VISUALIZING THE ANALYSIS PROCESS: CZSAW'S HISTORY VIEW

by

Nazanin Kadivar
Bachelor of Science, Alzahra University, 2004

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the
School of Interactive Arts and Technology

SIMON FRASER UNIVERSITY

Spring 2011

APPROVAL

Name: **Nazanin Kadivar**

Degree: **Master of Science**

Title of Thesis: **Visualizing the analysis process: CZSaw's History View**

Examining Committee:

Chair:

_____

Dr. Marek Hatala
Associate Professor, School of Interactive Arts and Technology

_____

Dr. John Dill
Senior Supervisor
Professor Emeritus, School of Interactive Arts and Technology

_____

Dr. Robert Woodbury
Co-Supervisor
Professor, School of Interactive Arts and Technology

_____

Maureen Stone
Supervisor
Adjunct Professor, School of Interactive Arts and Technology

_____

Dr. Tom Calvert
External Examiner
Professor Emeritus, School of Interactive Arts and Technology

Date Defended/Approved: February 28, 2011

Declaration of
Partial Copyright Licence

SFU SIMON FRASER UNIVERSITY
THINKING OF THE WORLD

STATEMENT OF ETHICS APPROVAL

The author, whose name appears on the title page of this work, has obtained, for the research described in this work, either:

(a) Human research ethics approval from the Simon Fraser University Office of Research Ethics,

or

(b) Advance approval of the animal care protocol from the University Animal Care Committee of Simon Fraser University;

or has conducted the research

(c) as a co-investigator, collaborator or research assistant in a research project approved in advance,

or

(d) as a member of a course approved in advance for minimal risk human research, by the Office of Research Ethics.

A copy of the approval letter has been filed at the Theses Office of the University Library at the time of submission of this thesis or project.

The original application for approval and letter of approval are filed with the relevant offices. Inquiries may be directed to those authorities.

Simon Fraser University Library
Simon Fraser University
Burnaby, BC, Canada

Last update: Spring 2010

ABSTRACT

Capturing and visualizing the data analysis process is a growing research domain. Visual Analytics tool designers try to understand the analysis process in order to provide better tools. Existing research in process visualization suggests that capturing and visualizing the history of the analysis process is an effective form of process visualization.

CZSaw is a Visual Analytics tool that provides visual representations of both data and the analysis process. In this thesis, we discuss the design and development of CZSaw's History View, a method we have designed and developed to capture and visualize the history of data exploration sessions with CZSaw. Our goal is to understand how analysts employ visual history in the analysis process. Achieving this goal helps us in providing better support for analysts. We conducted an informal user study based on which, we have developed a list of user expectations and suggestions for history visualization.

Keywords: History tracking, Visual Analytics, CZSaw, History visualization, Process visualization, History capturing, Branching history

پدر و مادر بزرگوارم

از گرمای آفتاب وجود شماست که سرما هرگز به زندگی من راه نیافته است.

حاصل این چند سال تلاشم، هر چند ناچیز است، ولی تنها دارایی من است که با عشق و احترام به شما پیشکش می کنم.

نازنین

Dedicated to my parents, Mansour Kadivar and Elaheh Farsi, with love.

ACKNOWLEDGEMENTS

Accomplishing this thesis was impossible without those who supported me, taught me, and contributed to this work. I wish to thank the following individuals for making this thesis happen:

My parents Mansour Kadivar and Elaheh Farsi, for encouraging me to face this challenge, with great support and sacrifice. My senior supervisor Dr. John Dill, for his mentorship and support, and for teaching me essential lessons not only for becoming a better researcher, but also for becoming a better person. Dr. Robert Woodbury for providing critical feedback on my work. Maureen Stone for offering her time to discuss this work, providing critical suggestions and commentary, and for supporting me emotionally when I needed it the most. Dr. Chris Shaw for providing essential feedback and suggesting new ideas, and for supporting me with faith. CZSaw development team: Saba Alimadadi, Victor Chen, Dustin Dunsmuir, Jeffrey Guenther and Eric Lee for sharing and debating ideas through these years and for their collaboration. My lab mates: Richard Arias-Hernandez, Andrew Wade and Aaron Smith who helped and inspired me. My brother Amin Kadivar for inspiring me with his hard work and dedication. My best friend and roommate, Melody Siadaty for always being there for me, always offering a hand to help and an ear to listen. My wonderful friend Behzad Behroozan for his friendship, help and support, and for trying hard to make me organized. Maryam Maleki, Roham Sheikholeslami, Arash Shiva and Sepideh Zolfaghari for their friendship and support.

I consider myself blessed for having so many wonderful friends that I cannot name one-by-one here. I am thankful to all my dear friends in Canada who helped me miss home much less, and my dear friends in Iran who never failed to cheer me up even though they are miles apart.

Last but not least, I would like to thank Joyce Black, Tiffany Taylor, Valerie Moorman, Desiree Nazareth and Patrick Phang, without whom my experience at SIAT would not be as wonderful as it is.

Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1: INTRODUCTION

We are faced today with the growing need to analyze and understand large collections of dynamic, incomplete and unreliable data. To deal with this, a new field called Visual Analytics was formed in 2004.

"Visual analytics is the science of analytical reasoning facilitated by visual interactive interfaces."(Thomas et al. 2005) Visual analytics tools provide powerful visual representations in order to support the sense-making process. Pirolli and Card (Pirolli and Card 2005) define sense-making as a set of analytical tasks that "consist of information gathering, representation of the information in a schema that aids analysis, the development of insight through the manipulation of this representation, and the creation of some knowledge product or direct action based on the insight".

Much of the current Visual Analytics literature focuses on data visualization. However, the data analysis process itself is a rich source of data. There is a growing body of research on capturing and supporting the data analysis process including (Springmeyer, Blattner, and Max 1992), (Amar, Eagan, and Stasko 2005), (Jankun-Kelly, Ma, and Gertz 2007), (Shrinivasan and van Wijk 2008a), and (Gotz and Zhou 2009). So far though, there has been relatively little effort focused on improving the analyst's awareness and understanding of the analysis cycles by allowing them to easily review their analysis interactions. The history of interactions with a Visual Analytics tool is an important part of the analysis process. Analysts benefit in several ways from having access to this history and from being able to review, edit and replay it.

Our group (Visual Analytics Research Laboratory at SFU-Surrey) has developed and continues to develop a new Visual Analytics tool called CZSaw, aimed generally at problems of investigative analysis. An important part of CZSaw is its set of integrated methods for capturing and visualizing the analysis process itself. This thesis explores history capturing and visualization mechanisms for Visual Analytics systems. A significant part of this study was the design and implementation of a history-capturing tool for CZSaw, which is called the History View. This thesis will discuss history in the context of Visual Analytics systems, and describe the design process for the CZSaw History View. It will also describe a qualitative usability study that we conducted to observe the usage of the History View in a data analysis session. This thesis will discuss the findings from the study and the resulting suggestions for future directions for improving both CZSaw and the History View.

CZSaw is a Visual Analytics tool that provides analysts with visual representations of the data and helps them develop insight by visually manipulating the data. In this thesis, this interaction with CZSaw is called "the analysis process". Besides providing visual representations of data, CZSaw provides a visualization of the analysis process to help support the sense-making process. CZSaw includes multiple views for visualizing the analysis process, among which, History View is the subject of this thesis.

Capturing the history of the analysis process is a way of documenting the analysis process. Being able to describe the data analysis and sense-making processes is important for the analysts and for collaboration. Analysts sometimes document the analysis process in journals or lab notebooks. Gotz defines insight provenance as a historical record of the process and rationale by which an insight is derived (Gotz and Zhou 2009). Journaling

the insight provenance is a technique to recall the analysis process later. However, journaling is interrupting: analysts must interrupt their analysis and thinking process to journal their findings. Lipford et al. conducted a study with real data analysts and showed that analysts become immersed in the data analysis process and interaction with data visualizations to the extent that they forget to think aloud or make notes about their findings even when they are explicitly asked to do so (Lipford et al. 2010). Journals prepared by the analysts without any support from the system are often incomplete and may include errors, due to humans' short memory or mistakes. Automation helps avoid journaling being a separate process and eliminates errors. A visual representation of the analysis history provides analysts with a precise visual journal of the analysis process. They can annotate this visual journal and recall steps more easily by looking at images and representations.

The data analysis process is usually meticulous and time consuming. During the analysis process, analysts may be interrupted, distracted by other issues or need to pause the process and resume it later. To resume the analysis process, we assume that a visual representation of the analysis history will help analysts recall analysis processes quicker and with more details. Capturing history helps support "undo", which allows the analyst to recover from errors in the system or reasoning.

Analysis is not linear. Data visualizations usually include multiple opportunities for exploration. In other words, there are usually multiple analysis avenues from any given data visualization. Nevertheless, analysts are able to explore only one of the possible avenues at a time. A visual history system with replay ability will allow analysts to quickly go back to a data-enriched visualization, and take new avenues of exploration.

We will call this “branching” in the rest of this thesis. Capturing and visualizing the analysis process history will support returning to the branching point and visualizing the different analysis paths.

During the data analysis process, analysts develop hypotheses and try to confirm them by finding evidence in the dataset. If a hypothesis proves to be wrong after some analysis, a visual history will help analysts to trace the path back to the point that they started, and restart the analysis from that point with a new hypothesis. This edit-ability characteristic of visual history allows analysts to discard steps based on an incorrect hypothesis and reuse the error-free steps for further data exploration.

Additionally, including a visual history in Visual Analytics systems facilitates communication between analysts. Having the data analysis process captured and visualized will make it easier for analysts to recall and describe the data analysis and the sense-making processes to others.

Overall, we believe that visual histories are helpful features for the Visual Analytics tools. We believe this area includes many opportunities and it is worth more research and development.

The main contribution of this thesis is designing and prototyping an example of a history capturing and visualization mechanism that uses timelines and screen snapshots for capturing and navigation. This history capability captures the state of the system after every analytical interaction, as well as after actions that cause changes in the system state. Adding a timeline to this history unit enables users to find the system states that they look for, quickly and efficiently. This design also facilitates revisiting previous states of the system and comparisons between multiple system states. Replaying the history and

changing the state of the system to an earlier state is enabled by simply choosing the desired screenshot. This history capability also provides an annotation facility for enriching and improving the journals.

To understand user expectations from a history visualization unit, we conducted a simple qualitative user study in which we observed users and documented their interactions in a data analysis session using CZSaw. The outcome of this study is a list of suggestions that illuminates our future direction.

Our research is focused on capturing and visualizing the history of interactions with a Visual Analytics tool during the data analysis process. We have designed a history capturing and visualization unit that supports the analysis process. Our history capturing and visualization facility is integrated with a Visual Analytics tool called CZSaw. CZSaw is a visual analytics system that captures and visualizes the analysis process and history of user interactions with the data. CZSaw includes a number of visual data representations that allow analysts to explore and understand the data. During the data exploration process, interactions with these data representations are captured in a scripting language (Kadivar et al. 2009). Our history capturing mechanism corresponds with the captured script, by providing a screenshot for every analytical interaction that is recorded in the script.

We designed and reviewed several different visualizations for representing the history of the analysis process, and chose the one that best pursues our goals. In order to understand the application of history representations in the analysis process, we conducted a qualitative study in which we tried to understand how analysts use a visual history facility during the analysis process. We also tried to understand whether a visual

history representation helps in recalling and describing the analysis process. Our findings helped us in creating a roadmap for our future direction, to create a more effective history visualization facility that responds to the analysts needs precisely and provides better support for the analysis process.

In chapter two, we review the research body of history capturing and visualization techniques in Visual Analytics and information visualization. In the following chapter, we describe the design and functionality of the visual history facility that we have designed and implemented. We call this facility “History View” in the rest of this document. We describe our design options, our limitations and constrains and the final design of our History View. Chapter four includes a description of our user study, our employed methodology, design of the study and its procedure. In chapter five we discuss the results of our study and the lessons that we learned from it. We will conclude with a roadmap for our future work on empowering our history capturing and visualization mechanism and summarize this thesis in the final chapter.

2: LITERATURE REVIEW

The word "history" in this thesis refers to a sequence of interactions with a computer application. History capturing is a mechanism for saving history for later retrieval. Capturing history shows the paths that computer application users took to complete a task. History visualization refers to applying information visualization techniques to represent the captured history in a visual format. History capturing and visualization is addressed in multiple domains in computer science. Computer applications in which users complete tasks through multiple steps often provide a history mechanism that enables undoing and redoing. Word processing applications like Microsoft Word and Image creating and editing applications such as Adobe Photoshop and Adobe Illustrator are examples of such software applications.

In this chapter, we look at existing history models in computer applications, with a focus on Visual Analytics tools.

2.1 What is history?

We define "history" as a sequence of interactions with a computer application. "Interaction" refers to the actions that users perform which affect the state of the artefact being produced or changed by the computer application. Meaningful interactions are defined by the system designer and in the scope of the system. For example in a text editing application such as Microsoft Word, where the final document is what a user expects from the system, "interaction" has a different meaning than a Visual Analytics

system like CZSaw, where the final artefact is the whole period that a user has been engaged with the system. For example in a text editing application where inserting a diagram into a document, cutting text or typing characters are considered interactions, double clicking on a word will not be considered as a meaningful interaction and therefore the history unit will not capture it. On the other hand, in a Visual Analytics tool such as CZSaw, double clicking on a node in a graph is considered a meaningful interaction and is captured by the application's history unit. Interactions are defined with respect to the domain of the software application. In 2.3.1we discuss the meaningful interactions in Visual Analytics domain in more details.

Many factors should be considered in designing a history-capturing unit for a computer application. Jeff Heer et al. present a design space analysis of history capturing systems, which can be used as a guideline for designing a history unit for computer applications (Heer et al. 2008).

