View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarWorks @ Georgia State University

Georgia State University
ScholarWorks @ Georgia State University

Computer Science Dissertations

Department of Computer Science

7-17-2009

A Task-Centered Visualization Design Environment and a Method for Measuring the Complexity of Visualization Designs

Xiaoyuan Suo

Follow this and additional works at: https://scholarworks.gsu.edu/cs_diss

Part of the Computer Sciences Commons

Recommended Citation
Suo, Xiaoyuan, "A Task-Centered Visualization Design Environment and a Method for Measuring the Complexity of Visualization Designs." Dissertation, Georgia State University, 2009.
https://scholarworks.gsu.edu/cs_diss/41

This Dissertation is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

A TASK-CENTERED VISUALIZATION DESIGN ENVIRONMENT AND A METHOD FOR MEASURING THE COMPLEXITY OF VISUALIZATION DESIGNS

by

XIAOYUAN SUO

Under the Direction of Ying Zhu

ABSTRACT

Recent years have seen a growing interest in the emerging area of computer security visualization which is about developing visualization methods to help solve computer security problems. In this thesis, we will first present a method for measuring the complexity of information visualization designs. The complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, number of visual units, and the efficiency of visual search. This method is designed to better assist fellow developers to quickly evaluate multiple design choices, potentially enables computer to automatically measure the complexity of visualization data.

We will also analyze the design space of network security visualization. Our main contribution is a new taxonomy that consists of three dimensions – data, visualizations, and tasks. Each dimension is further divided into hierarchical layers, and for each layer we have identified key parameters for making major design choices. This new taxonomy provides a comprehensive framework that can guide network security visualization developers to systematically explore the design space and make informed design decisions. It can also help developers or users systematically evaluate existing network security visualization techniques and systems. Finally it helps developers identify gaps in the design space and create new techniques.

Taxonomy showed that most of the existing computer security visualization programs are data centered. However, some studies have shown that task centered visualization is perhaps more effective. To test this hypothesis, we propose a task centered visualization design framework, in which tasks are explicitly identified and organized and visualizations are constructed for specific tasks and their related data parameters. The center piece of this framework is a task tree which dynamically links the raw data with automatically generated visualization. The task tree serves as a high level interaction technique that allows users to conduct problem solving naturally at the task level, while still giving end users flexible control over the visualization construction. This work is currently being extended by building a prototype visualization system based on a Task-centered Visualization Design Architecture.

INDEX WORDS: Information visualization, Complexity, Evaluation

A TASK-CENTERED VISUALIZATION DESIGN ENVIRONMENT AND A METHOD FOR MEASURING THE COMPLEXITY OF VISUALIZATION DESIGNS

by

XIAOYUAN SUO

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2009

Copyright by
Xiaoyuan Suo
2009

A TASK-CENTERED VISUALIZATION DESIGN ENVIORNMENT AND A METHOD FOR MEASURING THE COMPLEXITY OF VIUSALIZATION DESIGNS

by

XIAOYUAN SUO

Committee Chair: Ying Zhu

Committee: G. Scott Owen
Raheem Beyah
Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

This thesis consists of three parts: complexity analysis of the security visualization designs, survey of the current security visualization systems and a description of the system that has been developed.

1.1 Complexity Analysis

Today information overload is a major challenge in many areas. For example, computer security professionals need to process data from a myriad of sources, including network devices, firewalls, intrusion detection programs, vulnerability scanners, and operating systems. This overabundance of information incurs heavy cognitive load on the users. Data visualization has the great potential to alleviate the heavy cognitive load associated with processing this overabundance of information. However, poorly designed visualizations can be counterproductive or even misleading. Therefore, visualizations must be carefully designed and evaluated.

The evaluations of visualization generally involve user studies. Common measures of visualization include task completion time, error rate, or subjective satisfaction. However, these are largely black-box approaches and the results often do not explain whether or why certain features of visualizations cause the performance or usability problems. Besides, user studies are often difficult to manage and can only be conducted after a program has been developed.

In this thesis, we developed an alternative evaluation method – complexity analysis, which systematically evaluates a set of factors that influence the efficient processing of visual information. The proposed method is grounded in well established psychological theories [1-5] and the outcome of this analysis serves as an indicator of the cognitive load associated with comprehending a visualization design. To my knowledge, this research is the first attempt to systematically evaluate the complexity of information visualization.

The complexity analysis is particularly useful during the visualization design stage before any user study can take place. It allows designers to quickly evaluate multiple visualization designs in terms of their complexity. The results of the complexity analysis not only provide guidance to the design but also help generate hypotheses for user studies to verify. It helps to reduce the cost and improve the quality of visualization design. It also helps to reduce the cost and improve the quality of visualization design. This rather focused method could be combined with other heuristic evaluation methods, especially the user studies.

The evaluation method can be applied on the existing security visualization systems. Results obtained from the evaluation can be formulated and categorized based on their score or purposes; we therefore build a matrix for the results. The matrix can then be applied as guidance for future designs of the same purposes.

1.2 Task Centered Framework for Computer Security Data Visualization

A fundamental question for visualization design is what makes visualizations effective? There have been different answers to this question. Some researchers take a more data-centric view and suggest that effectiveness depends on the accurate interpretation of presented data [6-8], or a matching between data structure and visual structure [9, 10]. However, a number of psychological studies have also shown that the effectiveness of visualization is task specific [11, 12].

In this thesis, we propose a new task centered framework of visualization and apply it to computer security visualization. In our framework, a visualization system is optimized for specific tasks by mapping the task related parameters to the visual elements that have high accuracy, utility, and efficiency ratings.

Before visualizations are created, users specify tasks and their associated parameters. This process is essentially a task complexity analysis. Knowing the data parameters associated with a task helps users consciously control the complexity of the tasks and correlate task complexity and visualization complexity. This new framework provides a different way for users to interact with data set and potentially will provide new insights into how visualization can be better constructed to serve users' specific tasks.

1.3 Design Space of Current Security Visualization

Recent years have seen a growing interest in the emerging area of computer security visualization, which is about developing visualization methods to address computer security problems. In contrary, there has been little research in systematic analysis of the design space of computer security visualization; regardless of its many obvious benefits. In this paper, we analyze the design space of network security visualization by developing a new taxonomy. Within this taxonomy framework, we identify key parameters and classes that define the structure of this design space. Using taxonomy to define a design space is a common method that has been used in the area of computer security [13, 14], information visualization [15-17], and computer-human interaction [15, 18].

Our in-detailed analysis intends to provide the fellow developers with a better understanding of the structure of the design space. This would help network security visualization developers understand the tasks at a more detailed level and understand how various visualization techniques address the tasks. First the analysis can help to explore the design space and make informed design decisions. Second, it helps developers or users systematically evaluate existing techniques and systems. Third, it helps developers identify gaps in the design space and create new techniques.

Our main contribution is a taxonomy that consists of three dimensions – data, visualizations, and tasks. Each dimension is further divided into hierarchical layers, and for each layer we have identified key parameters for making major design choices, (figure 1).

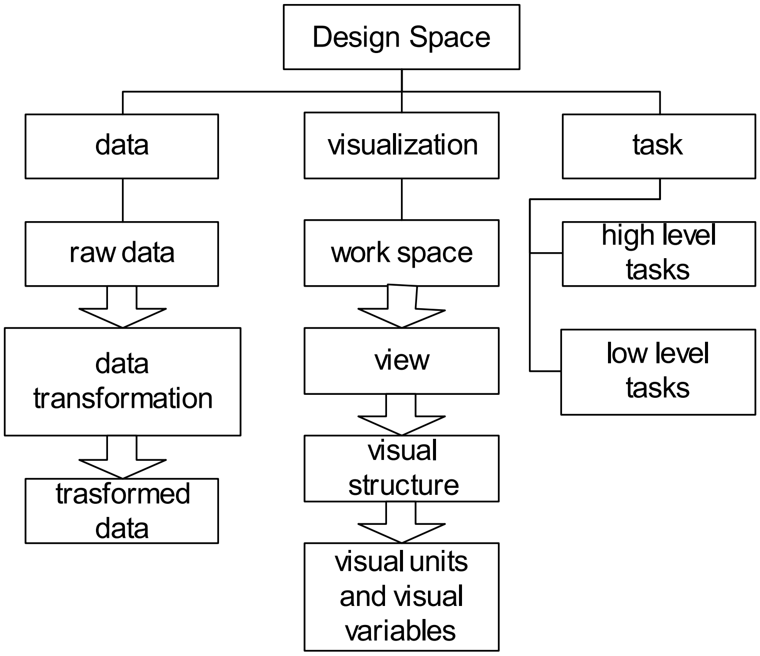


Figure 1: Three major dimensions that determine the design space of network security; and their sub-hierarchical layers.

We define the design space of network security visualization in three dimensions: data, visualization, and task. The *data* dimension consists of two layers: raw data and transformed data. The *visualization* dimension is divided into five layers: workspace, view, visual structure, and visual unit and visual variable. The *task* dimension consists of two layers: high level task and low level task, (figure 1).

In the next section, we will discuss each dimension of the design space in sequence. We will explain the relationship between different layers, identify key parameters, and list categories of possible design choices for each parameter.

Since design choices made in one dimension often affect the design choices in other dimensions, we will also discuss the relationship of a parameter or category with parameters or categories in other dimensions.

2 Background and Related Works

2.1 Complexity Analysis and Heuristic Evaluations

Heuristic evaluation is a discount usability engineering method for quick, cheap, and easy evaluation of a user interface design. Heuristic evaluation is a well known discount evaluation technique in human-computer interaction (HCI) but has not been utilized in information visualization (InfoVis) to the same extent [19].

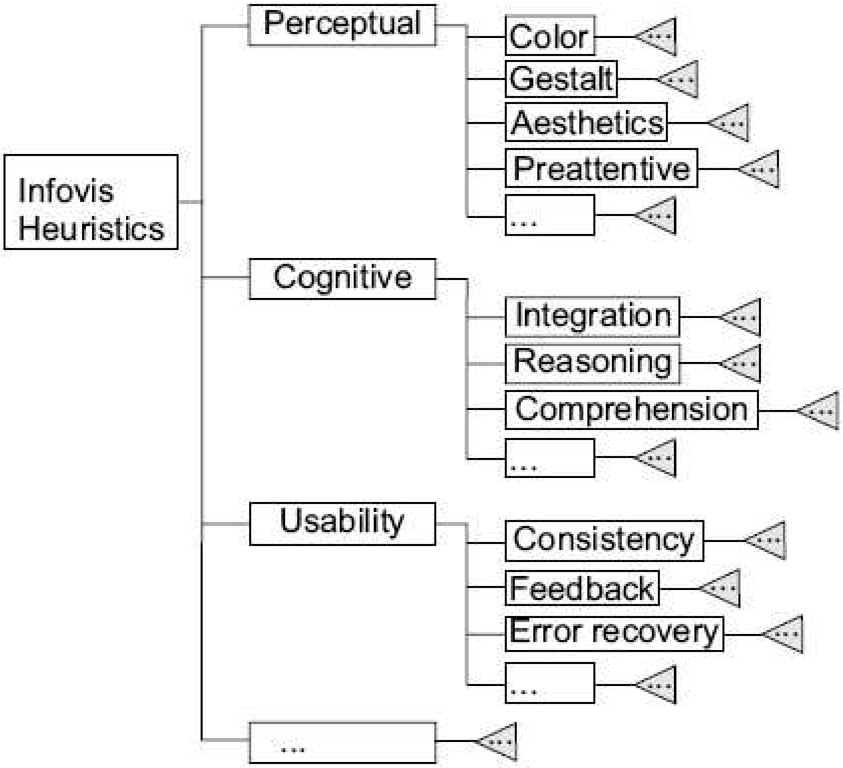


Figure 2: Heuristic evaluation tree [19].

The heuristic tree is shown above in figure 24; the three layers of the tree represent an organization intended to help evaluators. The authors of the work [19] also used existing evaluation methods and divided them into three different sets; the sets are shown in the table below.

Table 1: Heuristics applied in evaluation.

set	Heuristics
Zuk and Carpendale's *Selection of perceptual and cognitive heuristics [20]*	Ensure visual variable has sufficient length [20-22] *Selection of perceptual and* Don't expect a reading order from color [20-22] *cognitive heuristics [20]* Color perception varies with size of colored item [20-22] Local contrast affects color & gray perception [20, 22] Consider people with color blindness [20, 22, 23] Pre attentive benefits increase with field of view [20-22, 24] Quantitative assessment requires position or size variation [20, 21] Preserve data to graphic dimensionality [20, 21, 25] Put the most data in the least space [20, 25] Remove the extraneous (ink) [20, 25] Consider Gestalt Laws [20, 22] Provide multiple levels of detail [20, 22, 25] Integrate text wherever relevant [20, 22, 25] Overview first [26]
Shneiderman's Overview first *"Visual Information-Seeking Mantra"[26]*	Zoom and filter [26] Details on demand [26] Relate [26] Extract [26] History [26]
Amar and Stasko's Expose uncertainty [27] *Knowledge and task-based framework [27]*	Expose uncertainty [27] Concretize relationships [27] Determination of Domain Parameters [27] Multivariate Explanation [27] Formulate cause & effect [27] Confirm Hypotheses [27]

The approach proposed in the paper [19] provided useful results and revealed some characteristics, such as redundancy and conflict that may be generally useful when comparing different heuristics. The work used three sets of previously published heuristics (listed in the table above) to access a visual decision support system that is used to examine simulation data. The meta-analysis shows that the evaluation process and results have a high dependency on the heuristics and the types of evaluators chosen.

Shneiderman et, Al. [28] proposed a method called multi-dimensional in-depth long-term case studies. The steps discussed include: Specify research goals; Identify 3-5 users; document the current methods or tools being advanced by the new tool; determine user's expertise; establish observation schedule, instrument the tool to record usage data; make recording user comments available; provide training; conduct interviews; encourage users to continue use the best possible tool; modify tool as needed; document success and failures.

The work [28] inspired us on our future work. Our work differs from the traditional visualization work by providing the user with a visualization building tool rather than one or two pre-fabricated visualization, therefore user studies should be case dependent. In addition to the general steps as Shneiderman et, al. [28] has proposed, we feel the selection of evaluators should also be a critical step, since each evaluator may have specific need and experience on certain type of visualization.

When formal laboratory user studies cannot accommodate exploratory phase of research, expert reviews could be one of the possible alternatives. Tory et, al. [29] suggested that selection of experts is critical. The author suggested that chosen experts should have the quality of strong communication skills; experience conducting usability inspections; and experience with data display (not just usability). With several case studies, the author recommended: 1. including experts with experience in data display as well as usability, and 2. developing heuristics based on visualization guidelines as well as usability guidelines. The author also concluded that expert reviews should not be used exclusively, since experts might not fully predict end-user actions.

There are other types of evaluation metrics. The work by Brath [30] proposed one type of information visualization metrics, which intend to help designers create and evaluate 3D information visualizations. Some of the criteria included are:

1. Number of data points and data density. Key factors are data density and bounds (lower bounds and upper bounds). Data density is defined as the number of data points / number of pixels in the display. The author also proposed that visualizations with less than 500 data points are questionable visualizations. However, the lower bound depends on the use and interaction with the application.

2. Number of dimensions and cognitive overhead. Number of dimensions directly affects the complexity of the visualization. The author also mentioned maximum of the number of dimensions for each separable task representation; it depend on the definition of task.

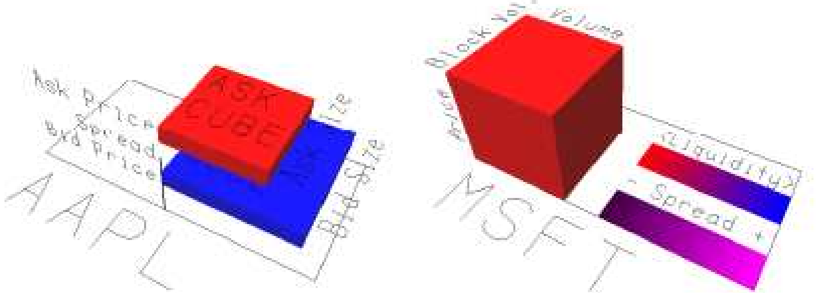


Figure 3: Number of dimensions representation.

The figure on the left hand side has a very specific but easy to understand mapping of data attributes to different visual objects. The figure on the right hand side is general but difficult to understand mapping of data to 5 different properties of a cube. [30]

3. Occlusion percentage. Occlusion percentage is defined as number of data points completely obscured / number of data points.

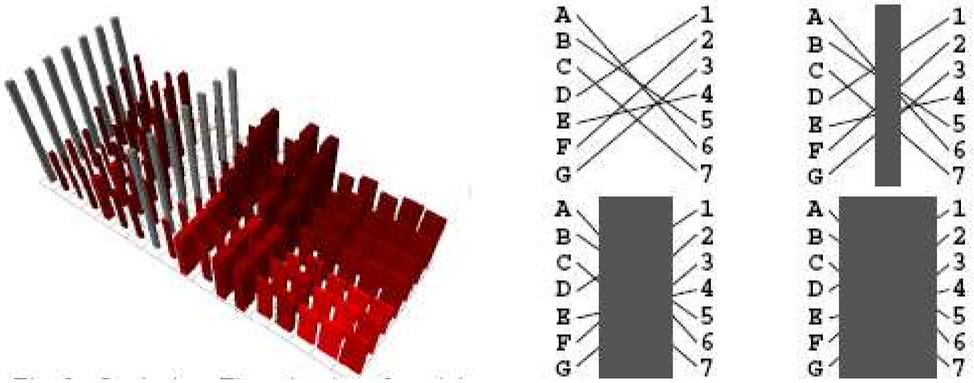


Figure 4: Occlusion percentage.

Figure(4) left has a lot of partial occlusion, but sufficient redundancy in the representation for comprehension. The figure on the right has increasing partial occlusion obscures the relationships [30]

4. Reference context and percentage of identifiable points. The two figures below showed the difference in using proper references in the graphs.

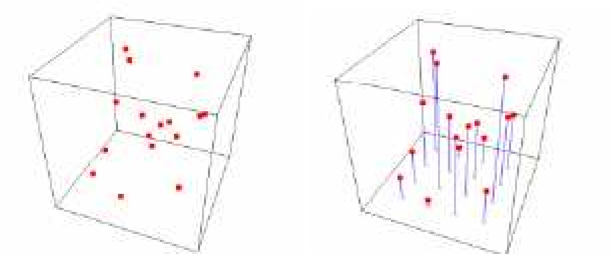


Figure 5: Reference context.

