financed by the Vienna Science and Technology Fund (WWTF) through project VRG11-010, and additional support has been provided by the EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680. We thank our user study participants and Anne-Kristin Stavrum for providing us with example scenarios.

REFERENCES

1. Archambault, D., Munzner, T., and Auber, D. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 900–913.

2. Ayers, E. Z., and Stasko, J. T. Using graphic history in browsing the world wide web. Technical Report GIT-GVU-95-12, Georgia Institute of Technology, 1995.

3. Benjamin, A. S. Memory is more than just remembering: Strategic control of encoding, accessing memory, and making decisions. In *Psychology of Learning and Motivation*, Aaron S. Benjamin and Brian H. Ross, Ed., vol. Volume 48 of *Skill and Strategy in Memory Use*. Academic Press, 2007, 175–223.

4. Bezerianos, A., Dragicevic, P., and Balakrishnan, R. Mnemonic rendering: an image-based approach for exposing hidden changes in dynamic displays. In *Proc. UIST 2006*, ACM (2006), 159168.

5. Borkin, M. A., Yeh, C. S., Boyd, M., Macko, P., Gajos, K. Z., Seltzer, M., and Pfister, H. Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2476–2485.

6. Burch, M., and Diehl, S. TimeRadarTrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum 27*, 3 (2008), 823–830.

7. Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. VisTrails: visualization meets data management. In *Proc. SIGMOD 2006*, ACM (2006), 745–747.

8. Card, S., Suh, B., Pendleton, B., Heer, J., and Bodnar, J. Time tree: Exploring time changing hierarchies. In *IEEE Symposium on Visual Analytics Science And Technology* (2006), 3–10.

9. Cernea, D., Truderung, I., Kerren, A., and Ebert, A. WebComets: A tab-oriented approach for browser history visualization. In *Proc. IVAPP 2013*, SciTePress - Science and and Technology Publications (2013), 439–450.

10. Ghoniem, M., Fekete, J., and Castagliola, P. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Symposium on Information Visualization, 2004.* (2004), 17–24.

11. Gou, L., and Zhang, X. TreeNetViz: Revealing patterns of networks over tree structures. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2449–2458.

12. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: capture, exploration, and playback of document workflow histories. In *Proc. UIST 2010*, ACM (2010), 143–152.

13. Hailpern, J., Jitkoff, N., Warr, A., Karahalios, K., Sesek, R., and Shkrob, N. YouPivot: improving recall with contextual search. In *Proc. CHI 2011*, ACM (2011), 1521–1530.

14. Heer, J., Mackinlay, J., Stolte, C., and Agrawala, M. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Dec. 2008), 1189 –1196.

15. Henry, N., Fekete, J., and McGuffin, M. NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1302–1309.

16. Hong, J. I., and Landay, J. A. WebQuilt: a framework for capturing and visualizing the web experience. In *Proc. WWW 2001*, ACM (2001), 717–724.

17. Javed, W., and Elmqvist, N. Exploring the design space of composite visualization. In *Proc. PacificVis 2012*, IEEE (2012), 1 –8.

18. Jensen, C., Lonsdale, H., Wynn, E., Cao, J., Slater, M., and Dietterich, T. G. The life and times of files and information: a study of desktop provenance. In *Proc. CHI 2010*, ACM (2010), 767–776.

19. Klemmer, S. R., Thomsen, M., Phelps-Goodman, E., Lee, R., and Landay, J. A. Where do web sites come from?: Capturing and interacting with design history. In *Proc. CHI 2002*, ACM (2002), 1–8.

20. Kong, N., Grossman, T., Hartmann, B., Agrawala, M., and Fitzmaurice, G. Delta: a tool for representing and comparing workflows. In *Proc. CHI 2012*, ACM (2012), 1027–1036.

21. Laadan, O., Baratto, R. A., Phung, D. B., Potter, S., and Nieh, J. DejaView: a personal virtual computer recorder. *SIGOPS Oper. Syst. Rev. 41*, 6 (Oct. 2007), 279–292.

22. Ma, K.-L. Image graphs – a novel approach to visual data exploration. In *Proc. VIS 1999*, IEEE Computer Society Press (1999), 81–88.

23. Macko, P., and Seltzer, M. Provenance map orbiter: Interactive exploration of large provenance graphs. In *Proc. USENIX TaPP 2011* (2011).

24. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., and den Bussche, J. V. The open provenance model core specification (v1.1). *Future Generation Computer Systems 27*, 6 (June 2011), 743–756.

25. Nakamura, T., and Igarashi, T. An application-independent system for visualizing user operation history. In *Proc. UIST 2008*, ACM (2008), 23–32.

26. Oliver, N., Smith, G., Thakkar, C., and Surendran, A. C. SWISH: semantic analysis of window titles and switching history. In *Proc. IUI 2006*, ACM (2006), 194–201.

27. Perin, C., Vuillemot, R., and Fekete, J.-D. SoccerStories: A kick-off for visual soccer analysis. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2506–2515.

28. Pirolli, P., and Card, S. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *2005 International Conference on Intelligence Analysis* (2005), 1–6.

29. Pretorius, A., and van Wijk, J. Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 685–692.

30. Rattenbury, T., and Canny, J. CAAD: An automatic task support system. In *Proc. CHI 2007*, ACM (2007), 687–696.

31. Schacter, D. L. The seven sins of memory: Insights from psychology and cognitive neuroscience. *American Psychologist 54*, 3 (1999), 182–203.

32. Schmidt, B., Doeweling, S., and Muehlhaeuser, M. Interaction history visualization. In *Proc. SIGDOC 2012*, ACM (2012), 261–270.

33. Sugiyama, K., and Misue, K. Visualization of structural information: automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man and Cybernetics 21*, 4 (1991), 876–892.

34. von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J., Fekete, J.-D., and Fellner, D. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum 30*, 6 (2011), 1719–1749.

35. Wexelblat, A., and Maes, P. Footprints: history-rich tools for information foraging. In *Proc. CHI 1999*, ACM (1999), 270–277.

36. Yeung, M., Yeo, B.-L., and Liu, B. Extracting story units from long programs for video browsing and navigation. In *Proc. ICMCS 1996*, IEEE (1996), 296–305.