2.1.1 Heer's guideline for designing a history unit

In their design space analysis, Heer et al. describe three main factors that should be considered for designing and implementing a history unit:

1. History models
2. History visualizations
3. Operations on history

They categorize existing history mechanisms based on these three factors. Table 1 summarizes Heer's categorization.

By History Models, Heer et al. refer to existing solutions for capturing the history of user interactions with the software application. History Visualization discusses the methods for visually representing the history items and designing the layout of the history visualization. By a "history item" Heer et al. refer to the information that a system captures after an interaction. Based on the system designers' strategy, a history item may be a screenshot, a block of script or a set of parameters. "Operations on History" presents a high level list of possible operations on the history visualization.

Heer approaches history tools from a usability perspective. There are other categorizations for existing history mechanisms with different approaches. Kreuseler et al. present another design space analysis for the history management units from the implementation perspective. Kreuseler's categorization is discussed in 2.1.3.

For designing CZSaw's history unit, we used Heer's categorization to help ensure that we have considered all the possibilities for designing a history unit, and to understand the effects of our design decisions on our history unit's functionality. We also use this categorization as a basis for comparing CZSaw's History View with other history tools.

In the next section, we describe History View based on Heer's categorization. This will help in better understanding both History View and Heer's design space analysis.

History Models	Actions vs. States	Developers must decide if their history system will maintain sequences of states, actions, or both.	
	History Organization	Stack Model	Items are placed on both undo and redo stacks. The redo stack is cleared when new actions occur
		Timeline Model	Items are stored items in the linear order in which they occur.
		Branching Model	Items are stored in a tree structure, and actions performed after undo operations form a new branch of the tree.
	Hierarchical Command Object	History items might exist at multiple granularities; e.g. group a sequence of low-level actions into a single higher-level action.	
	Local and Global History	History items can be organized by the objects on which actions are performed.	
Visual Representations of History	Visual Presentation	Using text description to represent history items or using graphical representations such as icons or thumbnails.	
	Spatial Organization	Linear sequence of items, continuous timeline, node-link tree diagrams, content-centric representations are various ways for visualizing history items	
Operations on History	Navigation	Undo-redo, click the thumbnail of a history state, timeline slider, content-based navigation	
	Editable Histories	Editing the history, replaying past actions or revising the history. Selective undo and redo are examples of editing the history.	
	Metadata and Annotation	Annotating history items. Bookmarking, keyword tagging, text comments and audio annotations are examples of annotations.	
	Search and Filter	Metadata such as time, action type, bookmarks and annotations are all potential search domains.	
	Export	Exporting and sharing parts of the history is important for communication and collaboration.	

Table 1: Heer et al.'s design space analysis for history tools

2.1.2 Description of CZSaw's History View based on Heer's guideline

In this section, we compare CZSaw's history unit with the guideline that Heer et al. have provided. Detailed information about CZSaw's history visualization, its functionality and limitations can be found in chapter three.

In CZSaw, we translate user interactions into sequences of statements in CZSaw's scripting language. Every statement consists of a function and a number of parameters. We define a "transaction" as the sequence of script statements needed to implement a user interaction, such as "Search". Another example of a transaction is expanding a node in a node-link diagram. Thus, transactions provide a semantic description of a user's

intention. This means that instead of capturing lower-level interactions like mouse clicks or mouse movements, CZSaw captures higher level interactions such as “search” or “expanding a graph node” as transactions. Semantically meaningful user interactions are described in more detail in 2.3.1. In sum, CZSaw captures every analytical interaction as a transaction.

History Models: In CZSaw, we capture a screenshot at the end of each transaction and save it as an image file. CZSaw captures both the states of the analysis process (screenshots) and the actions that transform one state into another one (transactions). Actions and states are represented in two separate visualizations in CZSaw: the script view and the history view, respectively. A brushing technique is employed to show the correspondence between these two visualizations. When users brush over screenshots in the history view, corresponding transactions in the script view are highlighted. Brushing script statements does not currently highlight corresponding nodes in the History View. This will be added to the next version of CZSaw.

In their design space analysis, Heer et al. state that history organization can be a stack model, timeline model or branching model. A stack model refers to history mechanisms that store history items in an undo-redo stack. The redo stack is cleared when new actions occur after one or more undos. The stack model does not capture the occurrence time of the history items. Although the history items are stacked in a temporal order, there is no notion of the time that they occurred. The timeline model stores history items on a timeline and allows selective undo. The occurrence time of each history item is stored and visualized in this model. The branching model captures the branches in the analysis process and allows selective undo. Occurrence of new interactions after a

number of undos, results in creating a new branch in the history tree. Our current history organization is a timeline model without branching capability. The structure of our timeline is explained in more detail in chapter 3. In the future work section we note that our goal is to change our current model to a branching model.

By global history model, Heer et al. refer to the history mechanisms that capture the history of the whole project, rather than its sections. In contrast, a local history model captures the evolution history of small sections of the project. For example in Visual Analytics systems, a global history model captures the state of all the views together after a new interaction. However, a local history model will capture the history of single data views separately. CZSaw's current history model is a global history in which the state of the whole system is captured and traced. In the future work section we describe that we aim to expand this model to a model that will cover both local and global history.

History Representations: In terms of representing the history, Heer et al. suggest that two factors should be considered: visual representation and spatial organization of the history items. Visual representation refers to whether text or images are being used in order to visualize history items. CZSaw's history representation consists of both text and image. The actions (transactions) are represented as text, and history states (screenshots) are represented as thumbnail images. Spatial organization refers to the information visualization technique that is used to layout the history items on the screen. In CZSaw's History View we represent the history items on a timeline. The timeline shows the overall analytical activity with CZSaw during the analysis period. History View also represents history items in a second panel, which we call the detailed view. The detailed view shows

the details of the captured screenshot. CZSaw also visualizes a linear sequence of transactions corresponding to user interactions as text in its script view.

Operation on History: Heer et al. categorize possible operations on history into five categories. The first operation on history is navigation, by which they refer to interaction methods for browsing and exploring the history visualization. Navigation in CZSaw's history visualization is via selecting thumbnails or by using the timeline slider. These navigation techniques are described in more detail in chapter 3.

The next operations on history are editing and replaying. Both of these operations are implemented in CZSaw. Users can replay the analysis process step by step to their desired state. They can also edit the history by interacting with the system after performing one or more undos.

Another operation on history is storing metadata and annotation corresponding to history items. CZSaw users may annotate history items. Besides users' annotations, we automatically store some metadata, such as occurrence time, about history items.

Search and filter are other operations on history. CZSaw's history model does not include these operations at this point. However, we are planning to develop these operations in the next version on History View.

The last operation on history based on Heer et al.'s model is export. By export they refer to the capability of creating reports from the history. This operation does not exist in current version of CZSaw, but we hope to add this capability to CZSaw in the future.

2.1.3 Kreuseler's guideline for designing a history unit

Another guideline for history capturing and visualization is presented by (Kreuseler, Nocke, and Schumann 2004). They state that adding a history unit to visual data mining tools (VDM) supports data analysts better and more effectively in four ways:

1) Simplifies the user interactions via undo/redo mechanisms,

2) Improves the user comprehension of the exploration process,

3) Improves the reusability of mining techniques, parameters and results, and

4) Supports selection of appropriate mining techniques for some dataset.

Kreuseler's history organization is a temporal branched history. They show history both as a tree, to show the branches, and as a timeline that shows the temporal order of branches. Kreuseler's guideline is summarized in Table 2.

Kreuseler's guideline is more on the technical side. Their guideline categorizes possible solutions for implementing a history unit with efficient memory usage and robust technical foundation. In contrast, Heer's guideline is focused on the usage of a history view from the user's perspective. Therefore, these two guidelines can be used together to create a history unit with reliable technical basis and efficient usability.

In this thesis our focus is on the usage of history units rather than the technical aspects of implementing them. Thus, we use Heer's guideline for describing and comparing existing history units.

Considerations in designing a history management unit						
Internal History Management				Interface between History Unit & VDM system	External storage of history tree	Visual representation of history
Store actions	Store actions & inverse actions	Store system state	Store system state changes			
All actions stored.	Actions & inverse actions both stored. (Enables undo.)	Entire state stored as image or text.	All internal data structures changed by an action are stored.	VDM system sends interaction information to history management, functionalities of VDM system can be externally driven by history management.	History tree & its metadata stored in editable & retrievable format & does not require significant memory space.	Visual representation of history should not overlap with data visualizations; history display updated for each new event; history visualization displays most important properties on current history.

Table 2: Kreuseler et al.'s guideline for designing a history unit (VDM = Visual Data Mining)

2.1.4 Description of CZSaw's History View based on Kreuseler's guideline

In this section we compare the implementation aspects of CZSaw's History View with Kreuseler's guideline. Detailed information about our implementation can be found in section 3.5.

Internal history management: In CZSaw we store all actions (transactions) and system states (screenshots). We store actions as text and system states as image files. Since CZSaw's history unit does not support undo, to roll back the state of the system to an earlier state, we have to rerun the script to the selected state.

The interface between the history unit and the VDM system: History View and the CZSaw engine interact through an interface layer by passing transaction names, which are unique in every CZSaw project. The CZSaw engine triggers History View for

storing new transactions and History View tells the CZSaw engine to run the script up to a specific transaction.

The external storage of the history tree: We store actions and metadata about screenshots in text format which is editable and retrievable and is memory efficient. We store history states as images, which are retrievable but is not memory efficient. It is stated in our future direction section that we have plans for saving system states as text rather than images in future versions.

Visual representation of history: History View gets updated every time that user interacts with CZSaw. History View is a separate window and user can change its location and size to avoid overlap with the other views. It shows system states and the timeline of the data analysis.

2.2 History in other computer science fields

History mechanisms have been implemented in computer applications in various domains. Regardless of the application domain, history units capture and visualize history either as a linear sequence of actions or thumbnails, or as a tree that visualizes branches in history. In linear history models, interacting with the system after rolling back multiple steps replaces the rolled back actions with new interactions. This means that the previous history is lost. In branching models, items are stored in a tree structure, and actions performed after undo operations form a new branch of the tree. (Vitter 1984) proposes an Undo, Skip, Redo (US&R) model that supports redo like a linear undo model, but allows skipping of some operations during the redo. Instead of a linear list, the US&R model keeps a tree data structure for maintaining history so that it becomes possible to restore

the state to any point in the history (unlike the linear undo model) (Prakash and Knister 1994). (Derthick and Roth 2001) present a model for visualizing the branches of the data analysis process as a tree on a timeline. (Brodlie et al. 1993) present a branching visualization for the problem solving process. (Hightower et al. 1998), (Card et al. 2001), (Klemmer et al. 2002) , and (Callahan et al. 2006) present various branching representation techniques. Both linear and branching history mechanisms are implemented in various computer science fields, including Visual Analytics.

Text editors such as Microsoft Word include a limited undo/redo stack that holds recent interactions in temporal order. Using undo in such tools is so common that a key combination (ctrl + Z) is used in a wide range of applications.

Image editing tools such as Adobe Photoshop include history units as well. Photoshop creates a stack of interactions for every project and captures a limited number of interactions in this stack. When the number of captured interactions becomes larger than the available stack space, Photoshop automatically deletes the oldest history items at the bottom of the stack to provide space for the new interactions. (Kurlander and Feiner 1998) present a history tracking mechanism for Chimera, a graphical editor tool. They show the history of interactions as a set of thumbnails. The user can scroll through the sequence of panels, reviewing actions at several levels of detail, and selectively undoing, modifying, and redoing previous actions. (Meng et al. 1998) propose a history system for graphical editor applications that visualizes history as a linear sequence of items. The linear representation facilitates visual scanning of the history and enables quick navigation by clicking on history items.

(Plaisant et al. 1999) present a history mechanism for collaborative learning environments. They describe the design space and challenges of implementing learning histories. They present a guideline for designing learning histories, and implement a learning history for an engineering learning environment.

Perhaps, the domain that includes most efforts on history capturing and visualization is web browsing. The literature is this domain is very rich in terms of history mechanisms. Web browser history visualizes users' browsing paths and allows them to easily find the web pages that they had visited before. Web browser history also shows how users have accessed a web page. (Ayers and Stasko 1995) present a history visualization for web browsers that is similar to our initial design for CZSaw's history unit. They show the browsing history as a tree. Nodes of the tree represent web pages. Users can zoon in on a node to see the content of the webpage and some meta data about it, or zoom out to see the browsing tree (Figure 1). (Waterson et al. 2002) present a visual history unit called PadPrints which is very similar to Ayers's history visualization. (Doemel 1995), (Fisher et al. 1997), (Frécon and Smith 1998), (Kashihara, Satake, and Toyoda 1998) , (Wexelblat and Maes 1999), (Klemmer et al. 2002), (Tabard et al. 2007) and (Haraty 2010) present models for capturing and visualizing the history of web browsing. They use node-link diagrams to show users' web browsing history. These visualizations differ mostly in the layout, interaction technique and interactivity level. (Turetken and Sharda 2007) survey many of these visualizations and produce a guideline for designing visual representations of web browsing history.