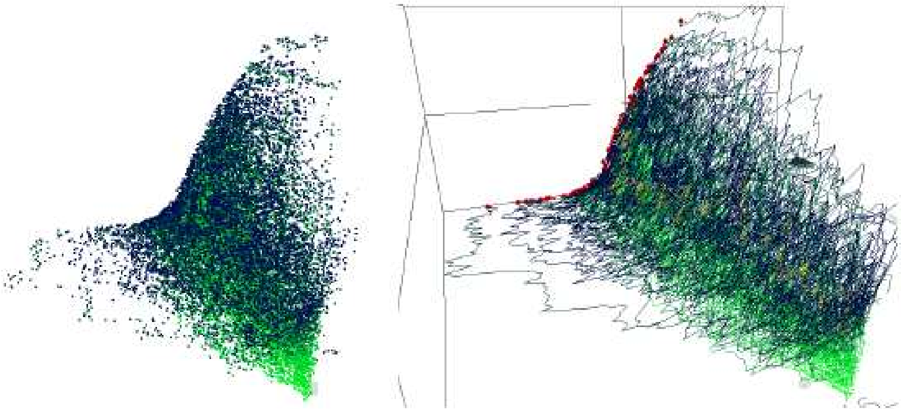
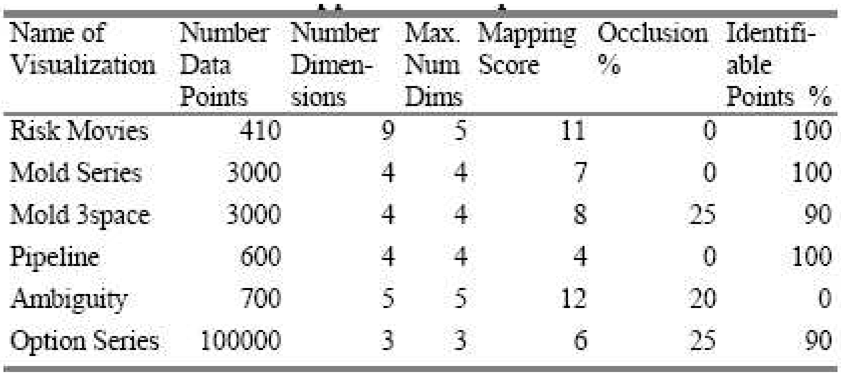


Figure 6: Reference context. (Scatter plot).

In the figure above, scatter points are difficult to locate. Drawing lines from scatter point to plane locates the points. [30] Each of the separable dimensions will be given a score based on 4 different categories: n-to-1 mapping, 1-to-1 general mapping, 1-to-1 intuitive mapping and preexisting understood representations. A sample table of measures as applied to sample visualizations is shown in the figure 7 below.

In the figure above, the scatter points on the left figure are difficult to locate in 3D space − the depth is not known. The figure on the right hand side has drawing lines from scatter point to scatter point as a set of series helps to visually locate the points and establish the field. [30]



Name of Visualization	Number Data Points	Number Dimen-sions	Max. Num Dims	Mapping Score	Occlusion %	Identifi-able Points %
Risk Movies	410	9	5	11	0	100
Mold Series	3000	4	4	7	0	100
Mold 3space	3000	4	4	8	25	90
Pipeline	600	4	4	4	0	100
Ambiguity	700	5	5	12	20	0
Option Series	100000	3	3	6	25	90

Figure 7: Table of measures as applied to sample visualizations [30].

Several visualization researchers have touched on the issues of complexity in visual display, but none of them have considered it in a systematic way. Bertin [21] and Trafton, et al. [3] have used number of dimensions as a measure for the complexity of visual displays, and considered visualizations with more than three variables to be complex. Brath [30] has proposed a heuristic method to measure the effectiveness of the mapping from the data dimension to the visual dimension by classifying the visual mappings into one of the four categories. My evaluation method is more comprehensive than the previous methods and considers many factors that are not considered by previous research.

The proposed complexity analysis is a type of heuristic evaluation methods of information visualization [31-34]. Existing heuristic evaluation methods are largely based on heuristic rules or guidelines that come from intuition. But as Scaife and Rogers [35] point out, the effectiveness of visualization cannot be evaluated by intuition, but rather through a set of interdependent factors. The proposed complexity analysis is an attempt to address this issue – it systematically evaluates a set of factors that influence the efficiency of visualization comprehension. More importantly, my evaluation process is grounded in well established psychological theories.

2.2 Cognitive Fit Theory

Cognitive fit theory was developed by Iris Vessey [36].Cognitive fit is an investigation of the fit of technology to task, the user's view of the fit between technology and task, and the relative importance of each to problem-solving or decision-making performance [36].

With a set of 128 MBA students in two identical, repeated measures designs, the authors produced the results:

- Performance improved markedly for symbolic tasks when the problem representation matched the task
- Performance effects also resulted from matching specific problem-solving skills to the problem representation and the task, and to a lesser extent when the skills matched the task alone.

- The incremental effects of matching skills to the problem representation and/or the task were small compared with the primary effects of cognitive fit-that of matching problem representation to task.
- A large proportion of problem solvers have insight into the concept of supporting tasks with certain types of problem representation and vice versa
- Participants preferred to use tables rather than graphs; they also preferred to solve symbolic rather than spatial problems.
- Finally, the problem representation more significantly influenced the mental representation than did task conceptualization. [36]

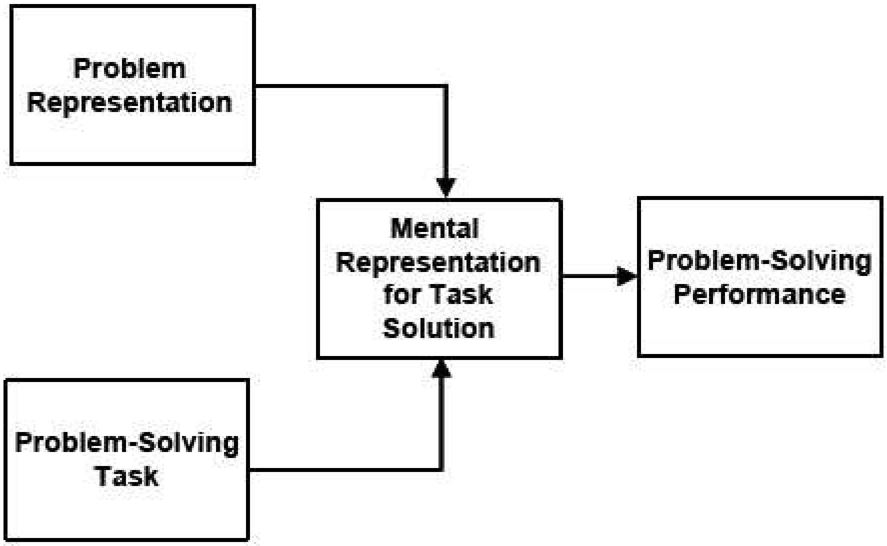

Problem Representation
Mental Representation for Task Solution
Problem-Solving Performance
Problem-Solving Task

Figure 8: Cognitive fit in problem solving [37].

Cognitive Fit Theory states that task performance improves when the problem representation match the cognitive characteristic of the task. In this thesis, we described a framework that supports visual problem solving based on this theory. The framework consists of two major components: a task window and a data window.

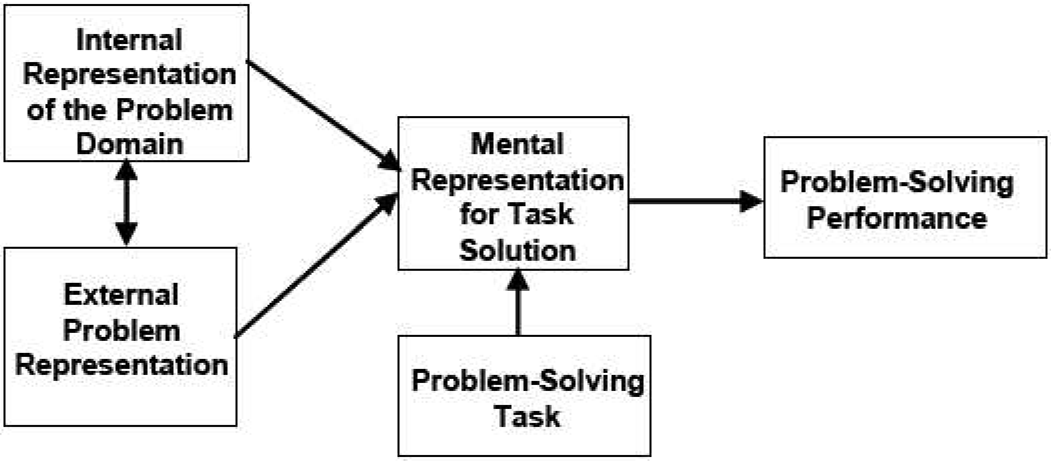

Internal Representation of the Problem Domain
External Problem Representation
Mental Representation for Task Solution
Problem-Solving Task
Problem-Solving Performance

Figure 9: Extended cognitive fit model [37].

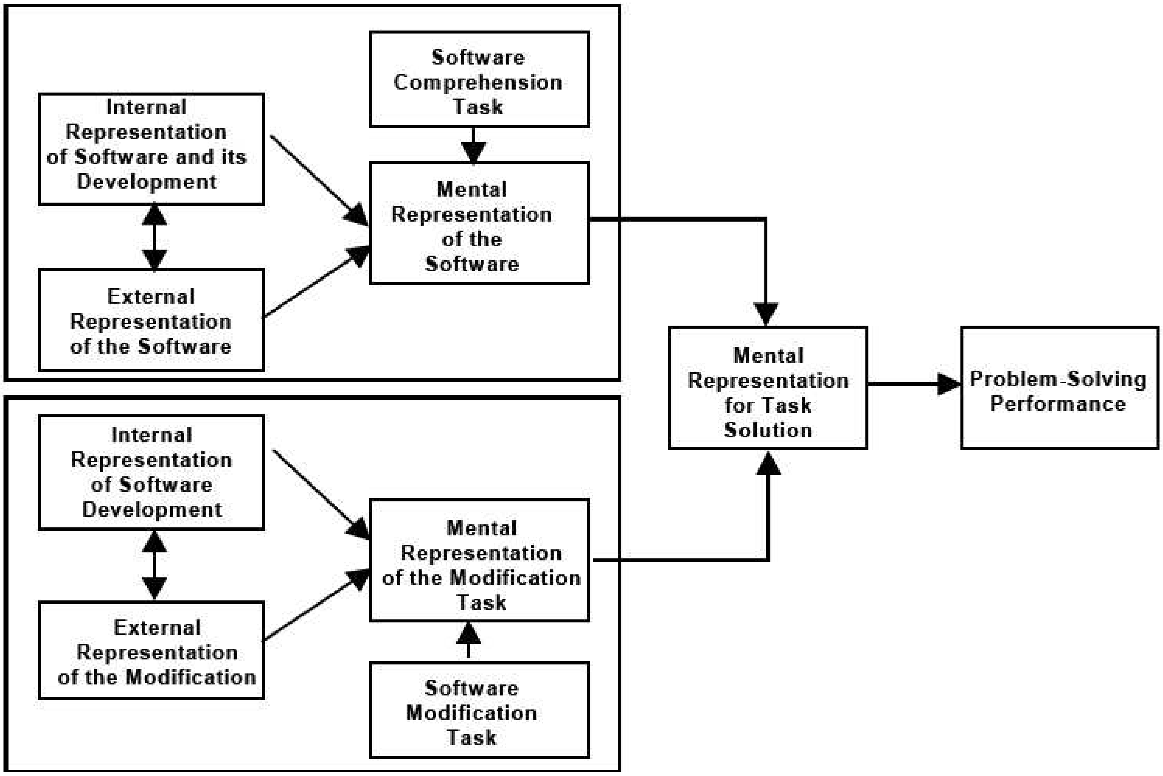

Software Comprehension Task
Internal Representation of Software and its Development
Mental Representation of the Software
External Representation of the Software
Internal Representation of Software Development
Mental Representation of the Modification Task
External Representation of the Modification
Software Modification Task
Mental Representation for Task Solution
Problem-Solving Performance

Figure 10: General model of interacting tasks in software maintenance [37].

In the task window, users' problem solving process and strategies are expressed in the form of a dynamic and interactive task tree, which contains a hierarchical set of tasks and sub-tasks. The data window contains multiple frames that are organized as a tree map. The tree map structure is synchronized with the task tree so that each task node on the task tree is dynamically linked to a data frame. In each data frame, data is presented in either visual or non-visual format based on the cognitive characteristic of the corresponding task. As users explore various problem solving strategies by editing the task tree, the tree map in the data window is automatically updated for the best "cognitive fit". This problem solving framework is particularly useful for complex problem solving with large amount of data. We present a computer security data visualization tool as an example of the proposed framework.

2.3 Task-Centered Visualization Designs

Many visualization designs have been proposed for computer security analysis. Noted examples include TNV [38], IDS RainStorm [39], PortVis [40], etc. Most of these designs, however, are prefabricated visualizations that cannot be easily reconfigured by users for different tasks. An implicit assumption is that users can use interaction techniques to customize data visualization for different tasks. While interaction is essential for making visualization usable, two important issues need to be addressed. First, for most existing visualization systems it is often not clear what specific tasks they are designed for. As a result, users may use the visualization for unintended tasks. Second, most existing visualization systems provide only low level interaction techniques, such as zooming, panning, that are restrained by the predefined visualization structure. They may be suitable for problem solving process with relatively stable procedure and task structure. However, many complex problem solving process are not so well defined. In many cases, problem solving is a process of searching in the solution space. This means that users may constantly testing different hypotheses and apply different strategies. The task structure may keep changing during the problem solving process. A new set of higher level interaction techniques are needed to support this dynamic problem solving method.

In the 8 vertical axes are shown that represent the 2.5 Class B IP addresses. The thicker horizontal lines between these axes show Class B's starting position. The other horizontal lines show the start and end of each department. Those addresses not in a department are either unallocated or reserved for special use by OIT and other departments. This screenshot shows an entire day's worth of real alarms generated. [39]

Note that all of the available visualization tools are present simultaneously, so it is easy to correlate data and mentally shift between visualizations. Visualization generally begins at the timeline (1), followed by the hour (main) visualization (2). The main visualization contains a circle, which helps users locate the magnification square in its center. Magnifications from the square within the main visualization are shown in (3); a port may be selected from (3) to get the port activity display in (4). Several parameters (5) control the appearance of the main display and port displays. The panel of options in (6) permits the selection of a data source to display, and offers a color-picker for selecting new colors for gradients. [40]

A detailed view and analysis of IDS Rain Storm is displayed in the figure below, this is a screen shot from the original system.

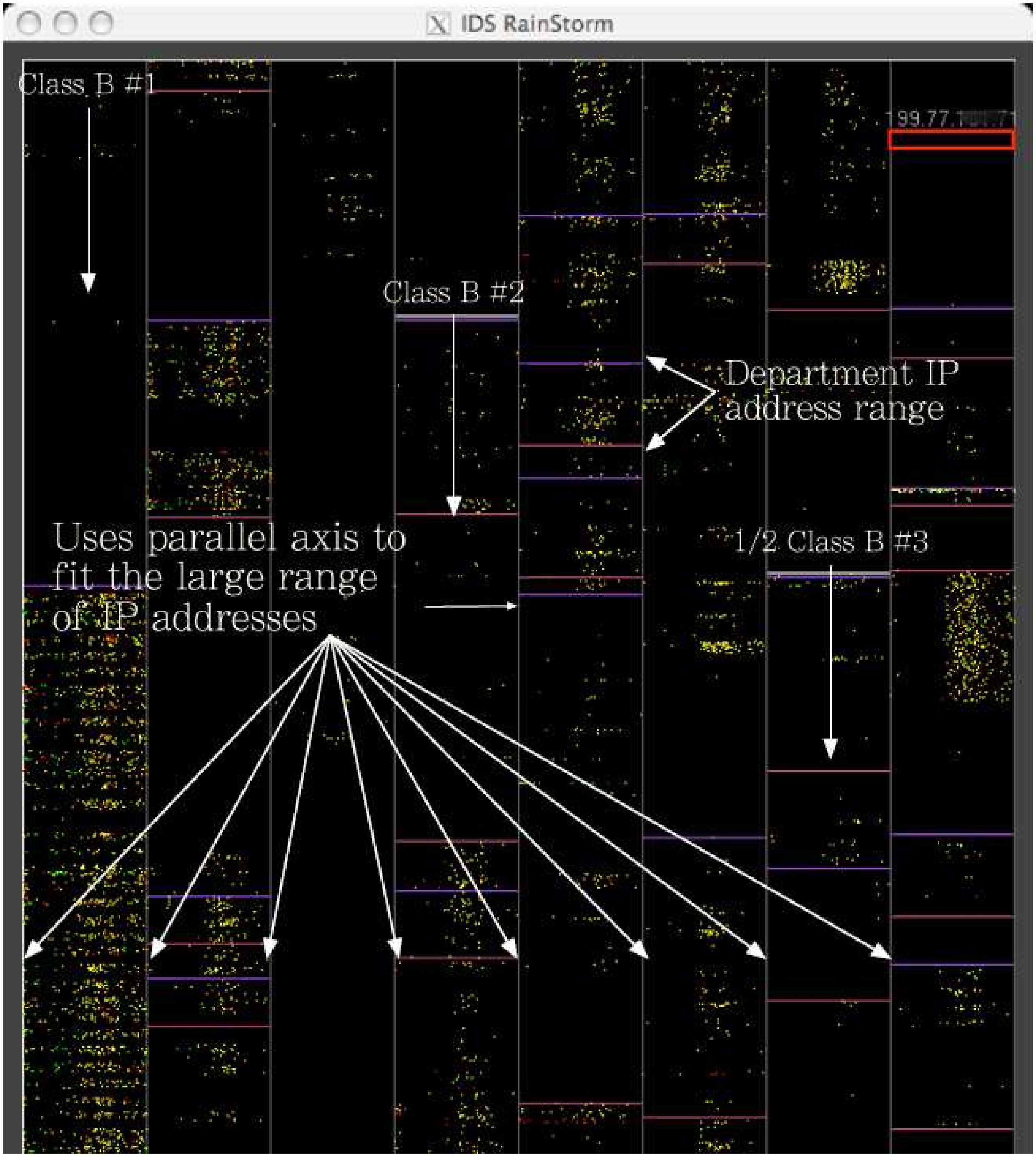
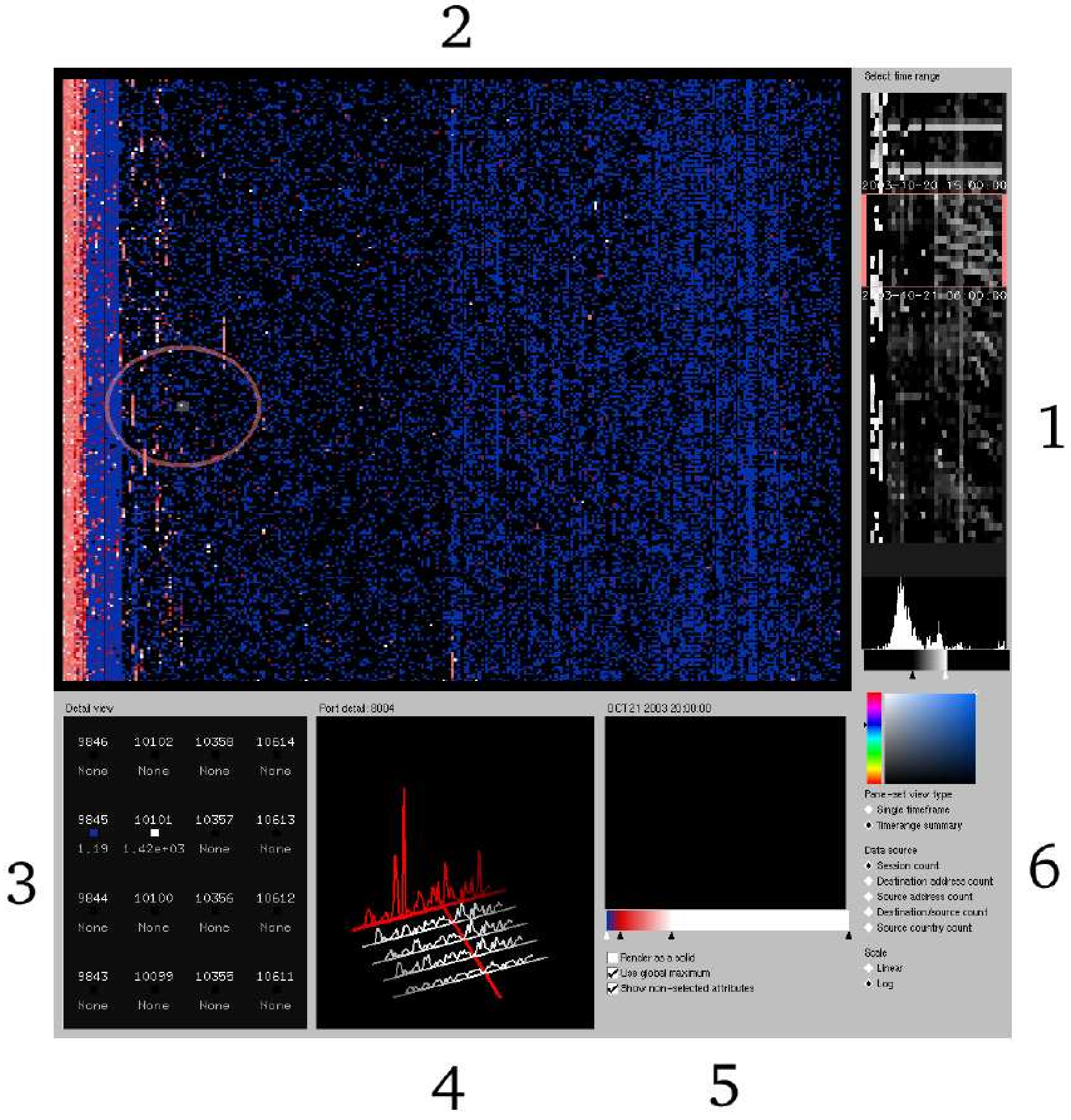

IDS RainStorm
Class B #1
199.77.1[redacted]71
Class B #2
Department IP address range
Uses parallel axis to fit the large range of IP addresses
1/2 Class B #3

Figure 11: IDS RainStorm main view.