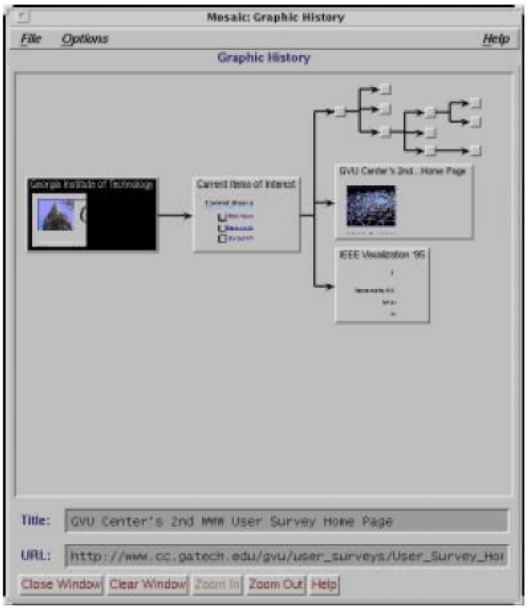

Figure 1: Ayers et al.'s history visualization for web browsing history

2.3 History in Visual Analytics

The literature on history tracking and mechanisms for Visual Analytics tool has increased significantly in recent years. Data analysis processes often occur over many sessions of several hours. The analysis process is an iterative process in which analysts modify their hypotheses and questions over time and through exploring data. A visual history of the analysis process allows analysts to recall the analysis process more easily, faster and more precisely after a hiatus. It also helps in finding the previous steps of the analysis process to restart the analysis path quickly with a modified or new hypothesis.

We have divided the literature about history mechanisms for Visual Analytics systems into two sections. The first section talks about which interactions should be captured in history. Papers in this section introduce taxonomies of meaningful analytical interactions. Papers in the second section introduce various history visualization projects. We have used Heer et al.'s guideline to evaluate these history visualization mechanisms.

Our experience suggests that there is a need for more research on evaluating history-tracking tools. Designing a proper user study for evaluating history tools is very challenging. History tools are designed to be effective over longer period of software usage. However, user study sessions are usually short and it is almost impossible to keep users motivated and committed to the study over a long period. Suggesting a guideline for evaluating such tools would be very beneficial for researchers working in this field.

2.3.1 What should be captured?

The first step in capturing and visualizing the history of the analysis process, is to define taxonomy of interactions that are semantically meaningful in the analysis process, and have to be captured. Several studies examine ways to characterize analytical activities with Visual Analytics tools and provide taxonomies of analytical interactions.

Gotz et al. introduce a taxonomy of semantically meaningful interactions with the Visual Analytics tools (Gotz and Zhou 2009). They form their taxonomy based on the user's intent when performing an analytical task. In a similar study, Amar et al. present a taxonomy of the ten most important analytical interactions (Amar, Eagan, and Stasko 2005). Another similar taxonomy is presented by (Springmeyer, Blattner, and Max 1992). They observed scientists in various fields analyzing their data, and created a taxonomy of analytical interactions similar to those presented by Gotz and Amar. Table 3, Table 4 and Table 5 show these taxonomies.

These three studies use different terms for similar concepts. They have different approaches to categorizing the interactions. Although the three proposed lists overlap on the analytical interactions that they suggest, their categorization perspective varies

considerably. In CZSaw 1.0 we capture a subset of these analytical interactions as CZSaw transactions. We do not apply any categorization on the transactions in CZSaw 1.0. In the next version of CZSaw we have to define a thorough list of actions based on the suggestions from the literature and our own observations during our user study, and provide categorization of actions in order to provide better filtering and search capabilities.

Gotz's proposed taxonomy for analytical interactions				
Exploration Actions		**Insight Actions**		**Meta Actions**
Data exploration actions	Visual exploration actions	Visual insight actions	Knowledge insight actions	
Filter Inspect Query Restore	Brush Change metaphor Change range Merge Sort Split	Annotate Bookmark	Create Modify Remove	Delete Edit Redo Revisit Undo

Table 3: Gotz et al.'s taxonomy of analytical interactions

Amar et al.'s taxonomy of analytical interactions
Get value Filter Computer derived value Find extreme Sort Find Range Characterize distribution Find anomalies Cluster Co-relate

Table 4: Amar et al.'s taxonomy of analytical interactions

Springmeyer et al.'s taxonomy of analytical interactions			
Interacting with representations	**Applying mathematics**	**Maneuvering**	**Expressing ideas**
Generate Examine Orient Query Compare Classify	Calculate Derive new conditions Generate statistics	Navigate Data management Data culling	Record Describe

Table 5: Sprinmeyer et al.'s taxonomy of analytical interactions

2.3.2 History visualization in Visual Analytics

An increasing number of projects are aimed at designing effective history visualizations to support the data analysis process. Most of the history mechanisms introduced in literature, like CZSaw's history unit, are designed to work along with a specific Visual Analytics application and thus are hardly applicable in other tools.

(Ma 1999) presents history as a node-link diagram in which nodes are small thumbnails of the analytical states and edges represent the actions that transfer one visualization state to another. Ma's history unit is implemented for a medical imaging system. This system includes only one data visualization. Therefore, by looking at the thumbnails, the changes that are caused by an action are clearly visible. This technique is not applicable for data analysis tools like CZSaw in which a user works with multiple visualizations. In such tools, the visualizations that are changed by the action might be fully or partly covered by other visualizations, or the change might not be easily distinguishable in the thumbnails.

(Edwards et al. 2000) propose a model for visualizing history on a timeline. Their model separates the local history of the smaller artifacts of the analysis and visualizes

them on a timeline. Merging the local timelines in a multi layer fashion shows the global history of the whole system.

(Shipman and Hsieh 2000) present branching history visualization for Visual Knowledge Builder. Their history mechanism visualizes the hypertext creation process on a timeline.

(Derthick and Roth 2001) propose a branching history visualization for a data exploration tool, Visage. They show the history items on a tree diagram which sits on a timeline. They capture interactions with visage as text. Their history tree is shown as simple even lines on the timeline. Active branches at each period are highlighted.

(Robinson, Weaver, and Center 2006) present Re-Visualization for capturing and reusing analysis sessions. Instead of capturing meaningful analytical interactions, such as those suggested by (Gotz and Zhou 2009); their history system captures low-level interactions such as mouse movements and mouse clicks. They suggest that analysts should group these lower-level interactions into groups of semantically meaningful interactions. They visualize these lower level interactions in multiple layers on top of the original data visualization.

(Jankun-Kelly, Ma, and Gertz 2007) introduce a general model for visualization state as a set of parameters, and actions as transformations of these parameters. They demonstrate a model to describe the analysis process and a representation to share it. Both the visualization technique performed and the process used to generate visualization results are captured by the model and representation.

(Heer et al. 2008) present a history visualization technique for the Tableau visualization system. They capture the state of the visualizations as VizQL statements based on the model presented by (Mackinlay, Hanrahan, and Stolte 2007). They have also created a list of possible interactions with Tableau's visualization, and used that for logging the actions in a textual format. They have enabled branching in their history unit.

(Shrinivasan and van Wijk 2008b) present a history mechanism that captures analytical reasoning steps as a branching history and provides revisiting and replaying facilities.

(Gotz and Zhou 2009) present a history visualization unit for HARVEST system that captures the semantically meaningful interactions with the tool in a temporal order. They capture the interactions in the history stack and user can roll back multiple steps to revisit previous analysis states.

In Table 6 we summarize the characteristics of these history units based on Heer's guideline for designing a history unit that we introduced earlier.

		CZSaw	Ma	Edwards	Shipman	Derthick	Robinson	Kelly	Heer	Shrinivasan	Gotz
History Models	Actions vs. States	a/s	a/s	a	s	a	a/s	a/s	a	a	a/s
	History Organization	tm, bm	bm	sm, tm	tm, bm	tm, bm	bm	tm, bm	tm, bm	sm	sm, tm
	Hierarchical Command Object	✓		✓			✓	✓	✓	✓	
	Local and Global History	g	g	l/g	l/g	l/g	g	l/g	g	g	g
Visual Representations of History	Visual Presentation	thu	thu	icon	text	text	text	thu, text	icon, text	icon, text	icon, thu
	Spatial Organization	lsi, ct, nltd	nltd	ct, nltd	ct	ct	lsi	lsi	nltd	lsi	ct
Operations on History	Navigation	clth, tls	clth	tls		tls		clth	clth	clth	tls
	Editable Histories	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Metadata and Annotation	✓						✓	✓		✓
	Search and Filter							✓			
	Export	✓						✓			

Table 6: Existing history visualizations for Visual Analytics tools based on Heer's guideline for designing history units (sm = Stack Model, tm = Timeline Model, bm = Branching Model, thu = Thumbnail, lsi = Linear Sequence of Items, ct = Continuous Timeline, nltd = Node-Link Tree Diagram, clth = Click the Thumbnails, tls = Timeline Slider)

2.4 Summary

Surveying existing history mechanisms and comparing them with each other and our own implementation has led us to create the following list of observations that will help us in improving our solution in the future, and will provide research questions for further explorations.

1. Most of the history units that capture the state of the system, translate the graphical characteristics of the data visualizations into textual parameters, and for retrieving the system states, they render the image based on those parameters. A few other history units capture system states as images, only when the user asks for screen capturing. None of the surveyed history units captures the screen into an image file after every transaction (as CZSaw does currently). The problem with our approach is that capturing a good resolution screenshot after every transaction causes the project file to grow very quickly and use lots of disk space. We believe that the best solution provided for capturing the system state is translating it into textual parameters. However, this approach has some limitations. First, it is convenient for data visualization systems like Tableau, where the user can only use one visualization type at a time to look at data. This approach might not work for Visual Analytics systems like CZSaw, Jigsaw (Stasko, Görg, and Liu 2008) or InSpire[1](Anon. 2010) where data exploration often requires multiple simultaneous visualizations. There needs to be further

[1] PNNL: IN-SPIRE™ - Home. May 2010. http://in-spire.pnl.gov/

exploration of designing state capturing technique for such text analysis systems.

2. Existing branching history units use tree visualization to show the history tree. We capture interactions with the system as a script file, which is editable and repayable. The problem of visualizing branching histories in sequential script file is not addressed in the surveyed research reports.

3. Searching and filtering history is mostly limited to bookmarks and annotations. Creating action/state-based searching and filtering will be beneficial.

4. Exporting history files for collaboration purpose in not addressed much in the surveyed literature. However, most of these research reports note that an important usage of history is journaling and collaboration. History units with an export facility produce history files readable only by the system that produced the history. Therefore, there needs to be more work on producing effective journals and reports that are readable by common co-operation tools.

5. Most of the effort in the history capturing and visualization domain is focused on designing and implementing an effective history unit. However, there are very few research reports about what we learn from these history units. The goal of visual history representation is to support the analysis process. However, none of the discussed research reports presents the findings from employing these history mechanisms in real data analysis sessions. (Lipford et al. 2010) conducted multiple user

studies that prove users recall the analysis process better by looking at visual representation of history in short analysis sessions. However, they state that they have doubts about the effectiveness of a visual history for recalling longer analysis sessions.

6. History capturing and process visualization units are designed to support long analysis sessions. Therefore, current research methods focused on short user study scenarios are not helpful for testing history-capturing units. Further research on designing proper study methods for testing such systems would be beneficial.

Simple forms of history capturing are common in many computer applications. 'Undo' is a common operation in many computer applications, and is an operation on history. History capturing and visualization in data analysis process and Visual Analytics application is a new research area. The literature in this area is growing and new techniques are being proposed. However, we still need to do more research and development on creating history capturing and visualization techniques that are proper for this domain.

In the next chapter, we will discuss a history capturing and visualization unit we have created for CZSaw, a Visual Analytics application.

3: HISTORY VIEW

We stated earlier that History View is the history capturing and visualization unit of CZSaw. In order to understand the functionality of History View, it is important to know what CZSaw is.

CzSaw (Figure 2) is a visual analytics tool that captures and visualizes the data analysis process and history of user interactions with the data. CzSaw includes a number of visual data representations that allow analysts to explore and understand the data. During the data exploration process, all user interactions are both captured by CzSaw in a scripting language and visualized in a visual history view so that analysts can look at the history of their interactions to find repetitive patterns and identify alternative investigative avenues. These other directions can then be explored without losing track of previous work. CZSaw also provides the ability to explicitly program the process. Steps in the analysis can be replayed on new data or in variations to remove the burden of repetitive actions. CZSaw has four main advantages: 1) Capturing and visually representing dependencies among primary and inferred data items, 2) Recording an editable sequence of analysis steps, supporting review and editing of an analysis process, 3) Visualizing the analysis process through history, script and dependency views, maintaining consistency among these views, 4) Enabling higher level analysis through its scripting language.

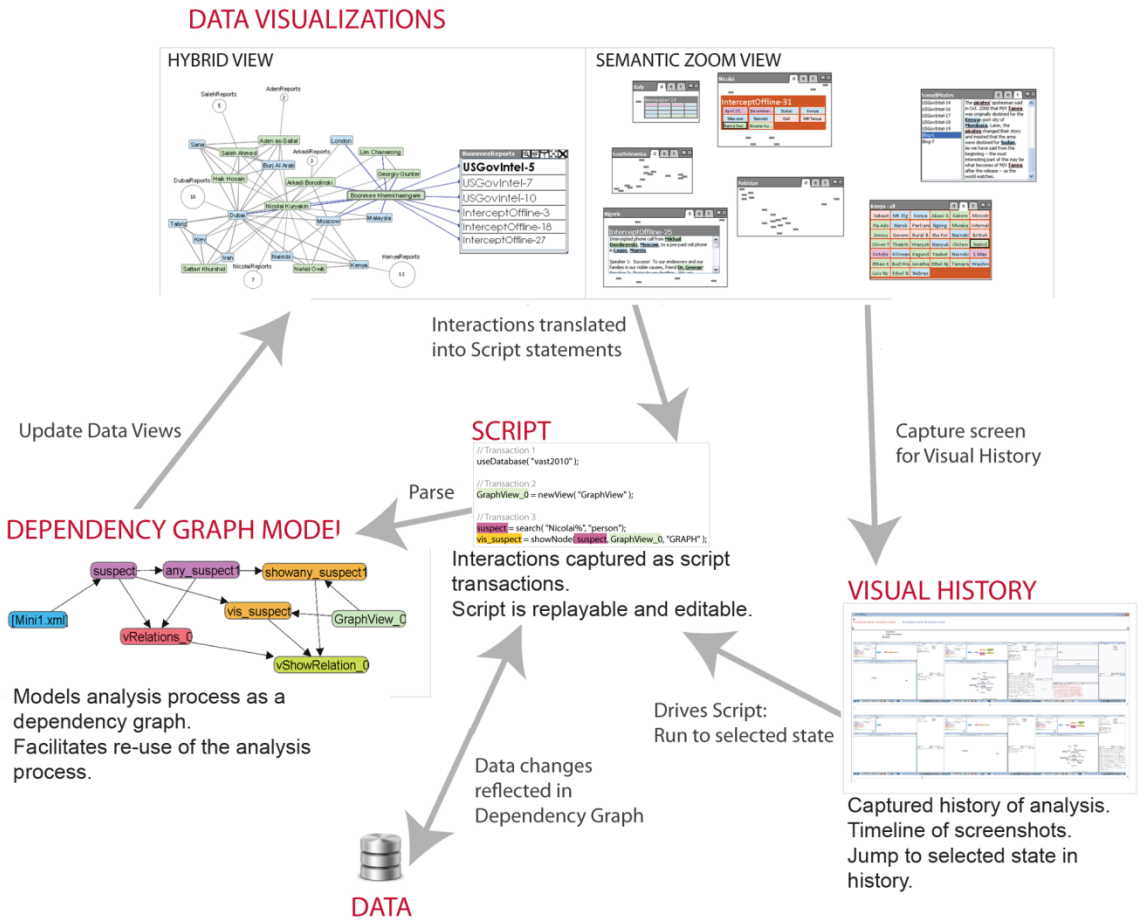

Figure 2: CZSaw architecture

CZSaw includes a number of data visualizations for exploring datasets, and three process visualizations. The three process visualizations are: 1) Script View: a view of the generated script statements in a reviewable, re-playable and editable manner; 2) History View: a visual history of the analysis process via a temporal sequence of data views. The history view is also the place where an analyst can add notes; 3) Dependency graph: a special graph view showing as nodes the various entities, collections of entities etc. generated during the analysis, with links indicating dependencies.

CZSaw includes a scripting language for performing operations meaningful to the analysis process. The language consists of commands a user may enter directly. Users of course also interact directly with data views and such user interaction is translated into a block of one or more commands that we call a transaction. Thus, analysts perform the analysis process by either interacting with views or by directly typing script commands. Transactions are at the level of meaningful tasks completed in the system; script commands are programming elements that implement transactions. Only actions that modify the visual or data model of the system are currently recorded.

3.1 History capturing and visualization in CZSaw

History View is designed to record and visualize the analysis process. Analysts benefit from History View when they need to look at the analysis process for the purpose of recalling the process, describing it to others or rolling back the state of the system to a state that they visited earlier.

Every time a user performs a meaningful analytical interaction with CZSaw, CZSaw captures the interaction in a script, translating the interaction into one or more programming language-like statements containing appropriate functions and variables. In CZSaw terminology, a transaction is a block of functions and variables that corresponds to one analytical interaction. Thus, transactions are building blocks or steps of the analysis process. The state of the system changes every time that a new transaction is processed. These changes might be visible in the views, or might change the analysis structure without having any visual effect. At the end of every transaction, we save the visual state of the system via screen capture as a node in history. Every history node represents a transaction and consists of a screenshot and relevant metadata.

Each node in History View corresponds to the state of the system at the end of completing a script transaction. Whenever analysts commit model-altering interactions with the system – interactions that generate a corresponding transaction – we capture and time-stamp a screenshot of all of the open CZSaw data views. This time-stamped image becomes the next "node" in the History View. Each history node has a name, which is also the name of the transaction that the history node refers to. We use transaction names as pointers into the script from History View. Whenever the analyst needs to step back to adjust the analysis process, or when he or she needs to step back to explore an alternative analytical avenue, they first find the history node at which they wish to try a different analysis path. Then, selecting that history node triggers CZSaw's script engine to replay the generated script of the analysis process from the beginning to the end of the selected node's transaction. This will update the state of all the views to the state that they had at the selected time.

3.2 Design goals and constraints

We had a number of goals that we tried to reach by designing and implementing History View. However, there are a number of constraints and limitations that were imposed on our development process.

Our main goal in designing History View was to create an editable and replayable history unit that would capture every analytical interaction with CZSaw. Our ideal history unit would pursue the following goals:

- It would capture every analytical interaction with the tool (such as those introduced in 2.3.1) and would filter out interactions that do not have any analytical intent (such as mouse clicks on the views without any specific

purpose). It would use minimum memory both for storing history data and for rerunning history.

- It would show the branches of the history on a single page without hiding important details and would allow zooming on history states to users' preferred zoom levels.
- It would visualize both the system states and their corresponding actions clearly and with appropriate detail and readability.
- It would provide easy navigation methods for browsing the history.
- It would provide searching, filtering, annotation and bookmarking.
- It would provide an easy and quick report generating facility.

Although we achieved some of these goals in our current prototype, we believe that most of them will be achievable in the future and as we find solutions to overcome the constraints and limitations that we are facing. These constraints are:

- History View captures the state of the views of the system after every transaction. Therefore, CZSaw should have a list of interactions that we consider as transactions. When these transactions occur, CZSaw should notify History View about occurrence of the new transaction. CZSaw 1.0 captures a limited number of transactions. Therefore, the history file of the projects that are created with CZSaw 1.0 might look incomplete and user might not find all the interactions that they expect in the History View. In CZSaw 2.0, we have addressed this problem by expanding the list of the transactions. However, we still need to work on creating a complete list of possible analytical interactions with CZSaw, and define our transactions based on that. By achieving the complete list of transactions, users will be assured that all of their analytical interactions will be captured in and retrievable from History View.
- We can capture the system states in two formats: text or images. Text format includes a number of parameters to describe the state of the visualizations, such as their location on the screen, the layout of the data items on the visualization and colours as well as a set of parameters to save the data that is visible on the screen. These parameters should be complete enough to allow

rendering the picture of the screen. The image format captures the state of the screen as an image file. Both formats use lots of memory, especially because analytical reasoning includes numerous interactions with the VA tool, which produces a large number of system states in every analysis session. Although an image is an easier format for capturing, it is not as precise as text because the state of the views that are fully or partially covered by other views would not be captured. On the other hand, capturing all of the properties of the visual state of the system, as well as the visible data at each state, is very tricky and error prone. We are studying the tradeoffs among level of details, accuracy levels, memory usage and rendering process in order to design an efficient method for capturing the system states.

- Pixel space for visualizations is limited and visualizing the tree of the analytical history without hiding important information such as branches is a challenge. Visualizing detail and context on the same page without hiding critical details is the subject of many research projects such as CZWeb (Collaud et al. 1996). We are studying possible solutions in order to design a visualization that fulfils the requirements of our system.
- Accessing real data analysts to interview about the usage of a history unit in the data analysis process, or to observe them using history units is very challenging due to high demands of their time.
- Testing History View is a challenge because we need to have users who are familiar with CZSaw and can use it to analyze data. CZSaw is still a research tool and we have only a limited number of people who are tool experts (our research group members), but are not expert data analysts.
- Another problem in respect to testing the effectiveness of History View is that user studies usually involve short sessions while history units are aimed at long analysis projects. It is thus difficult to generalize results from (short) user studies to realistically long data analysis sessions.

We hope to address and solve these challenges in future versions of History View.

3.3 Design evolution

To design the history visualization, we first started by making a list of the features that we wanted to visualize. Our goal was to show: 1) the size and shape of the analysis process, 2) branches in the analysis path, and 3) the analysis time and the gaps in the analysis period. Besides visualizing these features, we wanted to design a visualization that 1) showed detail and context on one page and, 2) allowed users to focus easily on various sections of the analysis path without losing the context.

To accomplish our design goals, we started by exploring a number of existing detail and context information visualization techniques like "Continuous Zooming" (Dill et al. 1994), "Fisheye" (Furnas 1986, vol. 17) and "Perspective Wall" (Mackinlay, Robertson, and Card 1991). We designed and prototyped two visualizations by using zooming techniques: 1) Continuous zooming history and 2) Zooming and panning history.

Continuous zooming was the first zooming technique that we used to show detail and context of the analysis path in our history visualization. Continuous zoom is a distorted view method for displaying hierarchically- organized, two-dimensional networks. The method is suitable for large networks. The continuous zoom shows detail in context, unlike simple pan and zoom techniques, and allows for more than one focus point, unlike fisheye viewing (Dill et al. 1994). Continuous zooming is implemented in CZWeb, a system that traces a user's path through web space (Fisher et al. 1997). Inspired by CZWeb we designed a history visualization in which older screenshots would collapse into clusters to provide space for newer screenshots. The main problem of this design was visibility of data. Collapsed screenshots were not visible on the screen and the

user had no idea about the length and shape of the analysis process. The other problem of this design was that there was no notion of time in the visualization.

In our next prototype, instead of continuous zooming, we implemented a simple zooming and panning technique (Figure 3). Screenshots would appear on the screen in temporal order along a horizontal line. Branches would be added vertically. Users could zoom in to see the details of the screenshots or zoom out to see an overview of the analysis process. Though this design solved the previous design's visibility issue, it was tedious for the user to zoom in and out continuously to look at different parts of the analysis path in more detail. This design also suffered from not including any notion of time in the visualization. Although history nodes were being captured in a temporal order, there was no explicit method of showing the time that every screenshot was captured or the gaps in the analysis period.

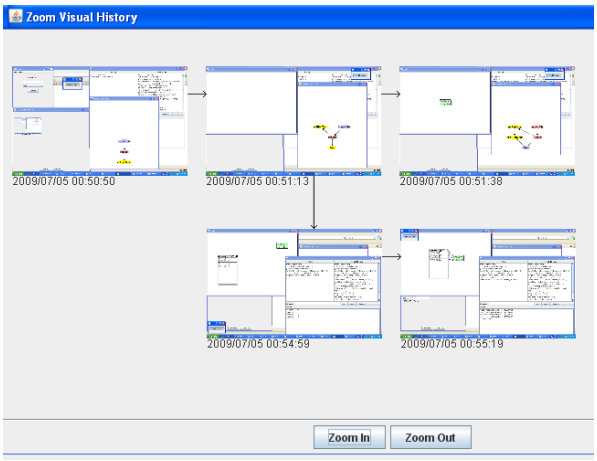

Figure 3: Zooming and panning history

By exploring the previous two visualizations, we realized that we would benefit from separating the overview and the detailed view in two individual panels. The main disadvantage of this separation is that users' attention would be divided among multiple

panels. However, this separation would allow users to see an overview of the analysis process at all the times in one panel, while they were focusing on the details of specific analysis states in the detailed view panel. Therefore, we decided to design a visualization that contained two views: 1) an overview that illustrated the analysis process, branches and the analysis time and 2) a detailed view that allowed users to see the details of the screenshots (system states).

We explored several techniques for each of the two views, seeking an effective combination to visualize the history of the analysis process.

3.3.1 The Overview panel design evolution

As mentioned earlier, we decided to show an overview of the analysis path on a separate panel in the history visualization. An overview should allow the user to understand the analysis time, pattern and branches. An overview would also help users find the analysis states that they wanted to focus on. Our goal was to design an overview to show three features clearly: 1) analysis path's size and pattern, 2) analysis time and 3) branches in the analysis path.

In order to find an appropriate design for the overview, we explored existing similar visualizations and created some prototypes. Based on the lessons that we learned by exploring the design space, and based on the feedback that we received from our potential users, we chose two of the prototypes to implement.

Our goal in designing the first visualization for the overview was to show the whole analysis process within the history window. We wanted to avoid horizontal scrolling for seeing the overview and thus spread whatever the current set of screenshots

was over the width of the overview window. Figure 4 illustrates this technique. When there is only one screenshot captured in the history log, we assign the whole window width to that screenshot. Every time that a new screenshot is captured, we redraw the overview.

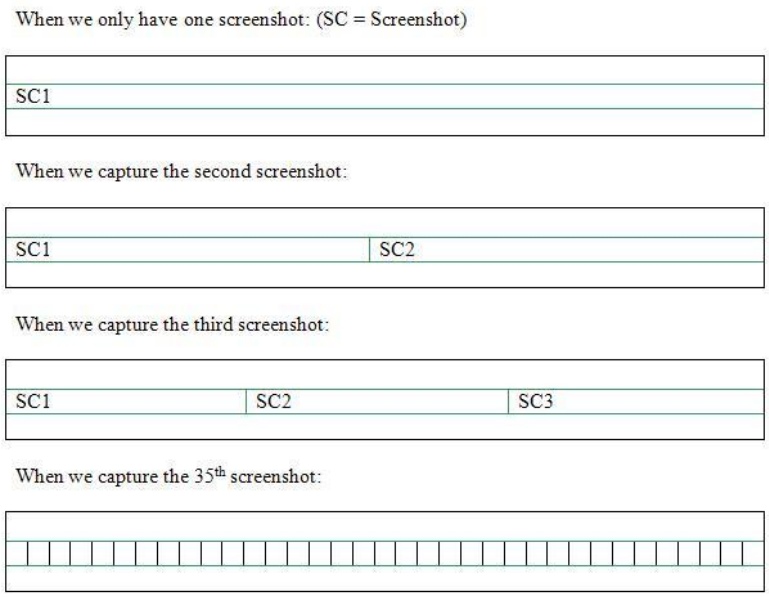

Figure 4: Overview design 1

Branches are added in a vertical order to this overview. To view the time, users hover the mouse pointer over the cells to see the time as a tooltip. This design fails to illustrate the analysis path when the number of the screenshots is larger than the window width, because we will have to collapse multiple screenshots into one pixel. It is even possible that multiple branches collapse into one pixel which makes it impossible to understand the analysis path by looking at the overview.