Some visualization systems, such as RUMINT [41], do provide a more configurable interface that allow users to assign parameters to different coordinate axes, or choose different types of diagrams. Outside the field of computer security visualization, Tableau Software is noted for its highly flexible and configurable interface that allows users to quickly construct different data visualizations. Another example is Many-Eyes [42], a web site that allows different users to construct different visualizations of the same data set.



Detail view

9846 None	10102 None	10358 None	10614 None
9845 1.19	10101 1.42e+03	10357 None	10613 None
9844 None	10100 None	10356 None	10612 None
9843 None	10099 None	10355 None	10611 None

□ Render as a solid
☑ Use global maximum
☑ Show non-selected attributes

Pane-set view type
○ Single timeframe
● Timerange summary

Data source
● Session count
○ Destination address count
○ Source address count
○ Destination/source count
○ Source country count

Scale
○ Linear
● Log

Figure 12: PortVis.

Although these are powerful user interface techniques, users are still operating at the visualization level. But most end users would prefer to operate at a higher level of thinking – ask questions, test hypotheses, etc. For end users, constructing and configuring visualization is a secondary activity to their primary tasks. Again, we need a higher level interaction technique to help end users operate at the level of tasks.

The research presented in this thesis is an attempt to address this issue. The central component of the proposed visualization framework is a task tree that is dynamically linked to data visualizations and data tables.

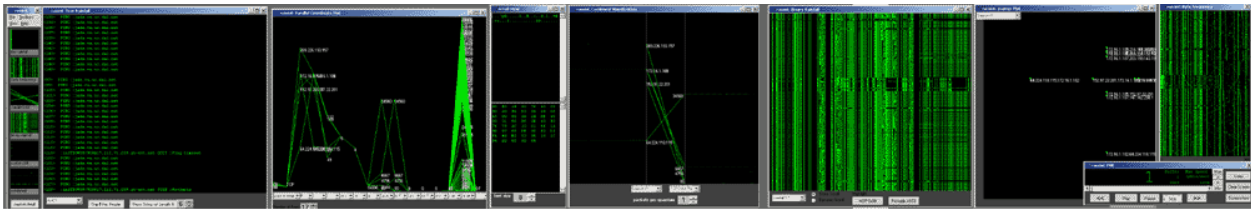


Figure 13: Rumint [43].

Users operate by constructing and maintaining a task tree. A frame of data visualization is created automatically (or semi-automatically) for each task on the task tree, with the support of a visualization engine.

2.4 Automated Visualization Designs

Automated layout of presentation is becoming increasingly important when it comes to usability and the vast amount of data being presented. Effective layout is one of the most important aspects of creating an information presentation. Majority of layouts today are done "by hand" [44]; it is typically done by a "human designer" or "layout expert"; such process can be very expensive and time-costly. Automated layout designs, in the recent years, have been seen in almost all contemporary user interface designs. Not only does it ease the programmers and layout designers' work, it also gives the end user more flexibility to deal with their specific tasks.

In the early 1990's, COMET [45] was developed at the Columbia University; it is being used in the field of maintenance and repair domain for a military radio receiver-transmitter. COMET generates multimedia explanations that instruct the user in how to carry out diagnostic tests. User may interact with COMET by choosing from a simple menu, or during the symptom diagnosis, the user can request an explanation for any diagnostic procedure the system specifies must be carried out. Even though COMET targets only in a small application domain, but the research prototype provided us with immersive insights for the knowledge-based user interfaces designs.

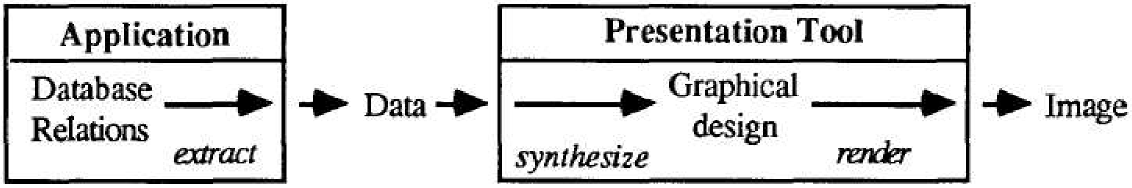


Figure 14: A linear model for generating presentations.

Another very early work by Mackinlay et, al. [6] can date back to the 1980's. The work developed an application-independent presentation that automatically designs effective graphical presentations (such as bar charts, scatter plots, and connected graphs) of relational information. Artificial intelligence techniques are used to implement a prototype presentation tool called APT (A Presentation Tool), which is based on the composition algebra and the graphic design criteria [6]. Artificial intelligence remains to be the similarity among all early works of automated visualization; it is mainly being used for choosing an appropriate graphical presentation of relational data. Graphic design issues are an important concern of user interface design. The work discussed a few important issues at the end, including user interface management; adapting dialogue specifications appropriate to the observed skill level of users.

This simplified model, which does not include feedback loops that are required for difficult design problems, describes the fundamental process of generating a graphical presentation. A graphical design synthesized by a presentation tool describes the basic structure and meaning of a graphical presentation. The rendering process fills in the details that are required to form the final image. [6]

Autovisual, designed by Beshers et, al. [46] is a rule-based system that designed to implement the n-Vision's virtual worlds. The user specifies the visualization task, rather than a particular visualization; and then Autovisual generates an interactive virtual world appropriate for the task. Autovisual is an early prototype. Among the issues it does not yet address if certain aspects of the relation are being visualized. Autovisual also does not respond to changes in visualization tasks by reusing or modifying an existing visualization, rather than creating a new one.

The user studies and analysis by Komlodi et, al. [47] leads the later design of TNV (recall from the previous section) [38]. This work gathered feedbacks from ID analysts' daily activities in order to understand their routine work practices and the need for designing information visualization tools. A three-phase process model that frames corresponding requirements for information visualization tools was also developed. The three phases are: monitoring, analysis and diagnosis, and response. Current security visualization lacks the second and third phase; most of the designers either focused on graphical designs. Intelligent response systems and intelligent visualization systems are crucial.

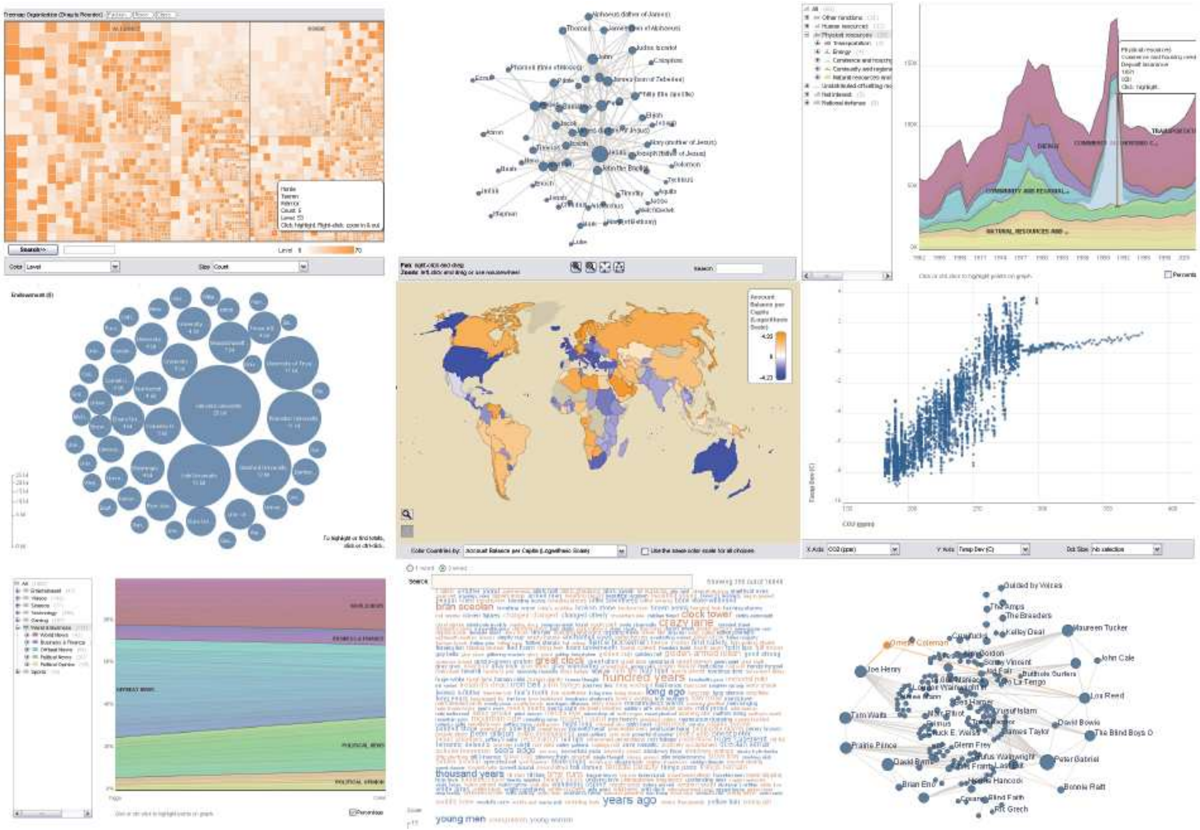
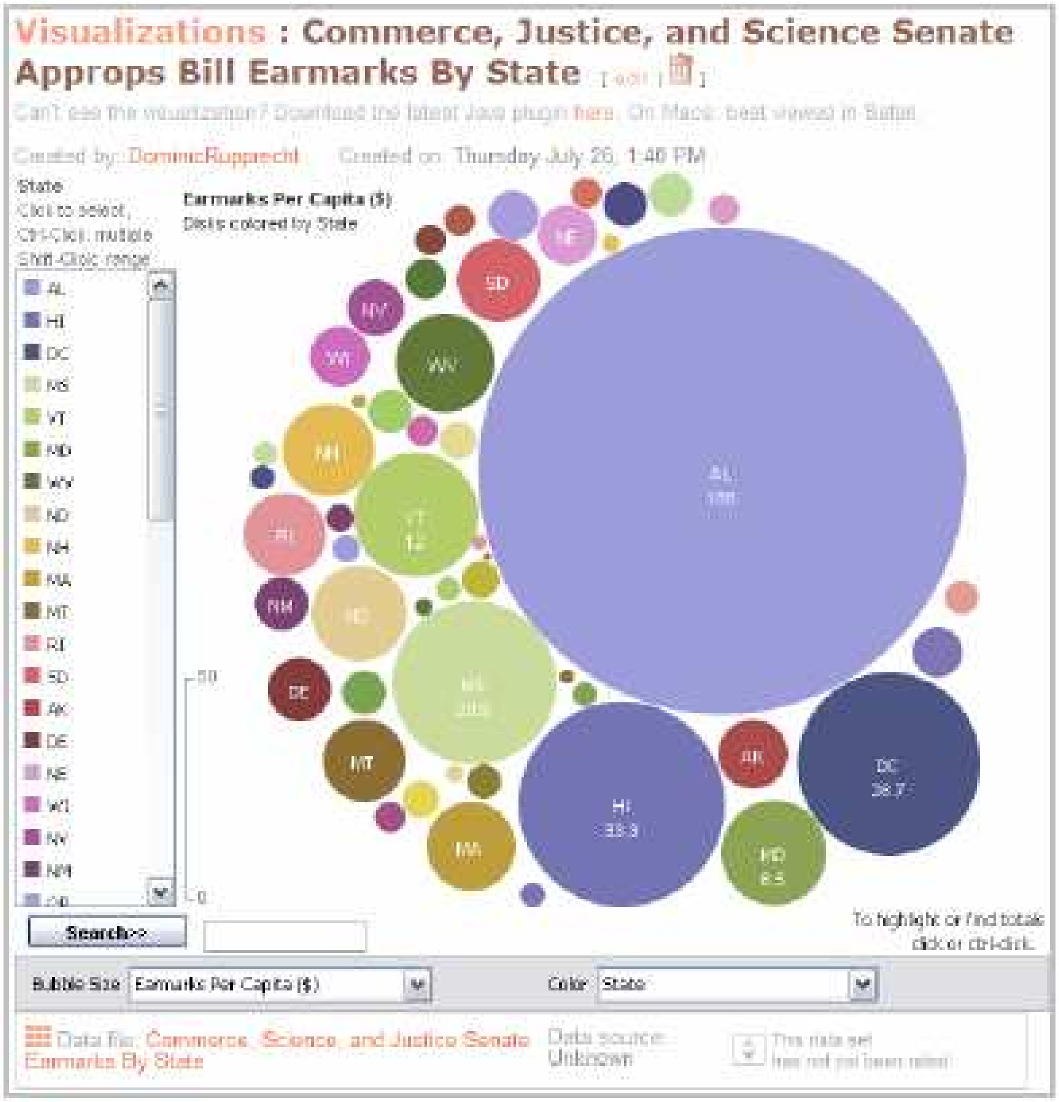


Figure 15: Visualizations user created from the Many-Eyes website. [48].

Another work by the research group from IBM at the TJ Watson Research Center [49] addressed the automatic designs of visualization system. Many-Eyes [50] is a website that provides collaborative visualization services, allowing users to upload data sets, visualize them, and comment on each other's visualizations. The goal of this site is to support collaboration around visualizations at a large scale by fostering a social style of data analysis in which visualizations not only serve as a discovery tool for individuals but also as a medium to spur discussion among users. To support this goal, the site includes novel mechanisms for end-user creation of visualizations and asynchronous collaboration around those visualizations.



Visualizations : Commerce, Justice, and Science Senate Approps Bill Earmarks By State [edit]

Can't see the visualization? Download the latest Java plugin here. On Macs: best viewed in Safari.

Created by: DominicRupprecht Created on: Thursday July 26, 1:46 PM

State
Click to select,
Ctrl-Click: multiple
Shift-Click: range

Earmarks Per Capita ($)
Disks colored by State

To highlight or find totals
click or ctrl-click.

Bubble Size: Earmarks Per Capita ($)

Color: State

Data file: Commerce, Science, and Justice Senate Earmarks By State

Data source: Unknown

This data set has not yet been rated

Figure 16: A visualization on Many-Eyes [50].

Figure 15 showed several visualizations users created by matching their own data to the website's design components [48]. This site is designed for all purpose visualization constructions.

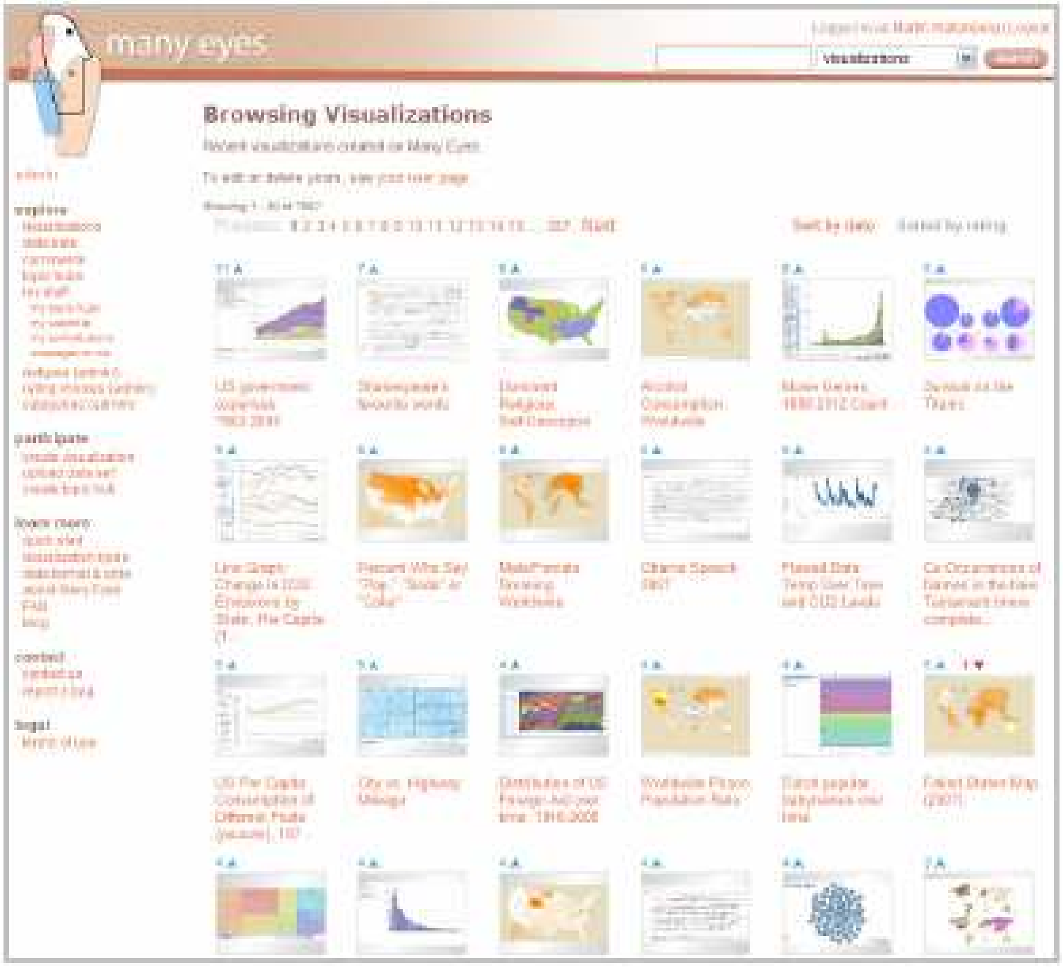


Figure 17: Users may browse visualization thumbnails [50].