Another problem of this design is the visibility of time. By looking at the overview, it is not possible to say how much time has been spent on the analysis process.

The amount of activity during the days, and the gaps between the analysis times are not visible in this visualization.

To solve the process path visibility and time visibility problems, we designed a new overview visualization which was inspired by a stock market diagram[2] (Figure 5). This overview included a hierarchical timeline (Figure 5) that showed analysis months and days. Users could see the number of the screenshots that were captured in a month, week or day as text labels above the timelines. Users could interact with the timelines by using two sliders to select the sections that they wanted to focus on. Although this method solved the time visibility problem, it still could not show the analysis path and branches.

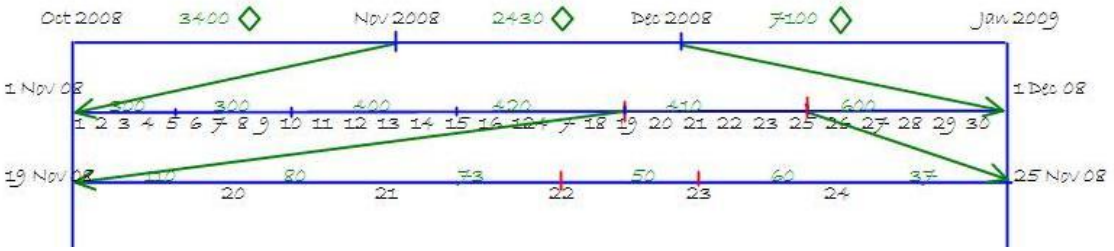

Figure 5: Overview design 2

In order to visualize the analysis path and time more clearly, we finally decided to add horizontal scrolling to the timeline. We assigned a fixed number of pixels to every time unit between the analysis' start and end dates. The sketches in Figure 6 show a timeline in which one pixel is assigned to every hour (24 pixels to each day). This view is scrollable and the timeline expands as the analysis time grows. Numbers show the date and the green bars below the day cells indicate the hours that the user has been active.

[2] http://www.google.com/finance?q=google+stock

Gaps in the analysis time are clearly visible in this design. However, the amount of activity in every hour is not clear. For example if the count of captured screenshots on November 10th is twice as much as November 17th, the diagram does not show this difference. Branches of the analysis path are not visible in this design either.

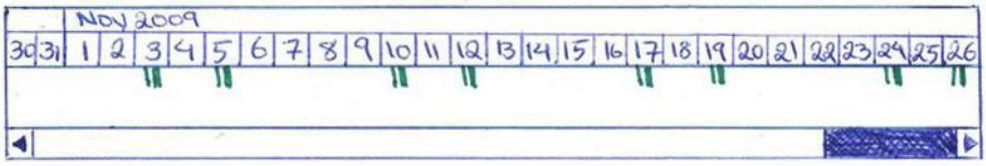

Figure 6: Overview design 3

In order to show the amount of activity in every hour, we extended the previous timeline to the timeline in Figure 7. In this timeline, we use the height of the green bars to show the amount of activity in every hour. This design shows analysis time and time gaps clearly. The level of activity during every hour is clearly illustrated in this design as well.

Figure 7: Overview design 4

Finally, to show the branches in the analysis path clearly, we decided to dedicate a panel in the history visualization to branches. We realized that this separation between the timeline and the branches helps in making them clearer.

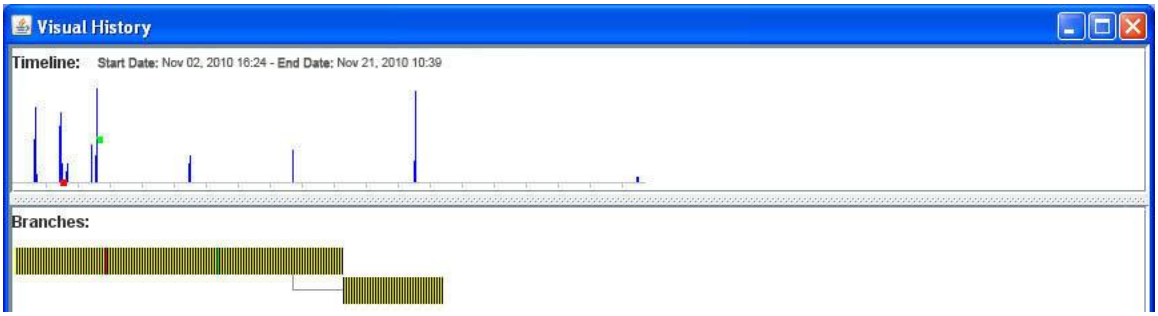

Figure 8: Final overview

Figure 8 illustrates the final design for the overview. The top panel shows the timeline that we described earlier. The x-axis shows time and the y-axis shows the activity level at each hour. The vertical blue bars show that user has interacted with the system at the time that corresponds to each bar. The height of the bars is proportional to the number of the captured transactions at the corresponding time. The empty spaces between the vertical bars represent gaps in the analysis process. The bottom panel shows the branches in the analysis path. In the branches panel, we assign one pixel to every screenshot. The alternating color on the vertical bars helps to distinguish separate pixels (screenshots). Figure 9 shows a cut from the zoomed view of the branches panel. Each yellow or black bar represents one screenshot. Branches appear in vertical order. The bar that is highlighted with a red border in Figure 9 shows a node from which a new branch is started. This panel is scrollable both horizontally and vertically. There is brushing between the timeline panel and the branches panel. We mentioned earlier that for every screenshot, we assign pixels in both timeline and branches panels. When a user selects the pixels that represent a screenshot in any of these panels, the pixels that represent the same screenshot on the other panel become highlighted to show the correspondence between the views.

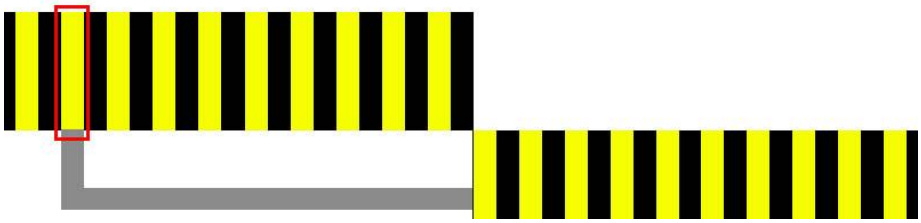

Figure 9: Branches panel

Among these prototypes, we implemented the first (Figure 3) and the last (Figure 8). The last timeline (Figure 8Figure 1) is the one that is currently included in our system.

3.3.2 The Detailed View panel evolution

While the overview panel provides an overall understanding of the analysis path and its characteristics, the detailed view panel shows the details of every screenshot (system state). Users can select interesting screenshots on the overview and see their details in the detailed view.

In order to design a useful visualization for the detailed view panel, we explored existing detail and context visualization techniques such as the Perspective Wall (Mackinlay, Robertson, and Card 1991) and the Fisheye lens (Furnas 1999). Our goal was to design a visualization that allowed zooming in and out on the screenshots and smooth navigation between them. We designed and implemented two visualizations for the detailed view.

The first version of the detailed view panel was a modified and simplified version of the Perspective Wall visualization without the 3D effect (Figure 10). In this visualization, there were five screenshots on the panel. The middle screenshot was the focused screenshot. Smaller screenshots on the left and right of the focused screenshot

showed the system states before and after the focused state. To navigate between the screenshots, users could click on any of the smaller screenshots to move them to the centre.

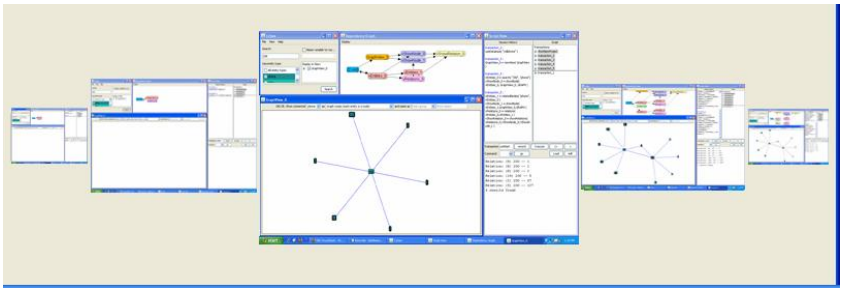

Figure 10: Two-dimension Perspective Wall detail view

In our second visualization, all of the screenshots on the panel are in the same size (Figure 11). Users can change the number of visible screenshots by changing the value of a dropdown list. The window width is divided evenly between the screenshots. To navigate between the screenshots, users can drag the slider on the slider bar that is below the screenshots.

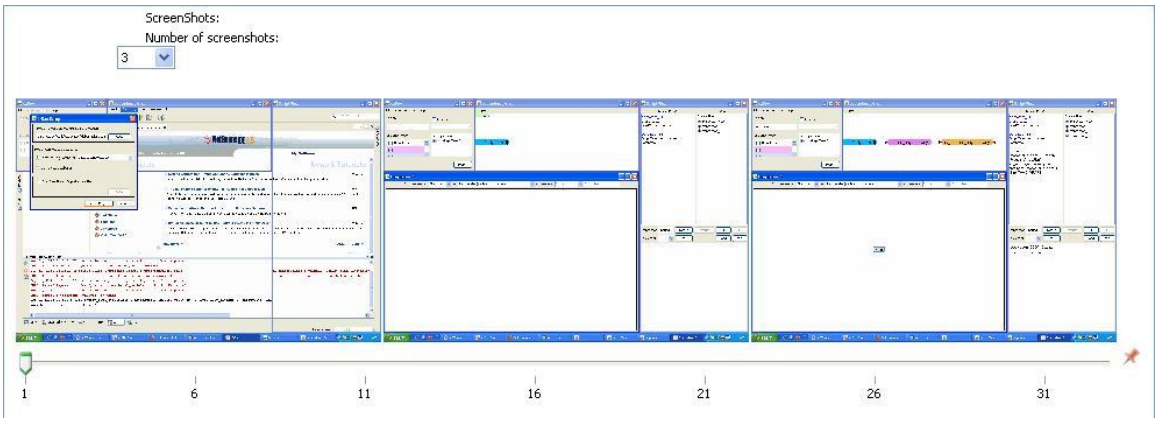

Figure 11: Final detailed view

We implemented both of these visualizations for the detailed view and by combining each one of them separately with the overview visualizations, we implemented two versions of History View. These two versions are described in more detail in the next section.

3.3.3 History View evolution

Besides the Continuous Zooming History and the Zooming and Panning History that we described in the previous section, we designed two more history visualizations.

Our third history visualization was the result of combining the 2D Perspective Wall detailed view with the overview in which we divided the window width between the screenshots. This history visualization is illustrated in Figure 12.

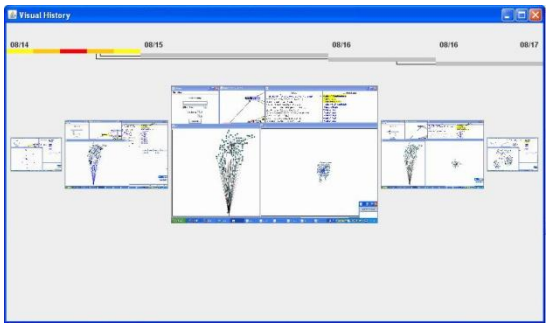

Figure 12: 2D Perspective Wall history

Unfortunately, we did not receive positive feedback about this visualization. Most of the users complained about the ambiguity of the overview and the need for continuous clicking on the detailed view to change the focused slide. We realized that by adding more information to the overview and adding a slider to the detail view we could solve these problems. Thus, we changed this design to better serve the requirements of the users.

Our fourth and final visualization, which is the visualization that is currently integrated into CZSaw, is described in the next section.

3.4 Design tradeoffs and the current design

Our initial designs for history visualization were all inspired by detail and context techniques. Our goal was to design a visualization that shows everything on one page. We also sought effective zooming techniques to allow users to focus on parts of the analysis process without losing the overall image (context). However, as we stated earlier in 3.3.2, by exploring multiple design options and discussing them with our potential users and researchers in Information Visualization and Visual Analytics domains, and by reviewing and learning from the similar research projects, we realized that detail and context methods like Continuous Zooming, Perspective Wall and Fisheye were not suited to visualizing the interaction history. All of the zooming techniques discussed in the previous section and in Chapter 2 affect the visibility of information. We realized that to make the history visualization more comprehensive and helpful, we have to avoid techniques that include collapsing multiple nodes into one node, or zooming out to the level that history nodes lose their readability. Hence, we decided to show detail and context on individual panels and provide brushing and linking between the panels.

Based on the discussions with the users and designers, we recognized the need to visualize analysis time explicitly and clearly in our design. We observed that users expect a history visualization to help answer questions such as: “How did the data visualization that I was looking at two days ago look?”, “What were the other exploration possibilities on the graph view that I created this morning?” or “What was the analysis state at a specific date?” This led us to show the time in the overview panel. The timeline (x-axis)

shows the activity level during the entire analysis period. It also enables users to look at the state of the system at any desired time.

One of the limitations that we faced in most of the designs that we explored, was the capability to focus on multiple parts of the analysis at the same time. During the interviews with potential CZSaw users, they brought up the question of how to compare the screenshots that are captured at different periods. To make this possible, we needed a visualization that showed more than one analysis period.

Based on these observations and the feedback that we received from the potential users, we designed a history visualization we called History View.

3.4.1 History View

History View, the history visualization that we developed for CZSaw, is capable of capturing and visualizing branched histories. However, due to the limitations of CZSaw 1.0, History View currently captures and visualizes only linear (i.e. non-branched) history. History View includes a timeline and a detailed view. A panel for showing the branches is part of History View but remains invisible until a future version of CZSaw supports branching. Figure 13 illustrates History View.