Our work is similar to IBM Many-Eyes [50], but focuses on task-centered visualization designs. Instead of building one visualization interface only, our system allows users to solve problems by building visualization systems based on their specific tasks. The tasks are higher level problem solving tasks; the possible tasks may include: problem detection; problem identifying; problem diagnose and problem response. Traditional visualization systems focus on more on problem detection; it is not enough in order to solve a problem. Our system is a not only a visualization building tool; it's also a problem solving device.

BOZ [51] is an automated graphic design and presentation tool that designs graphics based on an analysis of the task for which a graphic is intended to support. A key feature of BOZ'S approach is that it is able to design different presentations of the same information customized to the requirements of different tasks, BOZ is used to design graphic presentation of airline schedule information to support five different airline reservation tasks. Reaction time studies done with real users for one task and graphic showed that the BOZ-designed graphic significantly reduces users' performance time to the task.

Polaris [52], a work by the research group at Stanford University, is an interface for exploring large multidimensional databases that extends the well-known Pivot Table interface. The novel features of Polaris include an interface for constructing visual specifications of table-based graphical displays and the ability to generate a precise set of relational queries from the visual specifications. Several issues were addressed, including analysis. The author mentioned several potential improvements, such as display of data hierarchical structure, and generate database table from a selected set of graphical marks.

The proposed complexity analysis is based on a number of psychological theories [1-5], including Guided Visual Search theory [5], Gestalt theory [4], Cognitive Load Theory [1]. According to the Cognitive Load Theory, there are three types of cognitive load: intrinsic cognitive load, extraneous cognitive load, and germane cognitive load. The mental effort to comprehend data visualization is part of the extraneous cognitive load, which is a major factor that influences the task performance. The proposed visualization complexity analysis is an attempt to measure the extraneous cognitive load of visualization comprehension.

3 An Analysis of the Existing Security Visualization Systems

We define the design space of network security visualization in three dimensions: data, visualization, and task. The *data* dimension consists of two layers: raw data and transformed data. The *visualization* dimension is divided into five layers: workspace, view, visual structure, and visual unit and visual variable. The *task* dimension consists of two layers: high level task and low level task, (figure 1).

In the next three sections, we will discuss each dimension of the design space in sequence. We will explain the relationship between different layers, identify key parameters, and list categories of possible design choices for each parameter.

Since design choices made in one dimension often affect the design choices in other dimensions, we will also discuss the relationship of a parameter or category with parameters or categories in other dimensions.

3.1 Visualization Classifications

3.1.1 Data

The data dimension is divided into raw data and transformed data layers. Raw data becomes transformed data through data transformation operations.

Raw Data: in network security visualization, raw data is the network traffic data. Although a wide variety of network data are presented in network security visualization systems, the majority of them focus on the raw network traffic data at the transportation layer and network layer. From my survey of the network security visualization literature, we have identified the most commonly visualized raw data as follows.

- Source and destination IP addresses
- Source and destination port numbers
- Time and date
- Protocols (e.g. TCP or UDP)

Some applications also make use of application layer data such as user information and application type.

Transformed Data: When raw data is processed to fit into the proper visualization media; this process is defined as data transformation. More specifically, most network security visualization programs obtain their data from various log files such as IDS logs, net flows, syslog, firewall logs, etc. The designated data for the specific visualization purposes maybe extracted or normalized from the raw network traffic data.

Presumably, there are many data transformation methodology; the most common ones are listed as follows:

- Classifying
- Counting and aggregation
- Filtering

- Sorting
- Clustering

Classifying. Network traffic can be classified in different ways. For example, it can be classified by the application types [53], such as WWW, mail, or multimedia, and some systems visualize such data. Intrusion detection systems (IDS) try to classify network traffic into normal and malicious categories [39].

Counting and aggregation can help to reduce the size of the data and therefore allow visualization systems to display the data at multiple levels of detail [54]. In addition, certain user groups may not want their network traffic details to be exposed, and therefore the aggregated data is the only thing that is allowed to be visualized.

Filtering can help reduce the data size and dimensions, eliminate noises and duplications, and help users focus on the important data. Most network log files are filtered by their application programs. However often in times, data filtering can be done interactively by users as well [55-57].

Sorting is a very common type of data transformation. For example, users may want to sort network data by time or by source IP. Sorting is particularly useful in table based visualizations.

Clustering is often used as part of a data mining process. Again, the clustering can be done automatically by programs or semi-automatically with user intervention.

Relationship with the visualization dimension: Both raw network traffic data (e.g. IP address, port number) and transformed data (e.g. IDS classifications) may be presented in the same visualization.

The selection of visualization techniques are largely influenced by the type and number of dimensions of the data. For example, some visualization techniques, such as parallel coordinates, are particularly useful for visualizing multi-dimensional data. The volume of data is also an important factor in making visualization design decisions. Another important design decision is how to map different data attributes to different visual structures, visual units, and visual variables.

Relationship with the task dimension: Data transformation is closely related to tasks. When data transformation is integrated with the visualization system, data transformation operations become tasks. That is, interactive data transformations become user tasks, while non-interactive data transformations become developer/program tasks.

Table 2 classifies data being used for some major existing security visualization systems. The raw data, data transformation, transformed data are all listed in table 1 below. Some of the published work did not discuss data transformation method in detail, we leave those blank.

The detailed classification of the existing systems is shown in table below.

Table 2: Data classification of the existing system.

major techniques	data		
	raw data	data transformation	transformed data
TNV{Goodall, 2005}	IDS data, user feedback etc		network packet time stamp
NVisionIP{Bearavolu, 2005}	netflow logs-->IP	calculating statistics	Statistics Derived from NetFlows
canine{Li, Luo, 2005}	Cisco Netflows, argus netflows		netflow logs
ATN{Yao, Shin 2005}	credentials, policies, strategies	enter into negotiation engine, logs are produced	log
VisFlowClusterIP{Yin, 2005}			log
VisFlowConnectIP{Yin, 2005}	cisco/argus		netflow logs
IDGraphs{Ren, 2005}	Cisco routers caching recent fows		netflow logs
visualizingNetwork {Abdullah, Conti, 2005}	network traffic		packet captured data from honeynet
exploringThreeDimensional {Oline, Reiner, 2005}	NID data/raw network activities		timestamp, priority, (source, destination)IP, desitination ports
fusionSummarization of behavior {Erbacher, 2005}	IP		relative value for placement of the histogram values
IDSRainstorm{Abdullah, Lee, 2005}	log		time, IP,
InteractiveDynamicVisualPortMonitoring { Erbacher 2005}	port		source IP, destination IP, source port, destination prot, connection type, connection time stamp, packet length
IPMatrix {koike, 2005}	log		sorted unified snort alerts
methodNetwork	network stream	data filtering	scan data
RootPolar{Fink, 2005}			IP
userCenteredLook(Idtk){Komlodi,2005}	raw TCP packet data or IDS tool generated alerts	Coarse filtering, interactive filtering	filtered data
VisCapbilityofIDS{Erbacher, 2005}	libcap, high dimensional IDS data	simcap enviornment	segmented large files, added additional data,
VisualCorrelationOfHost (portal) {Fink, 2005}	network traffic	Windows -> polling linux->loadable kernel moduel,	log
VisualExploreMalicious {Conti, 2005}	network		packet data
VisualParadigm {Livnat, 2005}			log
portvis {McPherson, Ma, 2004}			protocol, port, hour, session count, unique source addresses, unique destination addresses, unique src/dest address pairs, unique source countries
Visual Analytics(from visData {Teoh, Kwanliu, cga04})	network traffic		OASCs
BGPEye	Raw BGP routing data		converage time out values/event time out values

VAST_Oberheide2006	Raw BGP routing data	relevant statistics are extracted and stored in various tree-based data structures.	processed data
ESVT_Li 2006	two categories of experimental data in a testbed experiment: node status log and link traffic dump		human readable figures or animations
Knave-II, Shabtai 2006	relevant raw data and knowledge from the appropriate sources, indicated by the user,(e.g., CPU usage, TCP connection Failures) (time-oriented raw data)	compute statistics	statistics for raw data types include descriptive statistics such as mean, maximum, minimum, standard deviations, etc
DNS_Ren_2006	DNS query logs	derive	frequency of the data
3DGrameEngine_Harrop	ACL on a Cisco system router		log
Visual Motifs_Wright_2006	IP		unigram packet frequency
Sybil Attacks_Wang_2006			information of network topology

3.1.2 Tasks

In the field of information visualization, tasks are often defined implicitly as the interactions between users and the visualization system. Here we define tasks in a broader sense.

We divide the task dimension into two layers: high level task and low level task. For network security visualization, we define high level tasks as the tasks that deal directly with problem solving, and define low level tasks as the tasks that indirectly support problem solving.

3.1.2.1 High Level Tasks

Based on my definition, high level tasks can be further divided into several categories: *problem detection, problem identification*, *diagnosis, problem projection,* and *problem response*.

With only a few exceptions [58, 59], the current research on network security visualization has been focusing on low level tasks. The only high level task that has been effectively addressed so far is problem detection – that is, to detect malicious or anomalous behavior patterns through visualization. Current network security visualization systems support two types of problem detection: *signature based* and *anomaly based detections*.

In signature based detection, users know the visual patterns (signature) of the malicious behavior and try to look for the patterns in the visualization. Such visual patterns are specific to visualization design (particularly visual structure, visual units and visual variables). As a result the visual signature of a malicious behavior (e.g. denial of service attack) is usually different from system to system. In anomaly based detection, users establish a visual profile of the normal behavior and use it to find anomalous visual patterns. As discussed in section 4.4, much of the visual problem detection is associated with the concept of *Gestalt.*

In summary, we still do not have a good understanding of the high level tasks of network security professionals and how visualization techniques can assist in their work. Much research needs to be done in this area.

3.1.2.2 Low Level Tasks

Low level tasks are mainly about information gathering and presentation. A key parameter for low level tasks is who initiates the task. The *initiators* can be users, developers, or program.

It is important to differentiate the task initiators because it helps guide the design. The visualization of network security data is often the result of a combination of design choices made by developers, users, and programs. In majority of the network security visualization systems, developers make most of the design choices. But in some information visualization systems [60-62], users construct the visualization at run time. In systems that automatically generate visualizations, design choices are made by programs.

Therefore, we can further classify low level tasks based on the dimensions and layers they are associated with. Table 2 contains such a classification. Note that the tasks listed in the table are example tasks and the list is not complete. More tasks can be added.

Table 3: Low level task classification

	Low level tasks
Raw data	Add, delete, or change data source
Transformed data	filter, aggregate, classify, sort, cluster
Workspace	Add, delete, arrange, or coordinate multiple views
View	Zoom, pan, overview, focus+context
Visual structure	Add, delete, or modify relations, define visual structure
Visual unit	Identify, locate, distinguish, categorize, cluster, rank, compare, associate, correlate, retrieve, find anomalies
Visual variable	Change visual mapping

Table 4: Task classification of the existing systems.

major techniques	task
	User/tasks
TNV{Goodall, 2005}	multiple levels of details, problem detection
NVisionIP{Bearavolu, 2005}	multiple levels of details, problem detection
canine{Li, Luo, 2005}	converts and anonymizes NetFlow logs, user interactive technique
ATN{Yao, Shin 2005}	problem analysis
VisFlowClusterIP{Yin, 2005}	high level(arranging relevant IP, anomaly based)
VisFlowConnectIP{Yin, 2005}	problem detection, investigate anomalous internal and external network traffic
IDGraphs{Ren, 2005}	problem diagnosis, anomaly based, discover corelated attacks
visualizingNetwork {Abdullah, Conti, 2005}	scalling technique to reduce occlusion of data. Problem detection
exploringThreeDimensional {Oline, Reiner, 2005}	findinig false alarms, problem detection(finding malcious activities)
fusionSummarization of behavior {Erbacher, 2005}	problem detection, problemanalysis
IDSRainstorm{Abdullah, Lee, 2005}	problem detection
InteractiveDynamicVisualPortMonitoring { Erbacher 2005}	problem detection, problem analysis
IPMatrix {koike, 2005}	user visualization technique
methodNetwork	problem detection, problem analysis
RootPolar{Fink, 2005}	problem detection
userCenteredLook(Idtk){Komlodi,2005}	problem detection, problem diagnoise, problem analysis, user interactive technique
VisCapbilityofIDS{Erbacher, 2005}	user interactive technique, problem detection
VisualCorrelationOfHost (portal) {Fink, 2005}	problem detection, user interactive technique
VisualExploreMalicious {Conti, 2005}	problem detection, user interactive technique, problem analysis
VisualParadigm {Livnat, 2005}	problem detection, user interactive technique
portvis {McPherson, Ma, 2004}	problem detection, detail analysis
Visual Analytics(from visData {Teoh, Kwanliu, cga04})	problem detection, problem analysis
BGPEye	problem detection
VAST_Oberheide2006	user interactive technique, problem detection
ESVT_Li 2006	a modular, component-based topology editor, a TCL script generator, a worm experiment designer, and a visualization tool
Knave-II, Shabtai 2006	user interactive technique, problem detection
DNS_Ren_2006	problem detection/problem understanding/problem response
3DGrameEngine_Harrop	user interactive tool. problem understanding/ problem response
Visual Motifs_Wright_2006	problem detection/ problem classification
Sybil Attacks_Wang_2006	problem detection

3.1.3 Visualization

We divide the visualization dimension into five layers*: workspace, view, visual structure, visual unit, and visual variables*. Each layer is built on top of the preceding layer. That is, each workspace consists of one or more views. Each view contains one or more visual structures. Each visual structure contains multiple visual units, which are defined by visual variables. Therefore each layer captures a different level of detail in the design space.

The main benefit of this taxonomy is that it gives order and structure to visualization design process. Developers can use either a top down or bottom up approach to design their systems. For example, using a top down approach, a developer would first decide whether to use multiple views in the workspace, and then for each view, decide what visual structure should be adopted. The developer can then select the appropriate visual units and map data attributes to different visual variables.

Relationship with the data dimension: all the design decisions in every layer need to be made with regards to the underlying data. The structure of the visualization should preserve the structure of the data. The relationship among visual units should reflect the relationship among data items. The most important data attributes should be mapped to visual variables that can be quickly perceived and easily interpreted.

Next, we will discuss each layer in detail.

3.1.3.1 Workspace

Visualization workspace is the top layer of the visualization dimension and contains two categories based on the number of views: single view and multiple views.

Multiple views have many benefits. It provides flexibility [61], encourages users to look at data from different perspectives [63], and also helps visualize multi-dimensional data. The majority of the network security visualization systems have multiple views.

Multiple view workspaces can be further classified based on two parameters: view coordination and data source.

Multiple views may be *coordinated* or *not coordinated*. "Coordinated" means that the views are dynamically linked or synchronized – if one view is changed, the other views will be automatically updated. Based on Roberts [64], the key concepts of view coordination include the scope of the correlation, initiator, and what is correlated.

Multiple views may share the *same data source* or use *different data sources*. In the former case, the same data is visualized in different ways. For example, one view may display the raw network traffic data, while the other view displays aggregated data generated from the same raw network traffic. Multiple views that share the same data source are usually coordinated.

Relationship with the task dimension: the arrangement of multiple views is a low level task. Such arrangement can be determined by developers (a developer task), by the program (a program task), or by users (a user task). A particularly interesting design is to allow users to snap-together multiple view and dynamically link them [61, 62].

3.1.3.2 View

A view can be a window or a frame within a window. It contains one or more visual structures. The key parameters for the view layers are *contents* and *viewpoint*.

Contents can be *dynamic* or *static*. Dynamic contents means that the data visualized in the view may be changed during runtime. The update of the view content is a low level task that can be performed automatically by program (e.g. network data or IDS data streaming) or manually by users (e.g. load a different IDS log file).

Viewpoint can also be dynamic or static. Dynamic viewpoint means that the viewpoint can be manipulated, mostly by users, while static viewpoint means that viewpoint is fixed. Dynamic viewpoint is particularly useful when the entire data set is too big to be visualized in a view, which is a typical problem for network security visualization due to the enormous size of network data.

Relationship with the task dimension: viewpoint manipulations are low level tasks. Typical examples include zoom, pan, and focus+context [15].

3.1.3.3 Visual Structure

A visual structure is made up of one or more visual units. This layer determines how visual units are organized. A large number of information visualization techniques belong to this layer. Examples include bar chart, scatter plot, color map, parallel coordinates, tree and graph, etc.

We identify three key parameters for the visual structure layer: *coordinate systems*, *relationship among visual units*, and *space filling*.

In network security visualization, the most commonly used coordinate systems are *Cartesian* coordinate systems (both 2D and 3D), *Polar* coordinate systems [65, 66], and *Parallel* coordinate system [67]. The majority of the network security visualization systems use 2D coordinates, while a few systems use 3D coordinates.

There are three types of relationship among visual units: *no connection, hierarchical connection, and non-hierarchical connection*. Hierarchical connections are used to represent tree data structures (such as attack tree), while non-hierarchical connections are used to represent general graph data structures (such as computer networks) [68].

The visual units may be *space filling* or *non-space filling*. Space filling means that the entire visual structure is occupied by visual units. Non-space filling means that there may be gaps between visual units. Space filling techniques generally make better use of the display space, but they may suffer from information overloading. Note that table based visualizations are considered as space filling techniques when we treat each table cell as a visual unit.

A visual structure may encircle other visual structures [38]. For example, a table may contain scatter plots or bar charts in its cells. A bar chart may be placed on top of a geographical map.

Relationship with the data dimension: the selection of visual structures is largely determined by the nature, the number of dimensions, and the size of the data set. For example, if there are relationships among data elements, then connections need to be established among visual units. Node-link diagram is often used to visualize network connections [69]. Multi-dimensional data set may be visualized using parallel coordinates or visual pivot table [70]. The space filling dense pixel map is often selected to visualize IP address space because it can visualize a large amount of information in a small space.

A typical design problem in the visual structure layer is how to map network data to the axes of the coordinate system. A common practice is to convert IP address into two numbers, each of them mapped to one axis of the Cartesian coordinate. Port numbers are often sequentially mapped to pixels in a dense pixel map, either by column or by row. Temporal data is typically mapped to a horizontal or vertical axis [39].

Relationship with the task dimension: the selection of visual structure is a low level task. Often visual structures are selected by developers and are not changed during run time. However, in automatic visualization generation systems, the visual structure can be selected by the program based on pre-defined rules. Or the visual structure can be defined by users at run time.

3.1.3.4 Visual Units and Visual Variables

Visual units are the building blocks of visualization. Some examples such as: *point, line, 2D shape (glyph), 3D object, text*, and *image*. Each visual unit is defined by, but may not limit to, seven visual variables [21]: *position, size, shape, value, color, orientation,* and *texture.* For each visual unit, a visual variable is either assignable or fixed. An empty cell means the visual variable is not applicable to the visual unit.

An assignable variable visual means that data attributes can be mapped to this visual variable, otherwise, if it is fixed, data attributes cannot be mapped to it. For example, for dense pixel maps, IDS classification can be mapped to pixel colors, but not to its size [39].

Table1 presents the relationship between some common visual units and visual variables.