Figure 13: Final History View

Overview: We mentioned earlier that we decided to add a scrollable timeline to the overview (Figure 14). We show time on the horizontal axis. As the analysis time increases, the horizontal axis - timeline - grows. Since the scaling method used allocates a fixed number of pixels to each time unit (in our case, currently 5 pixels/hour), if the amount of time to be shown becomes larger than the width of the History View window, a scrollbar appears allowing access to parts of the timeline beyond the window's edges.

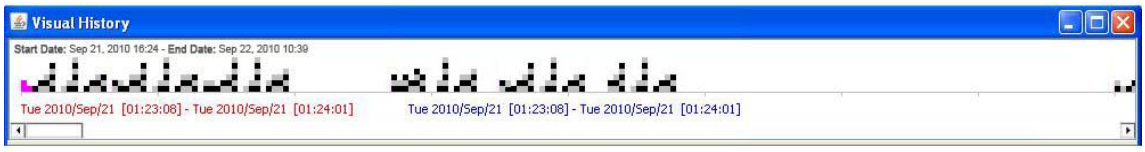

Figure 14: Overview

The vertical bars on the timeline show screenshots captured during the hour represented by that bar. The height of the vertical bars shows the number of screenshots that are captured in an hour. In other words, the height of the vertical bars shows the amount of interaction with CZSaw in one hour. Every screenshot is represented by a cell on the vertical bar. The height of every cell is five pixels and the colour of the cells alternates to distinguish individual cells. Empty spaces between two vertical bars indicate no interaction with CZSaw during that period.

Based on the feedback from users during the early stages of designing History View, we added the capability of looking at two different periods. This feature is helpful when users want to look at the screenshots of two different periods for comparison purposes. In the detailed view, we have two sequences of screenshots, each of which represents one of the periods that a user wants to see. We use colour to show the correspondence between the timeline and the detailed panel. The cells on the timeline that correspond to those in the detailed view are coloured in red or blue. Red shows that the cell represents a screenshot on the detailed view's red trail, and blue means that the cell represents a screenshot on the detailed view's blue trail. The red and blue trails are described in the detailed view description. If some screenshots are being shown in both red and blue trails, the corresponding cells on the timeline will be coloured in purple. Timeline updates these colours automatically every time that user interacts with the detailed view.

The line of text below the timeline shows the start and end date of the highlighted cells. The colour of the text (red / blue) corresponds to the colour of the highlighted cells.

Detailed view: This view shows the captured screenshot in two trails (Figure 15). In the current version, we have provided two trails to enable comparison between the screenshots. The ideal design would allow users to have as many trails as they want. However, we imposed the limitation of two trails in this version in order to keep our design simple for the user experiments. We will remove this limitation in the next versions of History View and will change the design in order to provide effective comparison facility for our users.

Each of the two trails consists of a sequence of screenshots, a slider bar and a coloured icon. The colour of the icon indicates the colour that represents the trail on the timeline. The slider bar shows the total number of the screenshots. By dragging the slider left and right, users can change the screenshots that they can see.

The two trails on the detailed view operate independently from each other. Having two trails allows analysts to compare two sections of the analysis.

The dropdown box above the trails allows changing the number of screenshots in the detailed view. When the number of the screenshots increases, we decrease their size in order to provide space to allocate the new screenshots on the screen. Hence, increasing the number of the screenshots reduces the amount of detail that is visible.

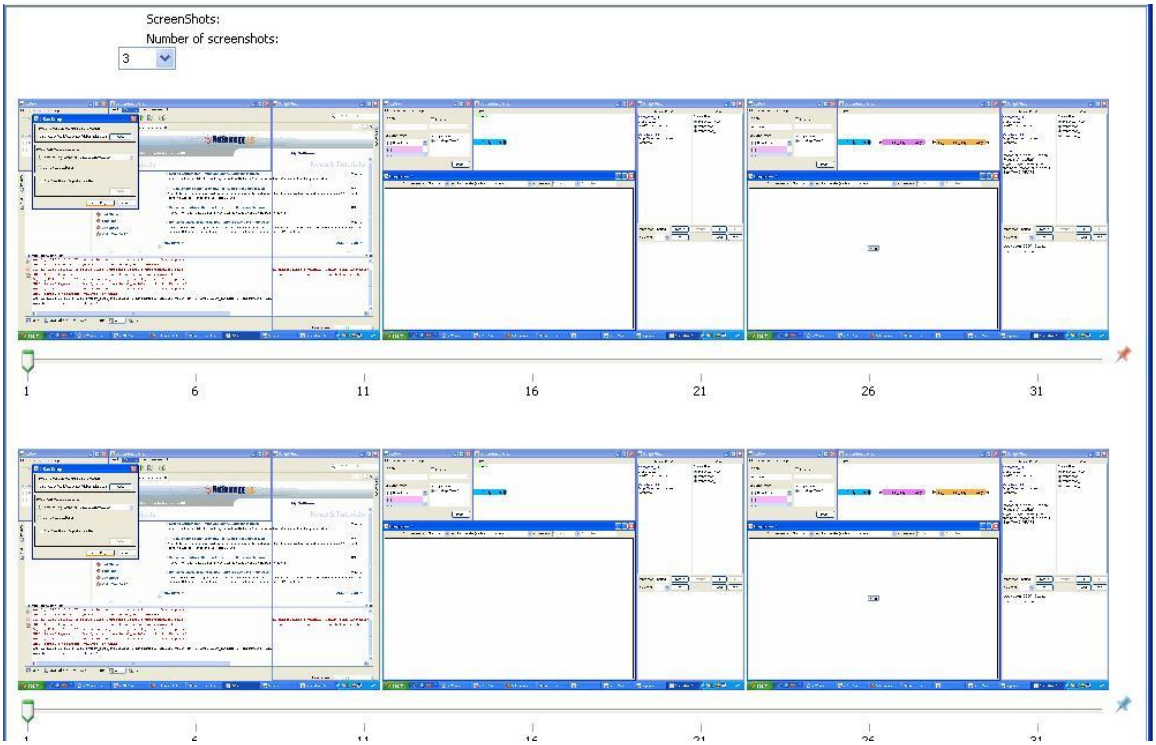

Figure 15: Detailed view

To see the screenshots in higher resolution, users can left-click on them. This will open a popup that shows the selected screenshot in higher resolution.

It is generally accepted that annotation is a desirable feature of a history visualization (Heer et al. 2008); correspondingly, we provide a basic annotation capability. To annotate the history nodes, users can right-click the screenshots and select "Add Note" option from the pop-up menu. This will open a note-taking window. We record the annotation of every history node as its metadata in the history database.

Replayability is one of the features of History View. Users can select "Set State" from the right-click pop-up menu on any of the screenshots, which will cause the CZSaw engine to re-run the script from to beginning to the transaction that corresponds to the selected screenshot, and change the state of all the views correspondingly. Our history

model is linear, by which we mean interacting with the system after a series of undo operations simply erases the corresponding steps, replacing them with the new interactions. For example, if the history included 10 transactions named: Transaction_1 to Transaction_10 and the user set the state of the system to Transaction_6 and then performed new interactions with CZSaw, Transaction_7 to Transaction_10 would be erased from the history, and the new transactions would be captured after the original Transaction_6.

History View provides a specific correspondence with Script View. History View shows the state of the system after every interaction. However, the user interaction and corresponding set of script statements that causes the change between two screenshots is not shown in History View since they are available in Script View. To help users find the script statements that correspond to a state change, we provide brushing between History View and Script View. Every time that user interacts with History View, we send a message to Script View. This message tells Script View to highlight the transactions that correspond to the highlighted screenshots in History View. The transactions in Script View become highlighted with the same colour as the screenshots that they correspond to in the History View. This will provide a visual correspondence to help users to find the actions that changed the states of the views.

3.5 Implementation overview

In this section, we describe the implementation aspect of History View. We describe how History View is implemented and how it cooperates with the other units in CZSaw 1.0.

3.5.1 Implementation technology

History View is part of CZSaw 1.0, implemented in Java. We capture screenshots as .gif files and save some metadata (such as the date and time that the screenshot is captured) for every screenshot in an XML file. The structure of the metadata file is described in the following section. We use the JDOM library for manipulating the XML file.

History View is implemented as a Java "package" with minimum dependence on the rest of the system. History View and CZSaw engine interact only as necessary, and through a messaging protocol. Therefore, changes in History View do not affect the rest of CZSaw and vice versa.

3.5.2 History capturing mechanism

At the end of each transaction, a history node is created. A history node consists of two parts: 1) Screenshot, and 2) metadata.

The screenshot is saved in a file `<transactionName.gif>` and saved in the directory: `Project Folder \ historyImages\.` It is a full resolution screenshot.

History node metadata currently consist of the date and time of the screen capture and the ID of the corresponding transaction. We assign an ID number to every history node that is only valid in the history package and does not affect other parts of CZSaw. If analysts annotate a history node, we store the annotation in the metadata file as well. We also save the transaction id of the predecessor of every history node as metadata to help keep track later when CZSaw supports branches. We currently save metadata into an

XML file in `Project Folder \ nodesIndex.xml`. In this file, each history node is saved as an XML node. The structure of the metadata file is as follows:

```
<HistoryNode ID="">
        <DateAndTime></DateAndTime>
        <Visible></Visible>
        <Note></Note>
        <Parent></Parent>
        <TransactionID></TransactionID>
</HistoryNode>
```

Figure 16 is taken from a sample metadata file that is saved for two history nodes in XML file:

```
- <HistoryNode ID="1">
    <DateAndTime>2010/09/21 01:23:08</DateAndTime>
    <Visible>true</Visible>
    <Note />
    <Parent />
    <TransactionID>transaction_0</TransactionID>
  </HistoryNode>
- <HistoryNode ID="2">
    <DateAndTime>2010/09/21 01:23:32</DateAndTime>
    <Visible>true</Visible>
    <Note />
    <Parent>transaction_0</Parent>
    <TransactionID>transaction_1</TransactionID>
  </HistoryNode>
```

Figure 16: Two history nodes in the history XML file

3.5.3 Integration with CZSaw

Figure 17 shows the model of interaction between the History View components as well as the model of interaction between History View and CZSaw.

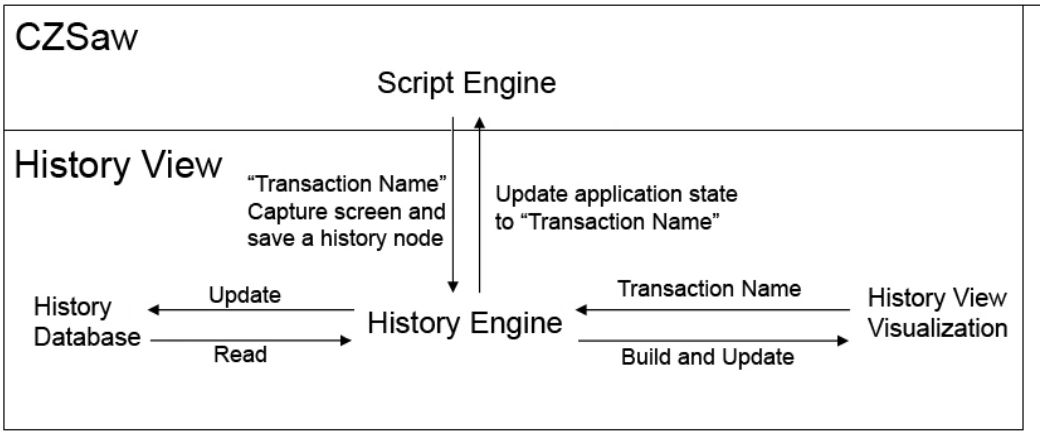

Figure 17: Integration with CZSaw

Every time that CZSaw's script engine captures a new transaction, it sends a message to the history engine that contains the name of the captured transactions. This message tells the history engine that the state of the system has changed and a new screenshot should be captured and added to the history database. The history engine also saves the metadata about the captured screenshot in the database. The history engine then redraws the History View.

When a user requests a state change (i.e. resetting state of the system to a selected screenshot), the history engine sends a message to CZSaw's script engine. This message includes the name of the transaction that corresponds to the selected screenshot. The script engine then changes the state of the system to the requested state.

4: STUDY

In order to understand how users employ the history visualization unit during the analysis process, and to obtain feedback for improving the design of the History View in CZSaw 2.0, we conducted a user study. The details of this study are described in this chapter.

4.1 Study goals

By adding history capturing and visualization capability to CZSaw, we were hoping to help analysts to recall and understand the analysis process easier and in a more effective way. We were also hoping to provide a facility that allows analyst to describe their analysis processes and reasoning to others more effectively. Another goal was to help analysts in storing the analysis path and finding the insightful visualizations for further exploration more easily.

In order to check if we were on track to achieving these goals, and to better understand how users made use of current History View capabilities and their evaluation of them, we designed and conducted a qualitative user study. The questions that we were hoping to answer in this study were:

- How do analysts use the History View during the analysis process?
- Does our History View design help analysts to recall and describe the analysis process in details?
- What do analysts think of our History View? What are their suggestions and recommendations for improving the current version of the History View?

As we will describe in the "Results and Discussion" chapter, the study answered these questions partially, and helped us in designing a roadmap for the direction that History Capturing and History Visualization should take in future.

4.2 Method

In this study, we employed a qualitative method. There were several reasons for employing this approach. Since History View adds new capabilities to CZSaw that were not available before, it is not possible to measure how these new capabilities optimize a task time or accuracy in a quantitative study. As well, our research questions are such that they cannot be answered in a quantitative study, or by only using questionnaires and surveys. A qualitative method thus seemed the best approach for this study.

4.3 Research questions

The research questions to be addressed by this study were:

How do analysts use History View during the analysis process? Does History View help analysts in recalling the analysis process and describing it to others?