Table 5: Visual units and visual variables.

	position	**Size**	**shape**	**value**	**color**	**orientation**	**text**
position	assignable	fixed	fixed	assignable	assignable	fixed	
line	assignable	assignable	fixed	assignable	assignable	assignable	
2D	assignable	assignable	assignable	assignable	assignable	assignable	assignable
3D	assignable	assignable	assignable	assignable	assignable	assignable	assignable
text	assignable	assignable	fixed	assignable	assignable	assignable	
image	assignable	assignable	fixed	fixed	fixed	assignable	

Sometimes the selection of a particular visual structure would limit the choices of visual units. For example, if parallel coordinate is selected as visual structure, then lines should be used as visual units. Similarly, the selection of a particular visual unit may also limit the choices of visual structure.

Perhaps we can add a special visual unit – *Gestalt*, as in Gestalt psychological theory. Here, a *Gestalt* is defined as a group of visual units that can be easily perceived by humans as a “pattern” due to the Gestalt theory – that is, the laws of proximity, similarity, symmetry, continuity, etc.

Gestalt is particularly important for network security visualization because one of the primary purposes of such visualization systems is to help users detect malicious or anomalous network traffic patterns. Such patterns are often visualized as a *Gestalt* of pixels, lines, or glyphs. For example, in current network security visualization systems, port scanning is often visualized as a cluster of lines or a group of closely packed pixels with the same color. If a malicious or anomalous pattern is not visualized as a *Gestalt,* then it is usually difficult to be detected by human.

Thus a fundamental challenge for network security visualization designers is “how to design the visualization so that the malicious or anomalous behavior can be perceived by users as *Gestalt*?” And the network security visualization systems should be evaluated by whether they can effectively convert malicious or anomalous behavior patterns to *Gestalts*.

Unfortunately, most of the current research in network security visualization still focuses on the low level details of how to map network data to visual units and variables, and the high level task of mapping malicious patterns to Gestalt has not received much attention.

Relationship with the data dimension: a basic design question a developer would face is how to map data items to visual units and how to map data attributes to visual variables. Again, the selection of visual units and visual variables is largely determined by the nature, the number of dimensions, and the size of the data set. For example, Chernoff faces [71], a 2D shape, is often selected to visualize multi-dimensional data. For large volume of data set, pixel is often selected as the visual unit because it allows more data to be visualized in a small display space.

The selection of visual variables is also influenced by the characteristics of visual variables. Bertin [21] has identified five characteristics: selective, associative, quantitative, order, and length. For example, color is selective but not quantitative, meaning it is appropriate to map categorical data (such as IDS classification) to color [39], but it is usually not appropriate to map quantitative data to color.

Relationship with the data dimension: the mapping of data to visual units and visual variables is a low level task. This visual mapping can be pre-defined by developers, automatically performed by programs based on certain rules [72, 73], or manipulated by users at run time.

Majority of the works we have found uses multiple views. Multiple views allow the user to browse several windows and several concepts at the same time; it's both cognitive efficient and view efficient.

Table 6 provided a visualization classification of the existing systems. Multi-view appeared to be a more popular style of window design; simply because it is more user-friendly. Some other systems chose single view, depending on their design goals.

Table 6: Visualization classification of the existing systems.

major techniques	visualization				
	window	**view/frame**	**visual structure**	**visual unit**	**visual variables**
TNV{Goodall, 2005}	multiple view	dynamic/multi source of data	color/catisian/connection	box/2D shape	color
NVisionIP{Bearavolu, 2005}	multiple view	dynamic/single source of data--Netflow IP	color/catisian/no connection	pixel	color
canine{Li, Luo, 2005}	single view	static/multiple source of data	conversion tool/no connection	text	N/A
ATN{Yao, Shin 2005}	multiple view	dynamic/single source of data	hierachical connection graph	2-D object or node	N/A
VisFlowClusterIP{Yin, 2005}	single view	dynamic/single source of data--Netflow log	connected nodes	2D shape	color
VisFlowConnectIP{Yin, 2005}	multiple view	dynamic/multi source of data	connections between hosts	3 axis of hosts, connections between hosts	axis, connection lines
IDGraphs{Ren, 2005}	multiple view	dynamic/single source of data	pixel/cartisian/no connection	pixel	pixel density/color
visualizingNetwork {Abdullah, Conti, 2005}	single view	dynamic/single source of data	box/cartisian/no connectin	2-D shape--box	color
exploringThreeDimensional {Oline, Reiner, 2005}	single view	dynamic/single source of data	3-D/Icon/no connection	3-D object	color
fusionSummarization of behavior {Erbacher, 2005}	single view	dynamic/single source of data	activity lines/connected histogram	2-D shape: activity lines	shape change of the activity line
IDSRainstorm{Abdullah, Lee, 2005}	single view	dynamic/single source of data	pixel/cartisian/no connection	pixel	color
InteractiveDynamicVisualPortMonitoring { Erbacher 2005}	single view	dynamic/single source of data	horizontal port lines/connected	2-D shape, connection lines between host	color
IPMatrix {koike, 2005}	multiple view	dynamic/single source of data	pixel/catisian(matrix view)/no connection	pixel	color(grid/pixel)
RootPolar{Fink, 2005}	single view	dynamic/single source of data	pixel/Polar coordinate	pixel	color
userCenteredLook(Idtk){Komlodi,2005}	multiple view	dynamic/multiple source of data	3Dobject/3D cartisian/no connection	3D object	shape, color
VisCapbilityofIDS{Erbacher, 2005}	multiple view	dynamic/single source of data	2-D rings/no connection	2D shape: 2D ring	shape
VisualCorrelationOfHost (portal) {Fink, 2005}	multiple view	dynamic/single souce of data	text/connected between ports	text	N/A

VisualExploreMalicious {Conti, 2005}	multiple view	dynamic/semantic levels	pixel/cartisian/no connection	pixel	shape change of the pixels arrangement, color
VisualParadigm {Livnat, 2005}	single view	dynamic/single souce of data	catisian, centered circle	nodes, hierarchical connections	length of the arc
portvis {McPherson, Ma, 2004}	multiple view	dynamic/single source of data	pixel/no connection	pixel	color
Visual Analytics(from visData {Teoh, Kwanliu, cga04})	multiple view	dynamic/single source of data	connection lines between IP and Ases	lines	color
BGPEye	Raw BGP routing data		converage time out values/event time out values	single/multiple view	dynamic/single source of data
VAST_Oberheide2006	Raw BGP routing data	relevant statistics are extracted and stored in various tree-based data structures.	processed data	multiple view	dynamics/single source of data
ESVT_Li 2006	two categories of experimental data in a testbed experiment: node status log and link traffic dump		human readable figures or animations	multiple view	dynamics/single source of data
Knave-II, Shabtai 2006	relevant raw data and knowledge from the appropriate sources, indicated by the user,(e.g., CPU usage, TCP connection Failures) (time-oriented raw data)	compute statistics	statistics for raw data types include descriptive statistics such as mean, maximum, minimum, standard deviations, etc	multiple view	dynamic/multiple source of data
DNS_Ren_2006	DNS query logs	derive	frequency of the data	multiple view	dynamic/single source of data
3DGrameEngine_Harrop	ACL on a Cisco system router		log	single view	dynamic/single source of data
Visual Motifs_Wright_2006	IP		unigram packet frequency	single view	dynamic/single source of data
Sybil Attacks_Wang_2006			information of network topology	single view	static/single source of data

3.2 Other Taxonomy

Various taxonomies have been proposed in the field of information visualization [15-17, 74-82]. Both Shneiderman [15] and Wehrend [76] proposed a two dimension taxonomy based on data types and low level tasks. However, they do not provide a classification for the visualization techniques. Tory and Moller [82] extended Shneiderman's work by dividing data into two categories, continuous and discrete, and introduced structure as a parameter. They also describe a data centered, two dimensional task taxonomy based on continuous/discrete data and data spatialization. Card and Mackinlay [15] analyzed the information visualization design space based on data type, data transformation, mark types, retinal properties (similar to visual variables), position in space time, view transformation, and widget – all of them are included in our taxonomy. Card, et al. [15] and Chi [16] described a pipeline based framework to classify visualization techniques. In general, our taxonomy is closer to the one proposed by Keim [17], who describe a taxonomy in three dimensions: data, visualization techniques, interaction and distortion techniques. Keim provides a categorical classification for each dimension, but does not further divide them into layers or identify key parameters. Our taxonomy is more comprehensive and detailed than other taxonomies.

Our analysis of the data dimension is influenced by the pipeline model proposed by Card, et al. [15] and Chi [16], who both classify data into raw data and transformed data. Our analysis of the data transformation operations is influenced by the work of Tang, et al. [83]

Our analysis of the visualization dimension is based a comprehensive survey of the literatures on computer security visualization, many of which are published in the proceedings of the first two

International Workshop on Visualization for Computer Security (VizSEC). The five layer hierarchical framework is partially inspired by Zhou and Feiner [73]. However, Zhou and Feiner discuss their hierarchical layers in the context of automatic visualization generation. Unlike our work, they do not identify key parameters and categories for each layer. The analysis of the visual unit and visual variable layers are influenced by the work of Card and Mackinlay [15] and Bertin [21].

There has not been a comprehensive analysis of the tasks in network security visualization. Our analysis of the low level tasks is based on our experience and many previous works on general visualization tasks [15, 16, 72, 78, 82-87]. Our high level task analysis of the network security visualization also benefits from some previous works [58, 59, 88].

4 A Method for Measuring Visualization Complexity

4.1 A Theoretical Framework for Defining and Measuring Effectiveness

Visualization should be: accurate, user friendly, and efficient.

The accuracy can be measured in three steps. First, develop a classification of visual attributes and structures. Second, conduct a domain specific data analysis and classify data attributes and structures. Identify the possible mappings between visual attributes and data attributes as well mappings between visual structures and data structures. Finally, an accuracy score can be assigned to each mapping.

An effective visualization should help the users to achieve the goals of specific tasks. The utility principle defines the relationship between visualizations and tasks. A data-visualization should be designed for specific tasks so that the utility and efficiency of the visualization can be evaluated. To measure the utility of visualizations, a task classification should be developed through domain task analysis. Second, a shared and annotated benchmark database should be created, with benchmark tasks and measurable goals clearly defined. Third, assess the utility of the visualizations by measuring how well they help achieve the goals of the benchmark tasks in comparison with non-visual representations. User studies can play a major role in this part; traditional user studies would normally assemble a group of undergraduate students, the result can be biased. An improved version of the user studies should be carried out based on the domain experts' opinions, using the same visualization performing the same task with the same database.

Table 7: Quantitative and qualitative measurements of effectiveness.

	Quantitative measurements	*Qualitative measurements*
Accuracy	• Measure the number of interpretation errors	• Observation • Interview • Expert/novice comparison
Utility	• Measure the number of achieved benchmark goals • Record the number of times a visualization design is selected by users to conduct a task	
Efficiency	• Complexity analysis • Record task completion time • Record eye movements • Measure the learning curve	

The three different criteria are closely related; their relationship can be best described by the following diagram:

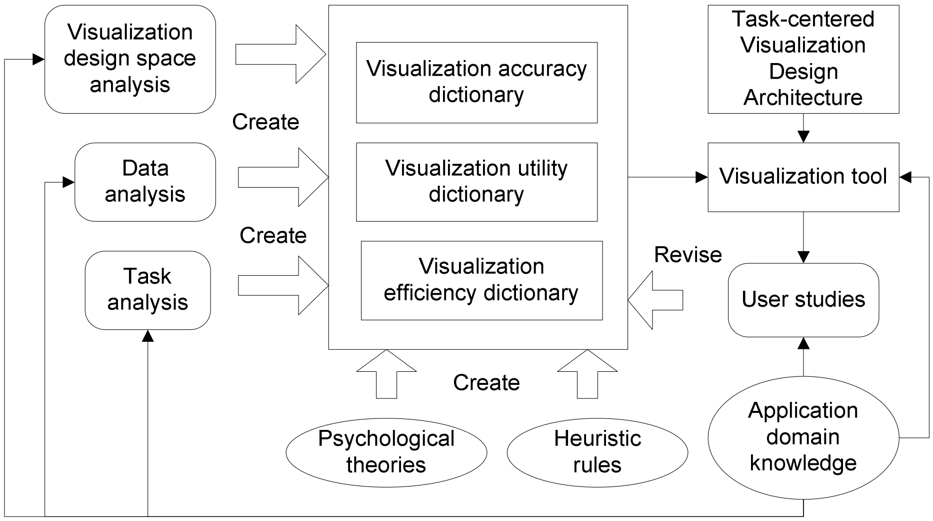

Visualization design space analysis
Data analysis
Task analysis
Create
Create
Visualization accuracy dictionary
Visualization utility dictionary
Visualization efficiency dictionary
Create
Psychological theories
Heuristic rules
Task-centered Visualization Design Architecture
Visualization tool
Revise
User studies
Application domain knowledge

Figure 18: Overview of the proposed design methodology.

4.2 Visualization Design Methodology

The processing of a data visualization depends on a host of psychological processes [89], including information read-off, integration, and inference [3].

The processing of a data visualization depends on a host of psychological processes [89], including information read-off, integration, and inference [3]. The main goal of the proposed complexity analysis is systematically evaluate the major factors that influence the efficiency of information read-off and integration. Visual inference is currently beyond the scope of this study due to a lack of understanding of its psychological process. However, as Trafton, et al. [3] point out, visual inference is likely to depend on visual integration and information read-off.

Graphically, the complexity measuring is carried out in the following steps:

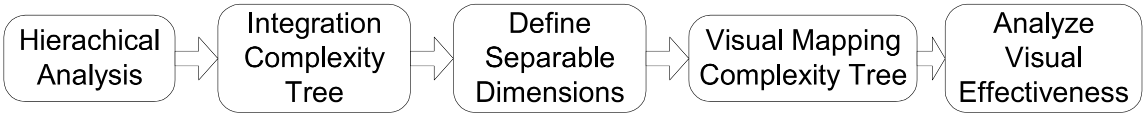

Hierachical Analysis
Integration Complexity Tree
Define Separable Dimensions
Visual Mapping Complexity Tree
Analyze Visual Effectiveness

Figure 19: General steps complexity analysis.

In the following sections, we give more details about the major factors of the complexity measurement; furthermore, we will incorporate the TNV visual interface as an example in each stage to demonstrate the idea.

4.3 Hierarchical Analysis of Data Visualization

We divide data visualization into five hierarchical layers: workspace, visual frame, visual patterns, visual units, and visual attributes. Each one is a component of the previous layer. A workspace is one or more visual frames that are designed for a specific purpose. A visual frame is a window within a workspace and contains multiple visual patterns. A visual pattern is a set of visual units that are readily perceived as a group; and they are identified based on four Gestalt laws [4]. Some examples of visual units such as: point, line, 2D shape (glyph), 3D object, text, and image. Each visual unit is defined by seven visual attributes [21]: position, size, shape, value, color, orientation, and texture.

4.4 Visual Integration

Larkin and Simon [90] point out that a main advantage of visualization is that it helps group together information that is used together, thus avoiding large amounts of search. In complex problem solving, the visual units need to be integrated [3], which adds to the extraneous cognitive load [1].

In this study, the cognitive load of visual integration is estimated by building a visual integration complexity tree (figure 3). For each visual frame, we identify the visual patterns in that frame based on four Gestalt laws: proximity, good continuation, similarity, and common fate [4, 89]. The number of nodes on the visual integration complexity represents the upper bound of visual integration a reader might perform.

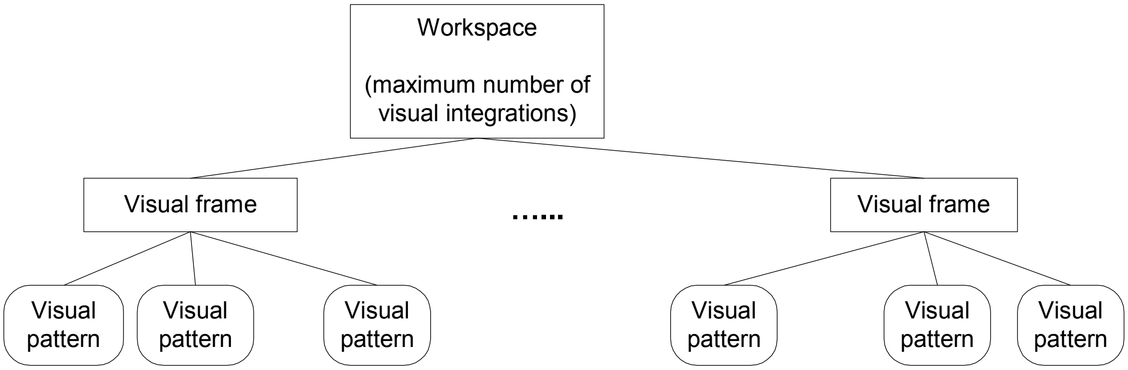


Figure 20: Visual integration complexity tree.

4.5 Separable Dimensions for Visual Units

Psychological studies showed that the eye can only see three variables at once, and the difficulty of using a graph is determined by how many fixations are required [91]. Therefore the number of separable dimensions is another field. The following are some of the visual units' criterion of separating dimensions:

- X and Y coordinates (position)
- Shape, Size, Color,
- Value (gray scale),
- Orientation

Groupings are subjected to change depending on the visualization. For example, in some cases, shape and size may be considered as integral dimensions, or X and Y axes may be considered separable dimensions. Generally, commonly associated dimensions should be considered as integral dimensions (e.g. profit/time, etc.) Otherwise, they are separable dimensions. Identifying separable dimensions can be different from evaluator to evaluator, but as long as it's consistent, the outcome of evaluation should not be affected.

4.6 Interpreting the Values of Visual Attributes

Readers also need to interpret the values of these visual attributes. The mental effort for such interpretation is another source of extraneous cognitive load [1]. In my analysis, for each separable dimension, we assign a score for the complexity of interpreting the values of the visual attribute based on the following criteria:

Table 8: Complexity scores for interpreting the meaning of visual units.

Complexity score	*Criteria*
5	Very difficult to interpret. There is no legend. A typical reader has to memorize the mapping between the value of the visual attribute and the value of the corresponding data parameter
4	More difficult to interpret. A typical reader needs to frequently refer to a legend to interpret the value of the visual attributes
3	Somewhat difficult to interpret. A typical reader needs to refer to a legend from time to time.
2	Relatively easy to interpret. A typical reader only needs to refer to a legend occasionally.
1	Easy to interpret. This is based on common knowledge. There is no need to memorize or refer to a legend.

For integral dimensions, each separable dimension has a complexity score. The complexity score for a type of visual unit is the sum of complexity scores of its separable dimensions. The complexity score for a visual frame is the sum of scores of different types of visual units it contains, and so on. The standard prototype of a visual mapping complexity tree is shown in figure 7.

Counting the total number of visual units is a quick way to estimate the amount of visual information a reader needs to process. In case of a very dense pixel map, an approximate count is usually sufficient for evaluation purposes.

4.7 Efficiency of Visual Search

According to Wolfe and Horowitz [5], target-distracter difference is the key to efficient visual search, and there are four major factors that affect the target-distracter differences -- color, motion, size, and orientation. Target-distracter difference indicates how a target stands out from the background; background can be any other surrounding objects or the neighboring background colors. Distracters as opposite to a targeted object can be identified based on different evaluators' intentions. The accurate calculations are performed through the following equations.

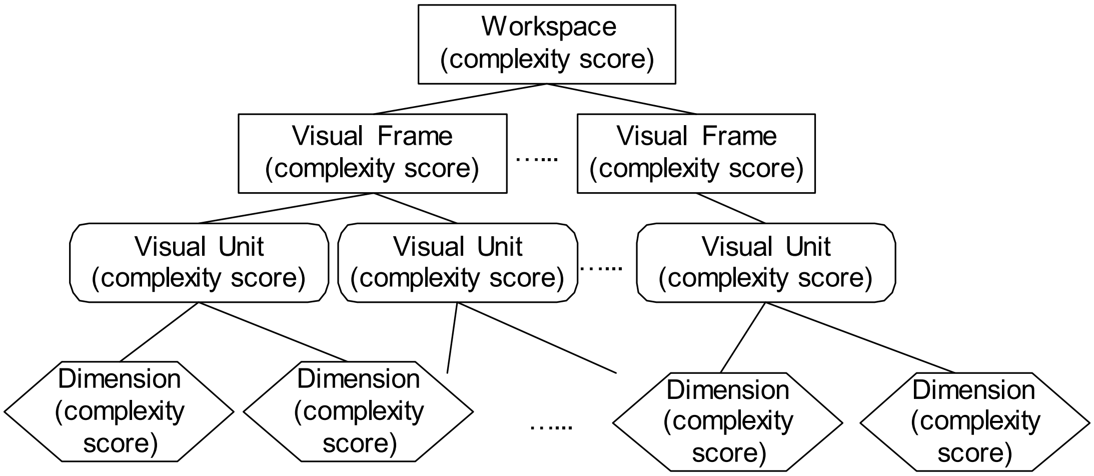

Workspace (complexity score)
Visual Frame (complexity score)
……
Visual Frame (complexity score)
Visual Unit (complexity score)
Visual Unit (complexity score)
……
Visual Unit (complexity score)
Dimension (complexity score)
Dimension (complexity score)
……
Dimension (complexity score)
Dimension (complexity score)

Figure 21: Visual mapping complexity tre**e.**

$$c = \left(\left| \frac{\sum_{i=1}^{n} T_r}{N} - D_r \right| + \left| \frac{\sum_{i=1}^{n} T_g}{N} - D_g \right| + \left| \frac{\sum_{i=1}^{n} T_b}{N} - D_b \right| \right) / 765 \qquad (1)$$

$$M = \frac{\left| \frac{\sum_{i=1}^{n} Tf}{N} - Df \right|}{F} \quad (2) \qquad s = \frac{\sum_{i=1}^{n} \frac{|Ts - Ds|}{s}}{N} \quad (3) \qquad o = \frac{\sum_{i=1}^{n} \frac{\sum_{i=1}^{n} |To - Do|}{N}}{180} \quad (4)$$

Equation 1: T_r , T_g , and T_b are the R/G/B values of target color, D_r , D_g , D_b are the R/G/B values of the closest distracter color. The N and n are the number of color components represented in the graphic scene. Number 765 is the distance between the two most distant colors.

Equation 2: The formula for calculating target-distracter motion ratio for graphics associated with entity node such that *Tf* and *Df* are the target and distracter frequency, F is the fastest moving frequency represented in the visualization scene, and N and n are the number of moving items.

Equation 3: Formula for calculating the target-distracter size ratio such that *Ts* and *Ds* are target and distracter graphics' size in ps, S is the visualization area in ps, and N and n are the number of entity nodes. The larger size ratio means it is easier for the user to find the pattern through the size hint provided by the visualization.

Equation 4: Formula for calculating target-distracter orientation ratio such that *To* and *Do* are the target and distracter graphics' orientation, and N and n are the numbers of distracters.

5 Task Centered Visualization Design Frame Work.

This section contains three parts: theoretical backgrounds, system implementation, and a case study using SNORT.

5.1 System Description

The following diagram describes the system functions:

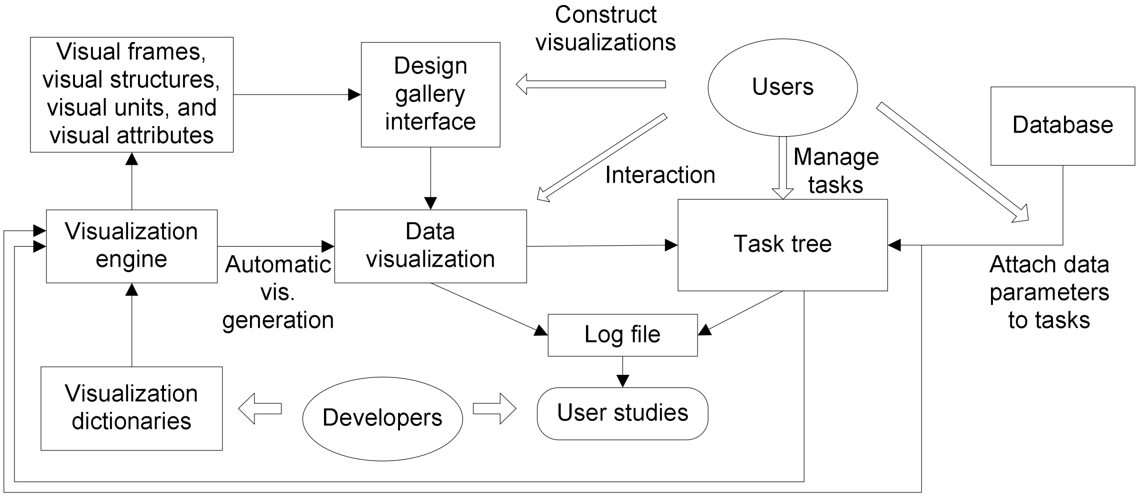


Figure 22: Overview of the proposed task-centered visualization architecture.

5.1.1 User Controlled Visualization Constructions

A central theme of TVDA is to enhance the role of domain experts in the collaborative construction of visualizations. In a typical visualization design process, domain experts are often limited to specifying requirements and testing the programs. With TVDA based visualization tools, domain experts are in control of constructing the visualizations. (Of course, developers can still construct visualization if necessary.) To domain experts, the visualization construction is no longer a black-box process – they can break visualization into different layers of visual mappings and use the visualization database to find the psychological theories and empirical studies behind these mappings. For example, domain experts will be particularly interested in finding the rationale behind the accuracy scores of these mappings.

User controlled visualization construction is necessary for several reasons. First, complex problem solving is a dynamic process. In search for a solution, users need to test different hypotheses or different strategies. This means the task structure may be constantly changing, and a good visualization tool should allow users to dynamically reorganize visualizations to accommodate this change – because the effectiveness of visualization is task specific. From PI's experience, such flexibility is very important to domain experts. Second, studies have shown that the effectiveness of visualizations depends on users' background and knowledge. Visualization is also a learned skill – as users become more experienced, their behavior for reading and constructing visualization may change [92]. Prefabricated visualizations combined with low level interactions – such as zooming, panning, and level-of-detail – are insufficient to address the individual differences. Third, self-constructed visualizations may assist problem solving in ways different from prefabricated visualizations [11, 93]. Over time, these benefits will outweigh the initial learning curve.

To construct visualization, users start with a visual frame and then drag and drop visual structures into the view. They are assisted by a design-gallery style interface [94, 95] that contains multiple visual structures provided by a visualization engine. Once visual structures are selected, users then map visual units and visual attributes to the selected data attributes of the selected task. Again, the visualization engine and design gallery interface will assist users to choose among multiple visual unit and visual variable combinations, which are ranked based on their accuracy, utility, and efficiency scores.

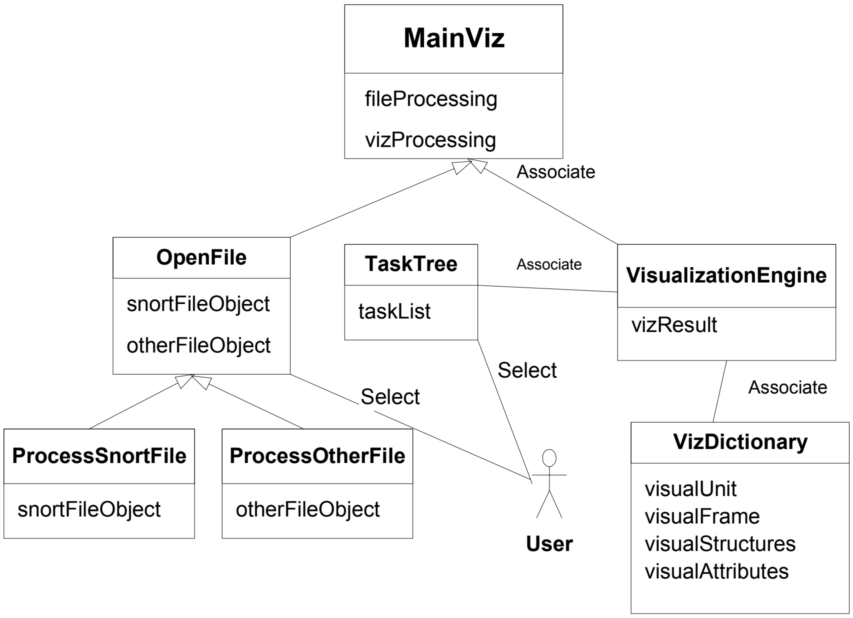


Figure 23: UML diagram of the proposed System.

A data-visualization may be constructed for multiple tasks, but the link between the visualization and the tasks must be explicitly identified.

5.1.2 Task Tree

Tree is an appropriate data structure for organizing and storing problem solving activities [96-98]. Each node on the task tree represents a specific task. For each task, users can add, delete, edit, merge or split tasks; they can create task hierarchy by dividing a task into sub-tasks. In the proposed system, we use a task tree to help user organize the tasks. For each task, users are required to explicitly identify the data parameters that are needed to perform the task. More specifically, these are the parameters that have to be kept in the working memory simultaneously in order to carry out the task. The proposed visualization tool allows users to open data files, select parameters, and attach them to a task.

The requirement for users to explicitly specify tasks and their related parameters is grounded in the proposed theoretical framework. It is based on the belief that the effectiveness of a data-visualization is task specific. Therefore visualization should be optimized for specific tasks by mapping the task related parameters to the visual elements that have high accurate, utility, and efficiency scores. Second, the process of specifying tasks and their associated parameters is essentially a task complexity analysis. Knowing the data parameters associated with a task helps users consciously control the complexity of the tasks and correlate task complexity and visualization complexity. For example, users would focus their attention on constructing visualizations for high complexity tasks, because visualizations are shown to be more effective for high complexity tasks than simple tasks [99-101].

A task tree also has other benefits. First, the task tree itself can be seen as a visualization of the problem solving process, reducing the cognitive load by externalizing the task structure that would otherwise be stored in the working memory. Second, a task tree is essentially a visual language for describing a specific problem solving strategy and expertise [102, 103], which can be shared or reused.

5.1.3 Interactions

In addition to the traditional interaction techniques (e.g. zooming, panning, level-of-detail [104]), the TVDA also includes a number of new interaction techniques:

- Merging and splitting visualizations. Users will be able to merge two or more visual frames. This technique is designed to reduce the cognitive load of visual integration and inference by externalizing the mental transformations [3, 105-107].

- Encode the legend in audio. For example, when users place the mouse cursor on a visual unit, the encoded data is spoken out through speech synthesis. This technique is inspired by the dual-coding theory [108, 109]. It reduces the cognitive load of memorizing the mapping between visualization attributes and data attributes, and also eliminates the need for moving the eyes between data visualization and the legend.

- User constructed annotation. Users are able to insert visual, textual, or audio annotations directly into the data visualization frames. This technique helps reduce the cognitive load by offloading part of the reasoning processing from working memory to external representations.

5.2 Implementation

In this section I will discuss a task centered visualization design framework, in which tasks are explicitly identified and organized and visualizations are constructed for specific tasks and their related data parameters. The center piece of this framework is a task tree which dynamically links the raw data with automatically generated visualization. The task tree serves as a high level interaction technique that allows users to conduct problem solving naturally at the task level, while still giving end users flexible control over the visualization construction.

I have built a system based on the proposed framework. This prototype is implemented with Java and uses the *prefuse* [88] – an open source interactive information visualization library.

5.2.1 Design Strategies and Criteria

There are two main visualizations in this case, a display that contains all visual frames and a task tree. The center control agent in this design is the task tree, task tree controls opening of all visual frames and manipulations of all currently opened visual frames, although manipulation can also be done through toolbars within the visual frames. Task tree nodes are the controlling agent, and leaf nodes are the manipulating agent. Task tree is not part of the display, but the display contains all other visual frames. Task tree can be modified directly by the user, but visual frames can only be modified through task tree nodes.

Table of data must be loaded before any other action takes place; the only type of data the system can accept is SNORT data.

5.2.2 Task Tree

A task tree servers as the center piece of this system; not only does it have control over all graphical visualizations, it also allows the user to have full control over the task tree. User may modify the tree by deleting, re-arranging, and adding new nodes; user may also add and delete the parameters associated with each individual task.

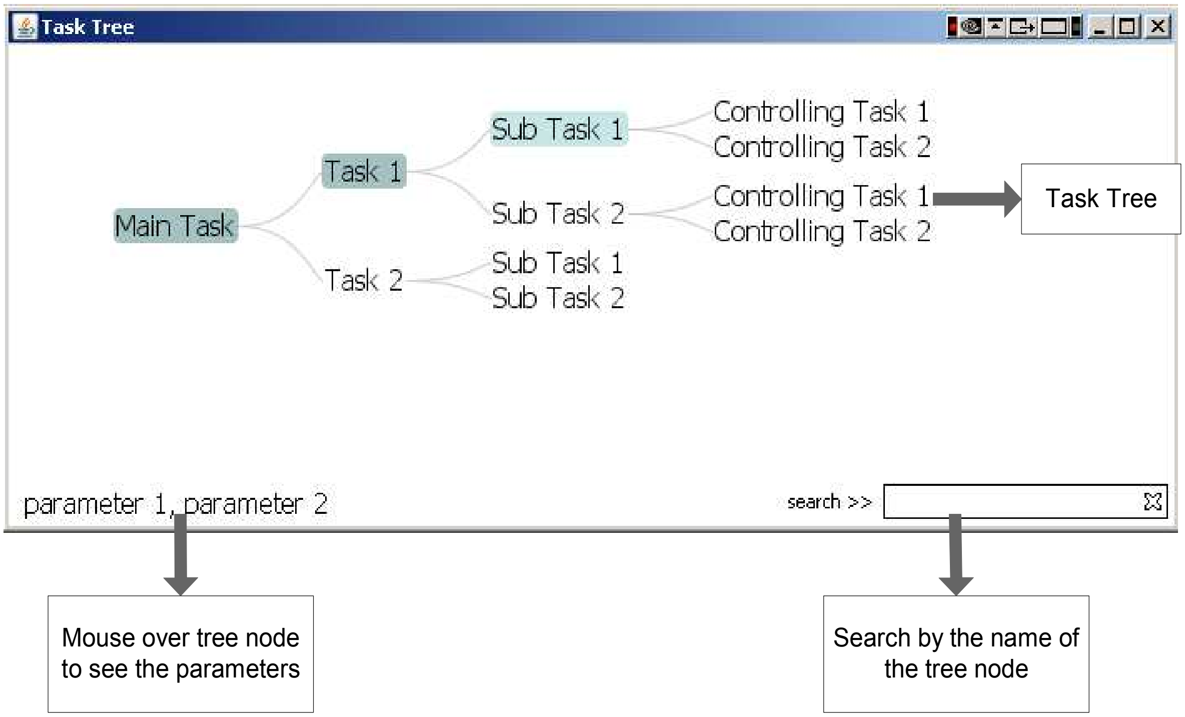
A typical task tree would contain a main task, which describes the main purpose of the task tree. The main task can be divided into sub-steps or individual tasks (task 1, task 2 etc.); the sub-steps are described or composited by sub-tasks; and sub-tasks can be even further divided if necessary. Finally, the leaf nodes of the task tree represent the controlling tasks or parameters for the each sub-task. The whole task tree can be viewed as a problem-solving process; each node represents a step to accomplish in order to solve its parent node's task. User may choose to divide the task into sub layers based on their need; there's no limit on the number of child nodes a task tree can contain.

Task tree is built to be visually efficient, only the clicked node will be expanded and the sub tree of the clicked node will be displayed. Without any user activities, the tree will only show limited number of nodes. This special feature is kept to reduce user's cognitive load by reducing their visual load significantly.

The figure below shows the general structure of the task tree, depending on the user's design goal, the task tree may vary accordingly. In this example provided, main task is the ultimate goal or the problem the user is trying to solve. Task 1 and task 2 are the tasks involved in this goal. Sub tasks are optional, they can be the general steps involved, or the individual frames that construct the main frame. Controlling tasks do not generate any new frames; they only control the opened visualization.

Each sub-task will generate an individual visualization which is pre-defined by the user. User may directly manipulate the visualization by click on each controlling task.

Task tree is built based on a pre-defined XML file. The XML file is publicly available and can be modified. User may also choose to modify from the display instead of reading through XML code and modify from there. The functions we provided are: adding task, deleting task, moving task and see parameters. Parameters defines a task, they are the essential component of a task. User may choose to modify the parameters by accessing the parameter table. The parameters can be deleted, modified, and new parameters may be added simply by typing in the parameters into the table.



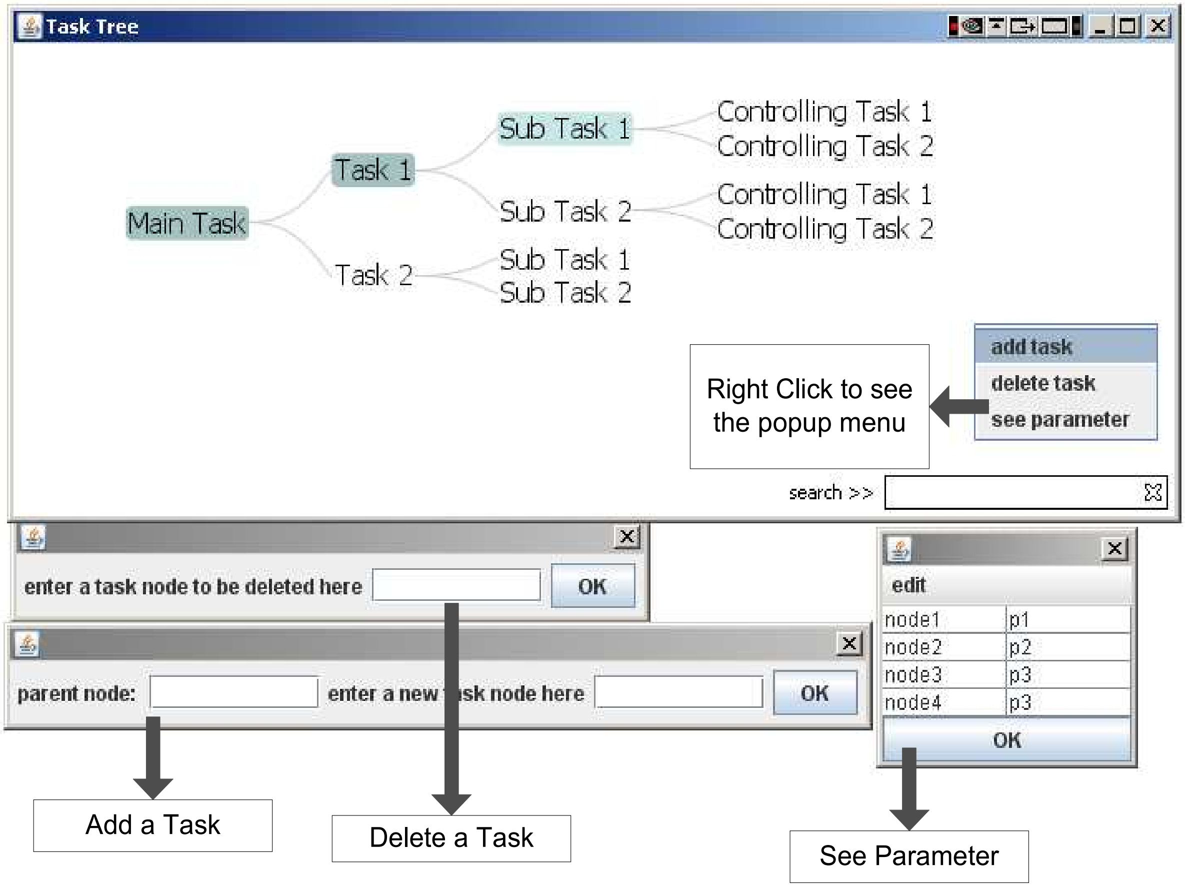
Task Tree

Mouse over tree node to see the parameters

Search by the name of the tree node

Figure 24: A general task tree.