For answering this question, we conducted short analysis sessions and asked our users to use CZSaw to do data analysis. We observed users as they were using CZSaw to understand the reasons that they use History View during the analysis process. The results of this study are described in more detail in the next chapter.

How can History View improve to better support the analysis process?

We asked participants what they liked or disliked about History View, and if they had suggestions for improving it in order to serve them more effectively during the analysis process.

4.4 Task

The first task in our study was exploring a dataset by using CZSaw for 20 minutes. The dataset that we used for this study is the VAST 2010 mini challenge 1 dataset[3]. This dataset includes artificial data about illegal arms dealing that may have taken place during 2008. We asked our participants to explore this dataset to identify individuals who met in Dubai during April 2009, and to find out as much as they could about such meetings and the participants.

The study consisted of an introduction and three parts:

Introduction: We described the study and our goal. We also demonstrated History View and described details of each section of this view. We asked users to do the analysis with CZSaw and try to use History View as much as they could.

Part one: User would use CZSaw for twenty minutes to explore the VAST mini challenge 1 dataset. They were asked to think aloud during these twenty minutes.

Part two: Users were asked to open the History View, and describe the analysis process by looking at the screen shots.

Part three: Users were asked to provide comments and suggestions for improving the History View.

[3] http://hcil.cs.umd.edu/localphp/hcil/vast10/index.php/taskdesc/index

The whole analysis lasted from 45 minutes to one hour for each user. A video camera aimed at the screen and desk participants worked at also recorded users' voices. We also used Camtasia to capture the screen contents. We gave users a notepad and a pen and asked them to use any note taking technique they preferred.

4.5 Participants

The goal of our study was to understand the usage of History View during the analysis process. Therefore, we needed participants who were familiar with CZSaw so they could focus on the analysis. Therefore, we could choose participants only from people who were familiar with CZSaw. Because of this restriction, we could only ask the students in the CZSaw research group to participate in this study.

We had four participants who were members of CZSaw Research group and were familiar with CZSaw. Participants were able to employ CZSaw to explore the dataset and find the answer to our question (Who attended meetings in Dubai during April 2009). We also had one user who was familiar with other VA tools but not CZSaw.

4.6 Data collection

The data collected consisted of video files, CZSaw project files, Camtasia files and users' written notes. All digital data was encoded and saved on a secure data server at the School of Interactive Arts and Technology. Hand written notes and consent forms were stored in a secure place as stated in the consent form (Appendix A).

4.7 Limitations

Among the limitations encountered in designing and carrying out this study was the small number of the participants. We needed participants who were familiar with CZSaw and could use it smoothly for data analysis but since CZSaw is not a commercial product, few people are familiar enough with it to employ it in solving data analysis problems. Our participant pool was thus limited to students in the CZSaw research and development team.

Another limitation of this study was time. Real analysis sessions often last from a few hours to a year. Real analysts are engaged in the analysis process over long periods of time. However, in order to keep the total time per participant to one hour, we could only ask for twenty minutes of our participants' time for data analysis. The rest of the study time was spent on using History View to describe the data analysis process, and to discuss the usage of History View and provide feedback and suggestions. During twenty minutes or half an hour of analysis, analysts do not produce enough "interaction history". In addition, a twenty-minute session is short enough for most analysts to clearly recall the whole analysis process without much effort or employing a visual history.

The third limitation was motivation. Real analysts are motivated to focus on a specific problem. Their motivation comes from being engaged in a real problem and trying to find an answer for it. They have clear questions in mind and try to find answers to these questions. In our study, answering the proposed question was not rewarding enough to get the participants involved in the analysis process. Although we were hoping to simulate a real analysis session, lack of motivation was one the factors that we did not achieve our goal satisfyingly.

5: RESULTS AND DISCUSSION

This chapter presents and discusses study findings. Our user study led us to create a list of requirements and develop a guideline for future work.

5.1 Analysis results

Our main question was about the usage of History View during the analysis process. We asked our users to use CZSaw for twenty minutes to explore a dataset. We observed participants to discover if and how they used History View.

Only one user out of five opened and used History View during the analysis process. Therefore, the result of this part of the study was not as precise and insightful as we were hoping. Based on our observation and users' comments, we realized that if analysis session were longer, participants would probably use History View during the analysis process.

Our second question was about the effectiveness of History View in recalling the analysis process and describing it to others. We asked our users explicitly to open History View and describe the process. We observed instances where a participant could not remember exactly what had happened in the screenshot they were talking about, and looking at the neighbour screenshots would help them in recalling the process. We also observed that users continuously looked back and forth between History View and Script View to recall the analysis process. This observation shows that users need to look at

both states (screenshots) and actions (script commands) to recall the details of the analysis process.

5.2 Interpretation of analysis results

The results of the user study are categorized based on our goals and questions about how analysts employ a history unit in the analysis process and how they use a history unit to recall the data analysis process and describe it to other analysts.

5.2.1 Usage of History View in the analysis process

Before starting the twenty minutes of analysis, we asked our users to use History View as much as they could. Our goal was to learn from observing how users employ this feature of CZSaw. Nevertheless, only one of the participants used History View during the analysis process. He used History View after the system crashed. He reopened the project and said that he wanted to continue his analysis from one step before the crash point. Therefore, he opened the History View and looked for the screen shot that showed the system before the crash point. After finding the screenshot that he was interested in, he changed the state of the system to the screenshot's state, and continued work from there. We observed that this user looked back and forth between History View and Script View to understand the actions (script transactions) that created the states (screenshots).

Another participant faced a similar problem. CZSaw crashed in the middle of the analysis process. The participant reopened the project, but did not use History View to find the desired system state directly. Instead, the participant used Script View to run the script, transaction by transaction, until they found the desired state of the system. Later, when we asked this participant about the reason for not using History View, he/she stated

that he/she forgot to use History View to find the state that she was looking for. Running the script, transaction by transaction is feasible for short analysis sessions, like our study, where the number of the transactions are limited (the average number of transactions that were generated in this study is 17). However, in real world situations when the number of the generated transactions increases very quickly, and analysis sessions are much longer, running the script is too time-consuming (at least for current versions of CZSaw).

5.2.2 Recalling and describing the analysis process

In this part of the study, we explicitly asked our users to open History View and describe the analysis process to us. None of the users had any problem in recalling the analysis process and describing it. Two participants used the brushing and linking between History View and Script View to find the transactions that corresponded to the screenshot that they had selected. Two other participants only looked at History View and could still remember the analysis process quickly. However, when these two participants were pointed to the brushing and linking between History View and Script View, they stated that they found it helpful and necessary. All users mentioned that the analysis session was short, and there was no interruption between the analysis session and the recalling session and therefore they had not forgotten the analysis process. They stated that having a visual record of what they did would probably be more helpful in longer and more realistic analysis sessions.

5.2.3 Comments and suggestions

While we were observing our participants using CZSaw and History View, we created a list of shortcomings and issues that we are going to address in next versions of

this tool. We also asked our users to provide us with feedback and suggestions about CZSaw and History View. Many of the suggestions that we received from our users matched the ones that we had captured during the observation. We therefore created a combined list of these observations and suggestions.

- History View is conjectured to be helpful in longer analysis sessions: Our participants claimed that they would use History View if the analysis session was longer or they were interrupted by something during the analysis.
- The captured interactions are not enough: Analysts need to see more details in History View. Interactions that change the layout of the data views such as zooming, relocating data items and changing the spatial location of the data view window need to be captured. Participants also mentioned that they would like to see their interactions with CZNote (CZSaw's annotating facility) captured in the history.
- Branches should be added to History View: Users claimed that if they could do branching in History View, they would use it more to explore alternative paths without losing the history of their work.
- Visibility of history nodes: Users stated that history nodes should show the screen in full resolution so that text and labels are readable.
- History of individual views: Users want to be able to see the history of individual views instead of screenshots of everything that is visible on the screen. Sometimes data views overlap and cover each other. Users want to be able to focus on the history of one view and see the changes in that view instead of the history of everything that has happened to all the visible views.
- Smooth transition in the slideshow: When user moves the screenshots slider by using the mouse or the arrow keys, the screenshots jump to left and right, rather than smoothly moving to the direction that user is aiming to see. This jump between the screenshots is confusing for the participants and there needs to be a smooth and animated transition between the screenshots.
- Zooming and scaling: Users suggested that out timeline should scale based on the analysis time. When analysis time is shorter, seeing data about the minutes

of the analysis would be interesting. However, as the analysis time grows, the timeline should scale to show larger time chunks such as hours or days. Users also suggested that they would like to be able to zoom in and out progressively on the screenshots rather than clicking on them to see the larger image.

- Annotation and bookmarking: Users suggested that a more powerful annotating and bookmarking mechanism for history nodes would help them. They also wanted to be able to highlight the annotated or bookmarked nodes in History View, for searching purposes.

Based on these comments and observations, we created a roadmap for the future versions of CZSaw and History view.

5.3 Focus on the future

Based on what we learned by observing our participants, and based on the feedback that we received from them, we created a roadmap for our future direction. We will address these issues in the later versions of CZSaw and History View.

5.3.1 Future direction: History View

The main goal for the next version of our history capturing mechanism is to capture branches in history. We will also have to design an effective visualization to depict those branches.

We need to change our logging mechanism. In the current version, we capture a screenshot after every transaction. After a short period, we will have many images saved on the hard disk, which makes the project file huge. We therefore need to implement a new caching technique to capture system state without consuming so much memory space. We also have to design a technique to capture and visualize the history of single

views rather than the whole system. We will add a more convenient annotation and bookmarking facility to History View.

In terms of the visualization, we need to address multiple issues. We have to increase the readability of the history nodes. We also need to make a smooth and animated transition between the screenshots. We should provide scaling capability for the timeline and add zooming to both timeline and screenshots panels.

We will use the list in section 5.2.3 as our guideline for taking future direction in improving History View.

5.3.2 Future direction: CZSaw

At the time of writing this thesis, CZSaw research group is working on designing and implementing CZSaw 2.0. In CZSaw 2.0, we will improve the system based on what we learned through developing, testing and using CZSaw for solving realistic problems (such as the IEEE VAST Challenge problems (Dunsmuir et al. 2010)).

Our user study on usage of the history visualization resulted in a list of suggestions and expectations from our users for improving CZSaw.

- **Make a complete list of transactions that we need to capture and implement them in CZSaw:** It is critical that we create a complete list of analytical interactions in the scope of CZSaw. The literature on this (see Chapter 2) provides a good foundation for this process. However, we need to customize existing lists based on the capabilities and specific features of CZSaw.
- **Enable dragging and dropping between the views:** Users asked us to allow them to drag multiple data items from one data view to another data view. Although CZSaw's dependency model clearly separates data and data

visualizations and allows users to visualize a data list in any data visualization by manipulating the script, users still preferred to be able to do this interaction visually. Users stated that they found dragging and dropping between the views easier than editing the script.

- **Enable brushing and linking between the views:** Users wanted better brushing and linking between the data and process visualizations. They wanted to see the data they work with in a data visualization, highlighted in other data visualization as well as the process visualizations.
- **Annotation facility:** Users requested a more powerful annotation facility, so that they could easily capture their findings within the system. CZNote (CZSaw's annotation facility) was at its initial stage at the time of this user study and users could not use it effectively to capture their notes. The relationship between the data analysis items and their annotations were sometimes broken or system would fail to capture and attach the annotations for some of the data items. We believe that these issues will be solved as CZNote further develops.
- **Ambiguity of the dependency graph:** Users found the current (CZSaw 1.0) Dependency Graph visualization difficult to read. The extensive number of nodes and crossing edges, as well as multiple colours and labels on the nodes that include abbreviations and long strings that are automatically assigned by CZSaw, made this graph hard to understand and read. We need to provide a better representation of this graph in future versions of CZSaw.

5.4 Summary

Though our user study did not prove the usability of the History View in the way we hoped, it was nevertheless a valuable experiment for understanding our current state and designing our future roadmap. We are confident that what we learned through this user study will help us in improving our design and implementation for the next version of History View. We also believe that this small study with our group members was a

worthwhile and valuable exercise that helped us in forming our questions and directions for designing a more effective user study for the next version of History View and with real data analysts.

6: SUMMARY AND FUTURE DIRECTIONS

This thesis describes the design and implementation process of a history capturing and visualization unit – called History View - for a visual analytic tool, CZSaw. A model for designing history-capturing units for Visual Analytics software is introduced, and used as a platform for describing the features of existing history capturing and visualization units and comparing them with CZSaw's History View. The results of an informal user study conducted to examine History View's effectiveness are presented and discussed. The study design process identified difficulties in developing suitable evaluation methods for history tracking units in Visual Analytics.

6.1 Summary

Data analysis is a long and iterative process usually occurring over multiple analysis sessions. To help review analysis steps and the reasoning processes, analysts often document the process in a journal. However, creating such journals manually is highly distracting because the reasoning process itself must be interrupted in order to document the process. Further, manual journals may be imprecise because analysts are typically immersed in the analysis process and find it difficult to track and document their reasoning process. It is also proved that analysts forget the details of the analysis process when they try to recall it for the journaling purposes (Lipford et al. 2010). It is thus highly desirable for Visual Analytics tools to support the analysts by providing history and process capturing mechanisms.

History and process visualization in Visual Analytics is a relatively new research topic. CZSaw is a Visual Analytics tool aimed initially at analyzing collections of text documents. A major additional goal for CZSaw is visualizing the analysis process itself, as well as providing interactive visualizations of the data. CZSaw's History View, its history tracking and visualization unit, provides both access to previous analysis states, and supports replay and edit functions on history. History View captures the analysis states as thumbnails and metadata, and provides a visual representation of history with a timeline and an easy navigation technique.