The figure below shows the general task tree, when user right clicks, a popup menu will provide options of “add task”, “delete task” and “see parameters”. Parameters can only be modified through accessing of the parameter table. Modifying of the parameters can be done by directly modification of the table entries, or “edit” menu item provided on the menu bar. In case the task tree is too large to browse through, user may also choose to search the task by using the search bar located on the right bottom corner of the display.



node1	p1
node2	p2
node3	p3
node4	p3

Figure 25: Right click on task tree.

Task tree can be controlled through simple mouse motion, such as clicking, dragging, double clicking, or mouse over. A brief summary of the common functions are described by the following table (table 9).

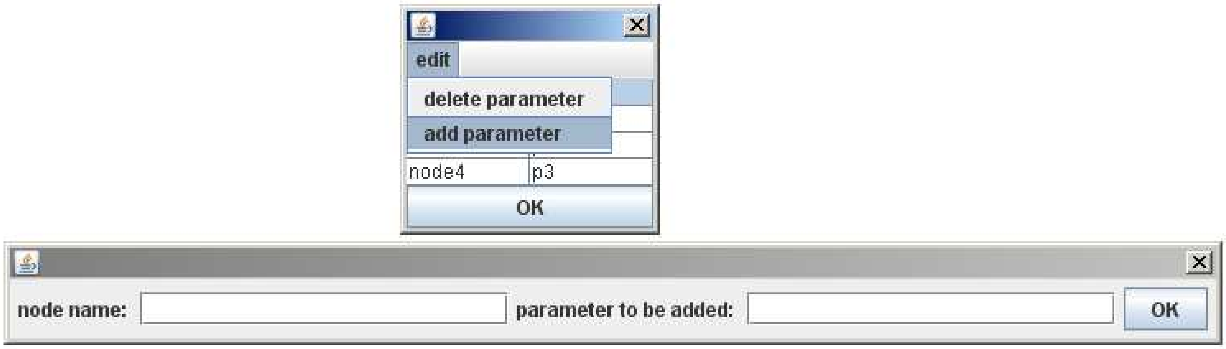


Figure 26: Right click on task tree then select “add parameter”.

Table 9: Functions for task tree.

Mouse Motion	Task Tree Nodes	Action description
Single Click on Tree Node	Main Tasks, Tasks, Sub Tasks, Controlling Tasks	The sub tree of node will be expanded, if there’s a sub tree.
Mouse Drag and Drop on Tree		The Tree will be relocated to the dropped position.
Double Click on Tree Node	Tasks, Sub Tasks	A corresponding visualization will be displayed; the display will be configured based on the number of visualization being opened. The visualization is pre-fabricated but can be modified by controlling tasks.

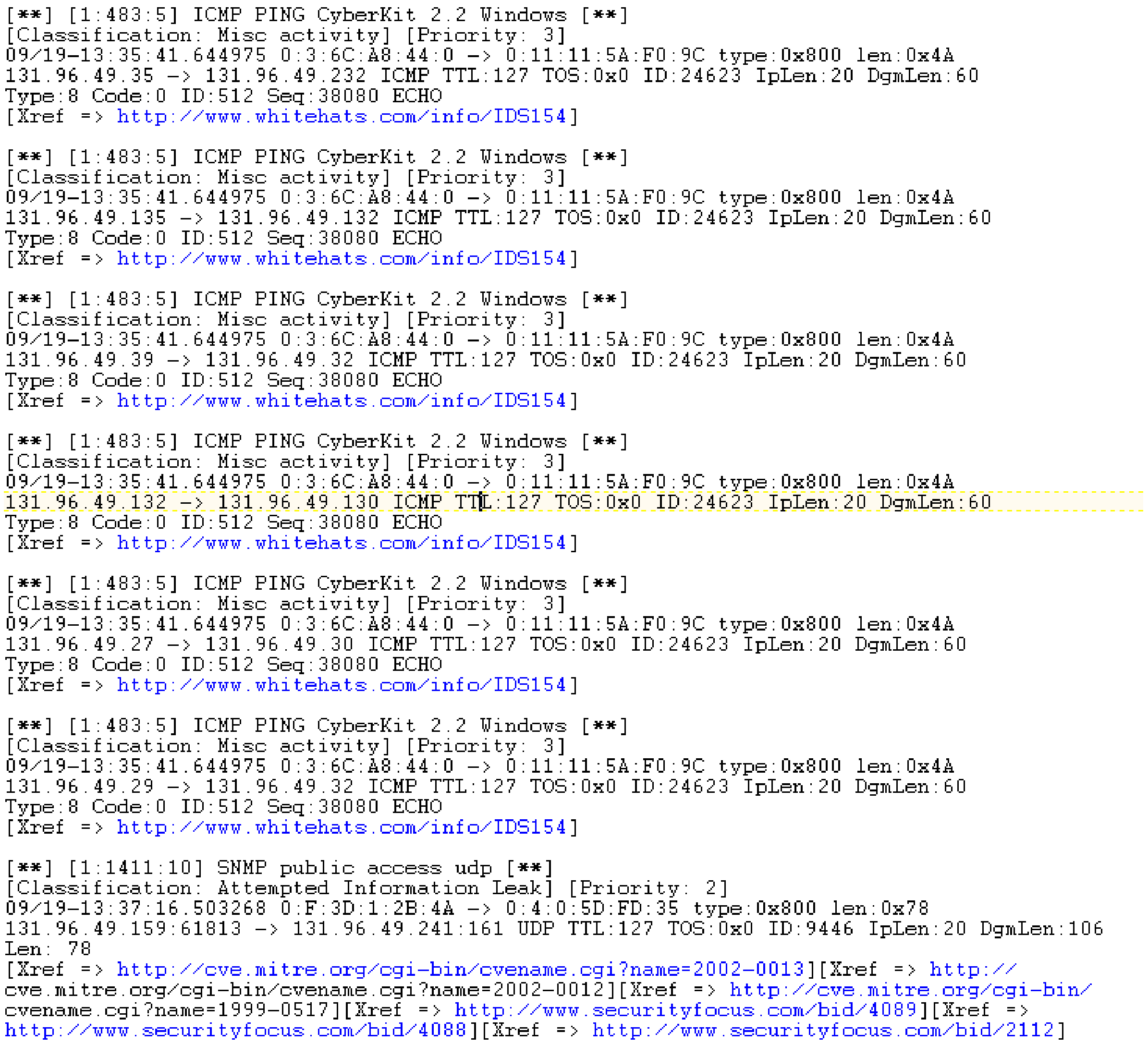
	Controlling Tasks	The control task will modify the corresponding visualization. The controlling tasks are pre-fabricated but can be modified.
Double Click on Tree Node the Second time	Tasks, Sub Tasks	The opened visualization will be removed, and visualization arrangement will be reconfigured
Mouse Enter on Tree Node	Main Tasks, Tasks, Sub Tasks, Controlling Tasks	The parameters defines the nodes will be displayed on the left bottom corner (Title Bar)
Mouse Leave on Tree Node	Main Tasks, Tasks, Sub Tasks, Controlling Tasks	Title Bar will be set to null. Nothing will be displayed
Right Click—select add task		Simply enter the parent node and the label for new task node.
Right Click—select delete task		Enter the label of the node to be deleted
Right Click—See Parameter		Click to see all the parameters, the first column displays the node names; the

		second column displays the related parameter.
Click Edit then Delete Parameter – See Parameter		Highlight a row in the table, and then click on delete parameter to delete the row.
Click Edit then Add Parameter – See Parameter		Enter a node label and the corresponding parameter. One parameter per row.
Mouse Drag and Drop on Tree Node	Tasks, Sub Tasks, Controlling Tasks	The dragged node and its child nodes will be added as child nodes to the dropped node.

5.2.3 Data Table

The only type of data TVDA is currently using is SNORT data. SNORT data will be further processed, and relevant information will be extracted, such as: alert name, classification, date, time, source IP, and destination IP etc. the extracted data are displayed in a table format for the user to view and look up.

The data shown on the data table will be displayed in the visualization. User may choose to open their saved data file by using the “file” menu item on the menu bar. In the following figure, we used a simple SNORT file gathered within the GSU network. The file has a format as the following figure showed:

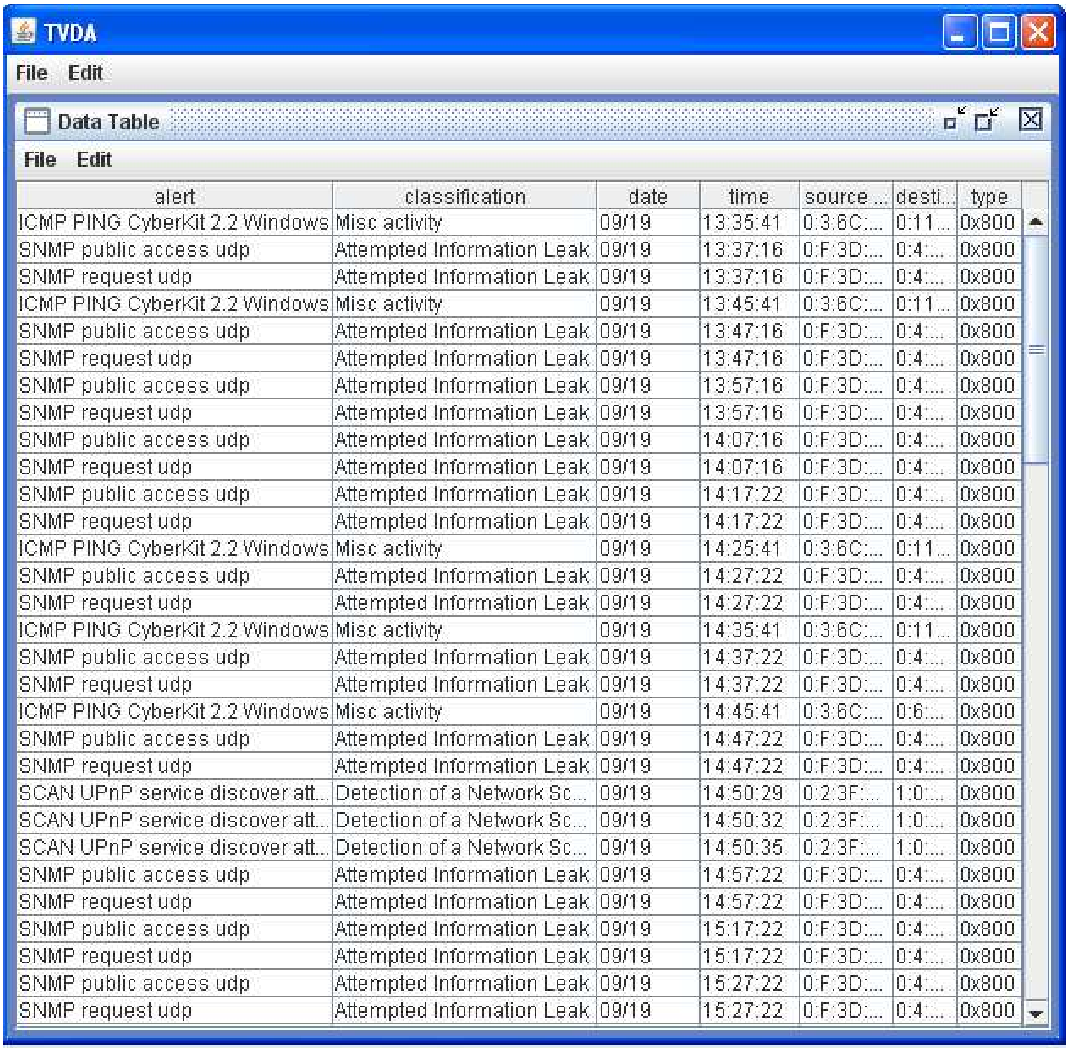


```
[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.35 -> 131.96.49.232 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.135 -> 131.96.49.132 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.39 -> 131.96.49.32 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.132 -> 131.96.49.130 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.27 -> 131.96.49.30 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/19-13:35:41.644975 0:3:6C:A8:44:0 -> 0:11:11:5A:F0:9C type:0x800 len:0x4A
131.96.49.29 -> 131.96.49.32 ICMP TTL:127 TOS:0x0 ID:24623 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:38080 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:1411:10] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/19-13:37:16.503268 0:F:3D:1:2B:4A -> 0:4:0:5D:FD:35 type:0x800 len:0x78
131.96.49.159:61813 -> 131.96.49.241:161 UDP TTL:127 TOS:0x0 ID:9446 IpLen:20 DgmLen:106
Len: 78
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://
cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://cve.mitre.org/cgi-bin/
cvename.cgi?name=1999-0517][Xref => http://www.securityfocus.com/bid/4089][Xref =>
http://www.securityfocus.com/bid/4088][Xref => http://www.securityfocus.com/bid/2112]
```

Figure 27: A portion of the SNORT alert file to be used by the system.

The user may choose to use a SNORT alert file with any length. The SNORT file can come from any network with any kind of alert. The data table displayed in figure 28 contains the processed data; processed data are displayed in a table format. Each column is one array that contains the relevant alert information. User may choose to scroll down the data table to see all available processed data. In this case, the raw data is the SNORT alert file; we processed the raw data using java regular expressions; the extracted data which displayed in the data table view is the extracted data.

The figure above is only a part of the SNORT data file we used for our system; the actual file contains more than 40 paragraphs.



alert	classification	date	time	source ...	desti...	type
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	13:35:41	0:3:6C:...	0:11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:37:16	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:37:16	0:F:3D:...	0:4:...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	13:45:41	0:3:6C:...	0:11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:47:16	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:47:16	0:F:3D:...	0:4:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:57:16	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:57:16	0:F:3D:...	0:4:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:07:16	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:07:16	0:F:3D:...	0:4:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:17:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:17:22	0:F:3D:...	0:4:...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:25:41	0:3:6C:...	0:11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:27:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:27:22	0:F:3D:...	0:4:...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:35:41	0:3:6C:...	0:11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:37:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:37:22	0:F:3D:...	0:4:...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:45:41	0:3:6C:...	0:6:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:47:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:47:22	0:F:3D:...	0:4:...	0x800
SCAN UPnP service discover att...	Detection of a Network Sc...	09/19	14:50:29	0:2:3F:...	1:0:...	0x800
SCAN UPnP service discover att...	Detection of a Network Sc...	09/19	14:50:32	0:2:3F:...	1:0:...	0x800
SCAN UPnP service discover att...	Detection of a Network Sc...	09/19	14:50:35	0:2:3F:...	1:0:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:57:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:57:22	0:F:3D:...	0:4:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	15:17:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	15:17:22	0:F:3D:...	0:4:...	0x800
SNMP public access udp	Attempted Information Leak	09/19	15:27:22	0:F:3D:...	0:4:...	0x800
SNMP request udp	Attempted Information Leak	09/19	15:27:22	0:F:3D:...	0:4:...	0x800

Figure 28: A data table.

5.2.4 Visualization

Visualizations will appear when user clicks on the sub-task nodes on the task tree. User may control the visualization by clicking on controlling task nodes. Depending on the task tree's design and the user's intention, several pre-fabricated visualizations will be displayed, some examples are: scatter plots, bar chart etc. Our system emphasis on spatial constraints, in order to increase the visual quality of the design.

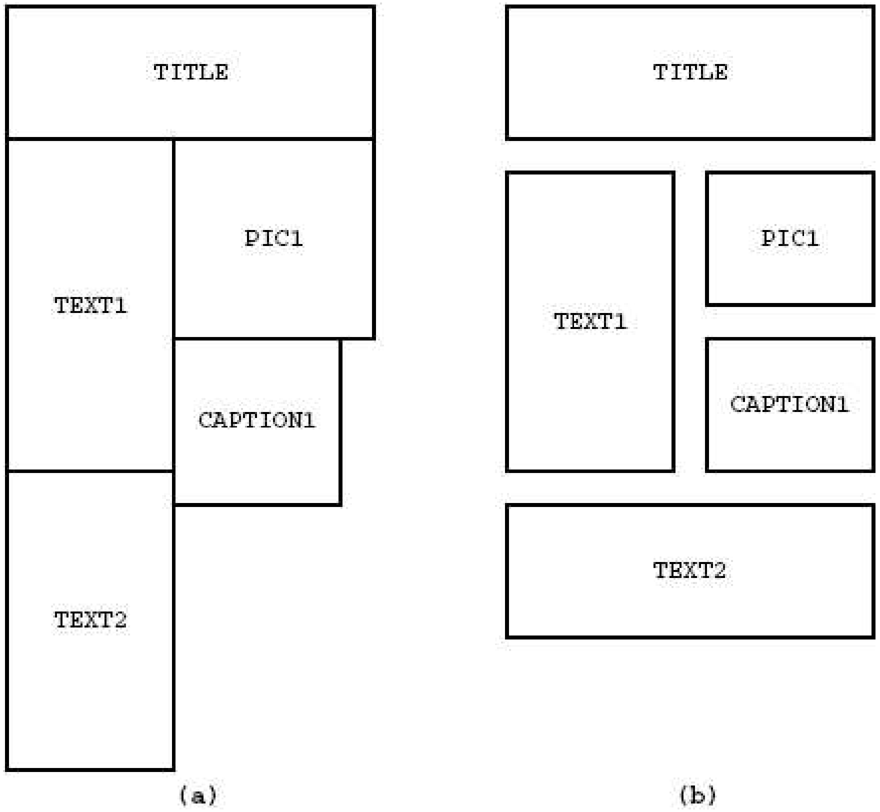


Figure 29: (a) A simple layout that can be generated by a system that only considers abstract relationships between components. (b) A layout of the same components where additional spatial constraints are enforced so that each component completely fills a regular grid and leave margins between the elements. [44].