Our main question in this research was to understand history-capturing in the analysis process. Creating History View is a first step in approaching this question. Later in a user study, we observed our users interacting with CZSaw to solve a data analysis problem, and recorded their interactions with History View. We also asked our users for comments and suggestions for improving this history capturing and visualization unit.

At the end of every user study session, we asked the participants if they thought that visual history helped them in recalling the analysis process and all of them stated that it did. Aided by user comments and discussions, we developed a list of expectations for a history-capturing unit. The study also confirmed that realistic usage of a history capability cannot be based on short analysis sessions and specific methods should be designed for evaluating such capabilities.

6.2 Future directions

We developed History View as a history capturing and visualization unit for CZSaw, a Visual Analytics tool. We learned lessons for improving History View, from

designing and developing this prototype and from interviews with our potential users and our user study. Although the results of the user study will help us in improving future versions of History View, we still need to interview real data analysts and understand their expectations of a history management unit before we move further in designing and developing History View. In a related but separate project, we recorded many hours of paired data analysis sessions with Tableau. In these sessions a real data analyst and a tool expert worked together to explore a dataset. Analyzing the usage of Tableau's history management unit in these tapes would be a starting point in understanding users' expectations and requirements.

CZSaw, and thus History View, currently captures only a portion of the interactions of interest. We need to develop a more complete list of meaningful analytical interactions with CZSaw that should be captured by the system and made available through history view. We also need to find an effective solution for storing history data as it grows very rapidly. A search and filter capability should be added to History View and existing annotation facility should be further developed to support digital annotation of the findings.

In terms of the history model, our goal is to understand the requirements of capturing a branching history prior to integrating this feature into History View. We will need to change CZSaw's structure in order to capture branches in the analysis process. We also need to design a mechanism for capturing the evolution history of each data visualization separately.

Various visualization improvements are desirable, including a better navigation technique, smooth transitions in the detail view and zooming and panning for both

timeline and detail view panels. Better visual cues for finding bookmarks and annotations are likewise needed along with a way of smoothly changing the scale of the timeline.

More challenging will be trying to develop methods for evaluating the effectiveness of history capturing units, in short study sessions. It will also be of major value to find and work with real data analysts.

Designing and implementing a powerful history capturing and visualizing capability for CZSaw 2.0 is our next goal; this work will be based on the lessons of this research.

APPENDIX

Consent Statement for the User Study as Appeared in the Ethics Documents

Consent Form

SFU Ethics Approval Application [2010s0571]
Document Viewing Study

Investigator Name: Nazanin Kadivar
Investigator Department: School of Interactive Arts and Technology

The University and those conducting this study subscribe to the ethical conduct of research and to the protection at all times of the interests, comfort, and safety of participants. This form and the information it contains are given to you for your own protection and to ensure your full understanding of the procedures, risks, and benefits described below.

Risks to the participant, third parties or society:

Participants do not give any identifying or personal information. Participants will analyze a public dataset consisting of text documents, using a new information visualization software tool. There are no foreseeable risks to participating.

Benefits of study to the development of new knowledge:

We have designed an information visualization tool for helping analyze text documents. This tool captures and visualizes the history of the analysis process. The purpose of this study is to better understand how analysts use a "Visual History" view, and how that view affects the analysis process. The study can inform the design of future tools to help analysts work more effectively.

Procedures:

The purpose of this experiment is to gather information on how analysts might benefit from access to the visual history of their analysis session. In this study you will be working with CZSaw (the information visualization tool) for 20 minutes to explore an IEEE VAST10 mini-challenge dataset to understand the relationship between the entities in the document collection. The computer screen will be videotaped using a video camera pointing over your shoulder. Your face will not be captured in the video. You are asked to think aloud during the analysis process and your voice will be recorded. **(Continued on 2nd page)**

(Continued from 1st page)
You are allowed to use paper and pen to take notes while you are exploring the dataset. After 20 minutes, you will open the History View and explain the analysis process by using the snapshots in the History View.

At the end, you will be asked to provide suggestions and comments about improving the design of the History View. Your suggestions and comments will be recorded in the video.
The VAST Challenge is a contest, held as part of the IEEE VAST 2010 Symposium. This contest is aimed at research groups, primarily academic groups, to give them a chance to try out their software on datasets that *emulate* real data, but *contain no actual real data*. The VAST challenge includes three minichallenges and one grand challenge. For each minichallenge, an artificial dataset is provided for the contestants to download from the VAST 2010 website. The dataset for minichallenge 1 will be used in this study and includes artifical data about illegal arms dealing that may have taken place during 2008. Contestants were asked to provide a forensic analysis of this activity and to organize their analysis by country.
In this study, we assume that some earlier analysis suggests that a number of meetings took place in Dubai during April 2008 in respect to illegal arms dealing. You are asked to explore the minichallenge 1 dataset to find information about these meetings, such as their dates, locations and participants.

Confidentiality

The identities of all people who participate will remain anonymous and will be kept confidential. After your participation, all electronically recorded data will be immediately stored on a secure server located in the secure server room at SFU Surrey. After the study, identifiable data will be stored securely on a CD-ROM in a locked cabinet. All data from individual participants will be coded so that their anonymity will be protected in any project reports and presentations that result from this work. All of the collected data will be destroyed after 2 years.

Contact information about the project

If you have any questions or require further information about the project you may contact Nazanin Kadivar (nka23@sfu.ca).

Contact for information about the concerns, complaints and rights of research subjects
If you have any concerns or complaints about your treatment or rights as a research subject, you may contact Dr. Hal Weinberg, Director, Office of Research Ethics at hal_weinberg@sfu.ca or 778-782-6593.

(Continued on 3rd page)
(Continued from 2nd page)

Consent

Your signature on this form will signify that you have received the above information which describes the procedures, possible risks, and benefits of this research study, that you have received an adequate opportunity to consider the information in the document,

and that you voluntarily agree to participate in the study. We are very grateful for your participation.

My participation is entirely voluntary and I may refuse to participate or withdraw from the study at any time and it will have no adverse effects on my grades or evaluation in my courses. I may obtain copies of the results of this study, upon its completion, by contacting the researcher named above.
I certify that I understand the procedures to be used and that I understand this document, and that I have been able to receive clarification of any aspects of this study about which I have had questions.

Print Name: _____

Signature: _____

Date **(use format MM/DD/YYYY): _____**

REFERENCE LIST

Amar, R., J. Eagan, and J. Stasko. 2005. Low-level components of analytic activity in information visualization. *Symposium on Information Visualization (InfoVis '05)*: 111-117.

Ayers, E. Z, and J. T Stasko. 1995. Using graphic history in browsing the World Wide Web. *Proceedings of the Fourth International World-Wide Web Conference*: 11-14.

Brodlie, K., A. Poon, H. Wright, L. Brankin, G. Banecki, and A. Gay. 1993. GRASPARC: A problem solving environment integrating computation and visualization. In *Proceedings of the 4th conference on Visualization'93*, 102–109.

Callahan, S. P, J. Freire, E. Santos, C. E Scheidegger, C. T Silva, and H. T Vo. 2006. Managing the evolution of dataflows with vistrails. *IEEE Workshop on Workflow and Data-Flow for Scientific Applications (SciFlow)*: 71-75.

Card, S. K, P. Pirolli, M. Van Der Wege, J. B Morrison, R. W Reeder, P. K Schraedley, and J. Boshart. 2001. Information scent as a driver of Web behavior graphs: results of a protocol analysis method for Web usability. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 498–505.

Collaud, G., J. Dill, C. V Jones, and P. Tan. 1996. The Continuously Zoomed Web - a Graphical Navigation Aid for WWW. *Proceedings of IEEE Visualization '96, Late Breaking Hot Topics* 5, no. 1: 1-3.

Derthick, M., and S. F Roth. 2001. Enhancing data exploration with a branching history of user operations. *Knowledge-Based Systems* 14, no. 1: 65–74.

Dill, J., L. Bartram, A. Ho, and F. Henigman. 1994. A continuously variable zoom for navigating large hierarchical networks. In *Systems, Man, and Cybernetics, 1994.'Humans, Information and Technology'., 1994 IEEE International Conference on*, 1:386–390.

Doemel, P. 1995. WebMap: a graphical hypertext navigation tool. *Computer Networks and ISDN Systems* 28, no. 1: 85–97.

Dunsmuir, D., M. Z Baraghoush, V. Chen, M. E Joorabchi, M. E Joorabchi, S. Alimadadi, E. Lee, et al. 2010. CZSaw, IMAS & Tableau: Collaboration among Teams. *Proceedings of IEEE Visual Analytics Science and Technology 2010*: 267-

268.

Edwards, W. K, T. Igarashi, A. LaMarca, and E. D Mynatt. 2000. A temporal model for multi-level undo and redo. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, 31–40.

Fisher, B., M. Agelidis, J. Dill, P. Tan, G. Collaud, and C. Jones. 1997. CZWeb: Fish-eye views for visualizing the world-wide web. *ADVANCES IN HUMAN FACTORS ERGONOMICS* 21: 719–722.

Frécon, E., and G. Smith. 1998. WEBPATH-a three dimensional Web history. In *Information Visualization, 1998. Proceedings. IEEE Symposium on*, 3–10.

Furnas, G. W. 1986. *Generalized fisheye views*. Vol. 17. 4. ACM.

Furnas, G. W. 1999. The FISHEYE view: A new look at structured files. *Readings in information visualization: using vision to think*: 312–330.

Gotz, D., and M. X Zhou. 2009. Characterizing users' visual analytic activity for insight provenance. *Information Visualization* 8, no. 1: 42–55.

Haraty, Mona. 2010. Supporting Exploratory Information Seeking. Master of Science, Canada: Simon Fraser University, Fall.

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

Hightower, R. R, L. T Ring, J. I Helfman, B. B Bederson, and J. D Hollan. 1998. Graphical multiscale Web histories: a study of padprints. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems: links, objects, time and space—structure in hypermedia systems*, 58–65.

Jankun-Kelly, T. J., K. L Ma, and M. Gertz. 2007. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*: 357–369.

Kadivar, N., V. Chen, D. Dunsmuir, E. Lee, C. Qian, J. Dill, C. Shaw, and R. Woodbury. 2009. Capturing and supporting the analysis process. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, 131–138.

Kashihara, A., Y. Satake, and J. Toyoda. 1998. A History Visualization for Learning-by-Exploration in Hypermedia on WWW. *Proceedings of WebNet 98*: 497–502.

Klemmer, S. R, M. Thomsen, E. Phelps-Goodman, R. Lee, and J. A Landay. 2002.

Where do web sites come from?: capturing and interacting with design history. In *Proceedings of CHI 2002, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 1-8.

Kreuseler, M., T. Nocke, and H. Schumann. 2004. A history mechanism for visual data mining. *IEEE Symposium on Information Visualization*: 49-56.

Kurlander, D., and S. Feiner. 1998. Editable graphical histories. In *IEEE Workshop on Visual Languages,*, 127–134.

Lipford, H. R, F. Stukes, W. Dou, M. E Hawkins, and R. Chang. 2010. Helping Users Recall Their Reasoning Process. *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*: 187-194.

Ma, K. L. 1999. Image graphs—a novel approach to visual data exploration. In *Proceedings of the conference on Visualization'99: celebrating ten years*, 81–88.

Mackinlay, J., P. Hanrahan, and C. Stolte. 2007. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*: 1137–1144.

Mackinlay, J. D, G. G Robertson, and S. K Card. 1991. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, 173–176.

Meng, C., M. Yasue, A. Imamiya, and X. Mao. 1998. Visualizing histories for selective undo and redo. In *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, 459–464.

Pirolli, P., and S. Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, 2005:2–4.

Plaisant, C., A. Rose, G. Rubloff, R. Salter, and B. Shneiderman. 1999. The design of history mechanisms and their use in collaborative educational simulations. In *Proceedings of the 1999 conference on Computer support for collaborative learning*, 348–359.

Prakash, A., and M. J Knister. 1994. A framework for undoing actions in collaborative systems. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1, no. 4: 295–330.

Robinson, A. C, C. Weaver, and G. Center. 2006. Re-Visualization: Interactive Visualization of the Process of Visual Analysis. In *Workshop on Visualization, Analytics & Spatial Decision Support at the GIScience conference*.

Shipman, F. M, and H. Hsieh. 2000. Navigable history: a reader's view of writer's time. *New review of hypermedia and multimedia* 6, no. 1: 147–167.

Shrinivasan, Y. B, and J. J van Wijk. 2008a. Supporting exploration awareness for visual analytics. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, 185–186.

Shrinivasan, Y. B, and J. J van Wijk. 2008b. Supporting the analytical reasoning process in information visualization. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 1237–1246.

Springmeyer, R. R, M. M Blattner, and N. L Max. 1992. A characterization of the scientific data analysis process. In *Visualization, 1992. Visualization'92, Proceedings., IEEE Conference on*, 235–242.

Stasko, J., C. Görg, and Z. Liu. 2008. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization* 7, no. 2: 118–132.

Tabard, A., W. Mackay, N. Roussel, and C. Letondal. 2007. PageLinker: integrating contextual bookmarks within a browser. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 337–346.

Thomas, J. J, K. A Cook, Institute of Electrical, and Electronics Engineers. 2005. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society.

Turetken, O., and R. Sharda. 2007. Visualization of web spaces: state of the art and future directions. *ACM SIGMIS Database* 38, no. 3: 51–81.

Vitter, J. S. 1984. US&R: A new framework for redoing. *IEEE software* 1, no. 4: 39–52.

Waterson, S. J, J. I Hong, T. Sohn, J. A Landay, J. Heer, and T. Matthews. 2002. What did they do? understanding clickstreams with the webquilt visualization system. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, 94–102.

Wexelblat, A., and P. Maes. 1999. Footprints: history-rich tools for information foraging. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, 270–277.