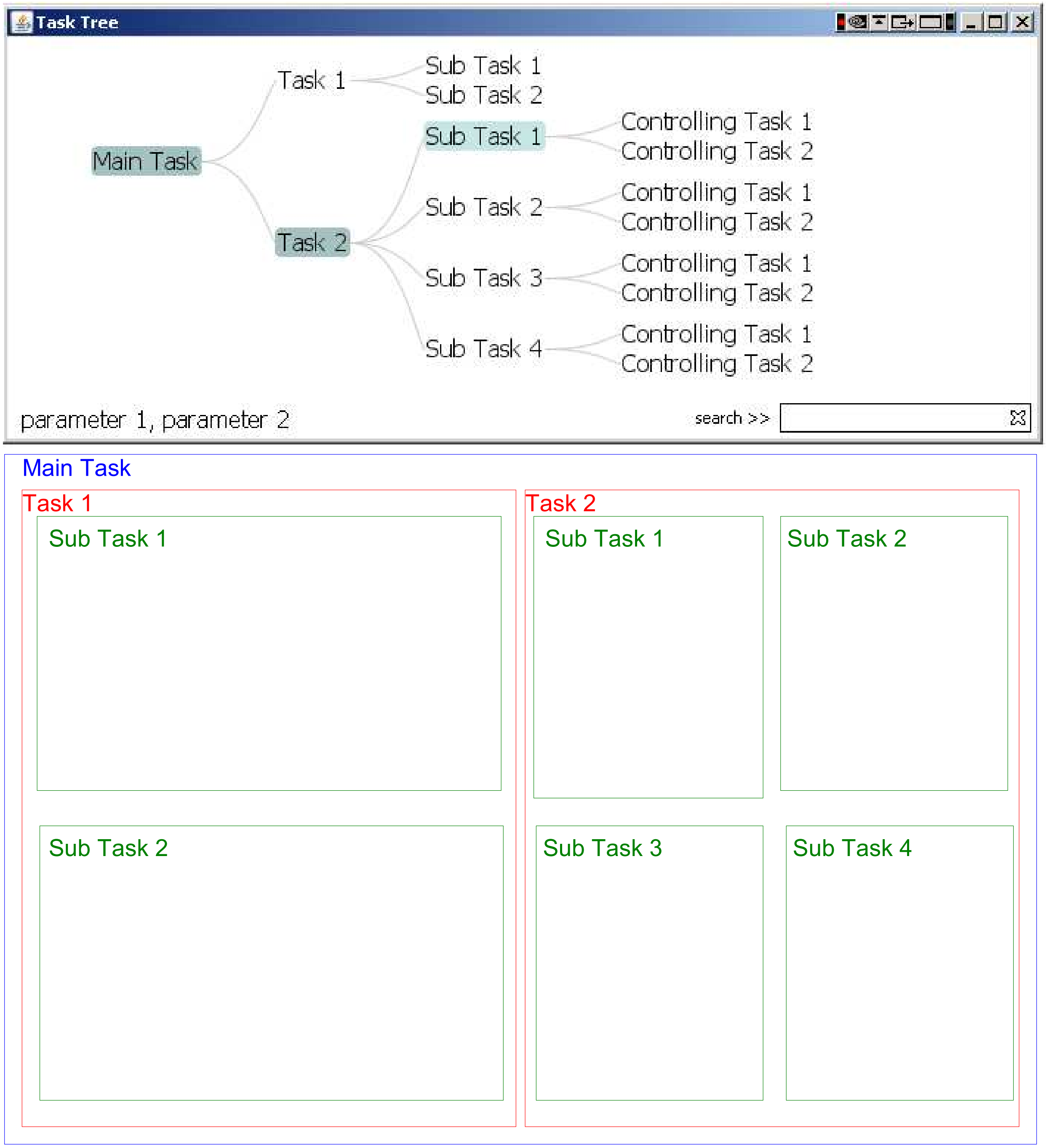


Figure 30: Visualization tree map arrangement and associated task tree.

Spatial constraints are relations that directly express the geometric structure of the presentation [44]. Proper design of spatial constraints can maximize the visual quality of a visualization design. Figure 16 exemplifies what might happen in a system that employs abstract constraints without spatial constraints. A system that considers only abstract constraints will not be able to generate layouts with the same aesthetic appeal as systems that consider both because the system has no visual restrictions on where components of the layout are placed [44].

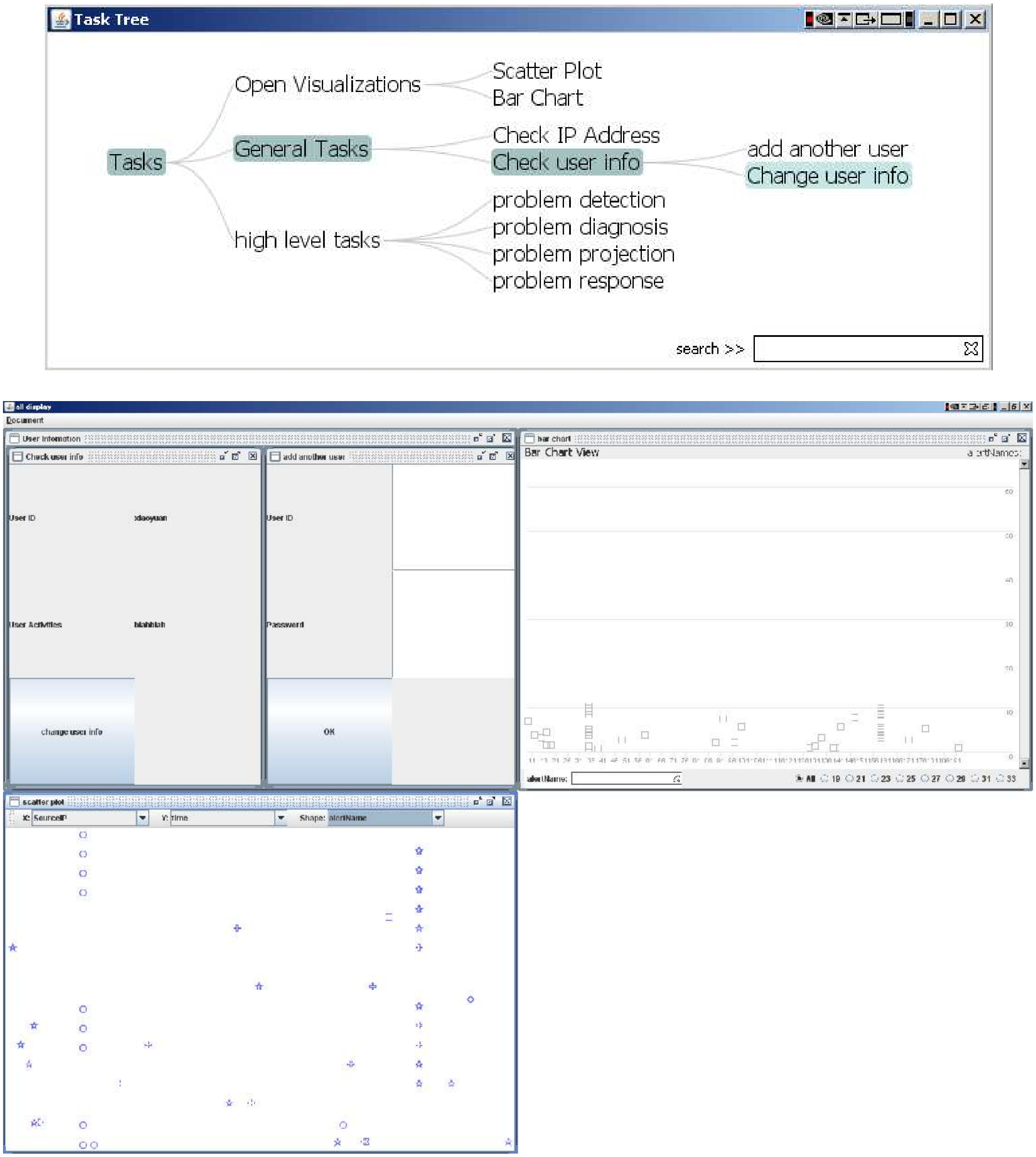
The visualizations are arranged on a single display; we use tree map to organize our visualizations. [89] The tree map visualization technique makes efficient use of the available display space, maximize the visualization quality by proper use of spatial constraints [110]. The visualization maps hierarchies onto a rectangular region in a space-filling manner, so all opened visualization can be properly fitted into the display. This efficient use of space allows large hierarchies to be displayed and facilitates the presentation of semantic information. [89]

Each task will be displayed within their parent task, and arranged sequentially based on the order which they have been clicked to open. While the visualization has been opened, User may double click on the node again to close the visualization; the display will be arranged again to accommodate tree map rule. The figure below describes the arrangement as well as the association with the task tree.

6 Case Study

In this case study, we used SNORT alert information gathered during a short period of time.

6.1 Visualization Management



User ID xdaoyuan

User Activities blahblah

User ID

Password

X: SourceIP

Y: time

Shape: alertName

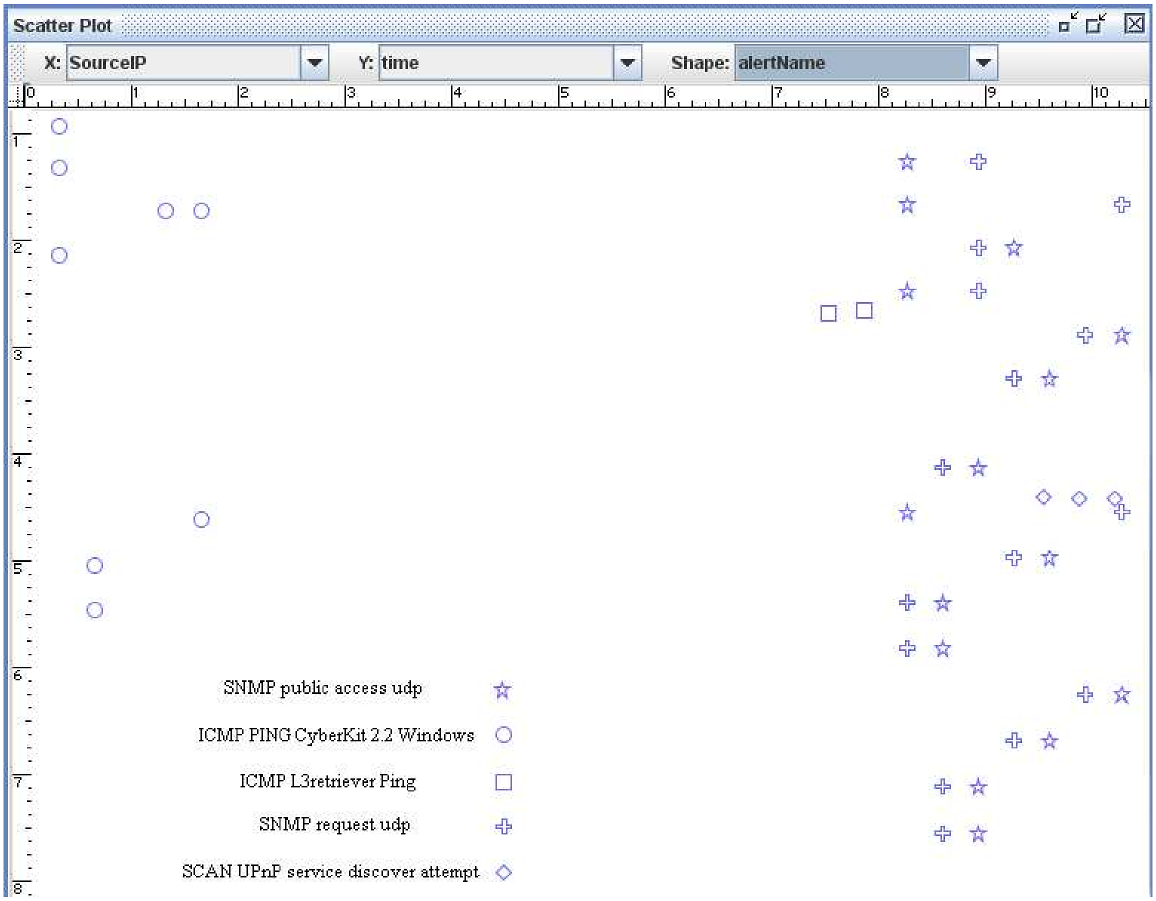
Figure 31: Task tree and tree-map arrangement of the opened visualizations.

Once the data table is loaded, the task tree will be triggered. By clicking on each sub-task, the corresponding pre-fabricated visualizations will be displayed accordingly. The purpose of this design is to show the data we gathered from our network using SNORT during a short period of time. The data table is the same as referred back to section 4.2. In this example, user has clicked on “ scatter plot”, “Bar chart view” then “check user info”, and “add another user”. Since “add another user” is a controlling-task for “check user info”, it is displayed within the frame with “check user info”.

The following Scattered Plot (figure 32) showed source IP on X-axis, time of each packet’s capture on Y-axis, the shapes in this case represent the five different alert names we have received. Evaluators may choose to select different values for X-axis, Y-axis and Shape by using the menu options on the tool bar. A legend of different shapes and their represented alert names is displayed on the bottom of this display. User may also choose to mouse over on a certain shape to see the specific alert information.

Different shapes in scatter plot represent different alert types. User can choose different parameters (Source IP, time, date, priority, alert name) for the X and Y axes.

In the non-traditional bar chart view shown in the figure below, the x-axis represents the different Source IP address while the y-axis represents the time the packet has been received in number format. Each packet’s priority is shown in an unequally different color. In this case, 2 different colors are shown. User may mouse over on any shape, and the corresponding alert names will be displayed on the right upper corner of this window.



Scatter Plot

X: SourceIP Y: time Shape: alertName

Figure 32: Scatter plot.

In this window, the denser column mean more alerts from the same IP address, therefore more vigilant IP addresses may easily be identified by looking for the denser columns. User may refer back to the data table for detailed alert information. User may also search for a specific alert name on the display by typing in the alert name in the search bar located on the left bottom corner. Since we only had data gathered within a short period of time, this display is largely empty, user may also choose to modify the y-axis by accessing design file. Direct visual modifications are not available yet.

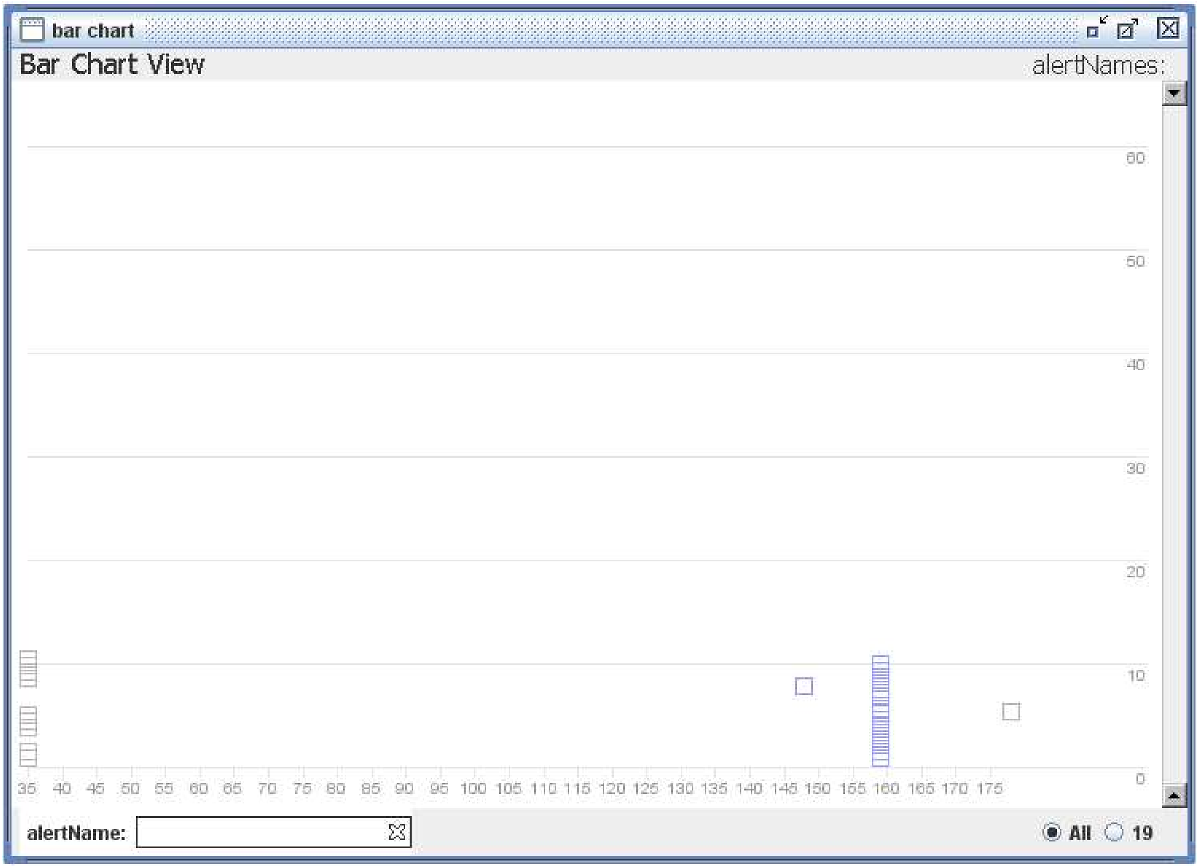


Figure 33: Bar chart view.

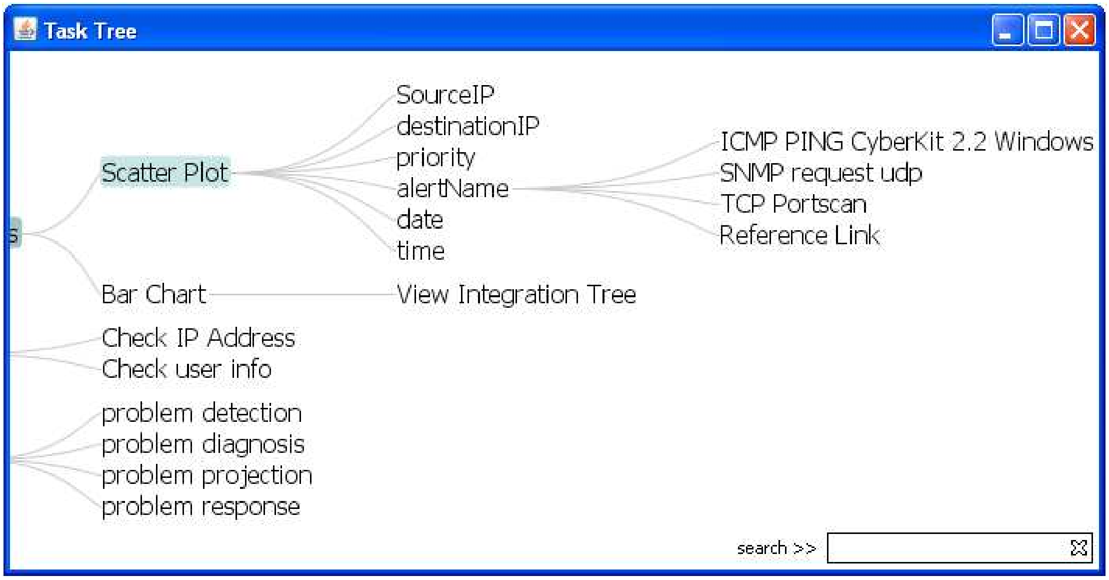


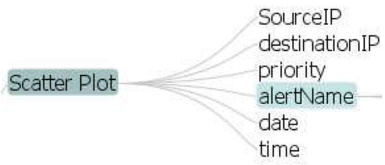
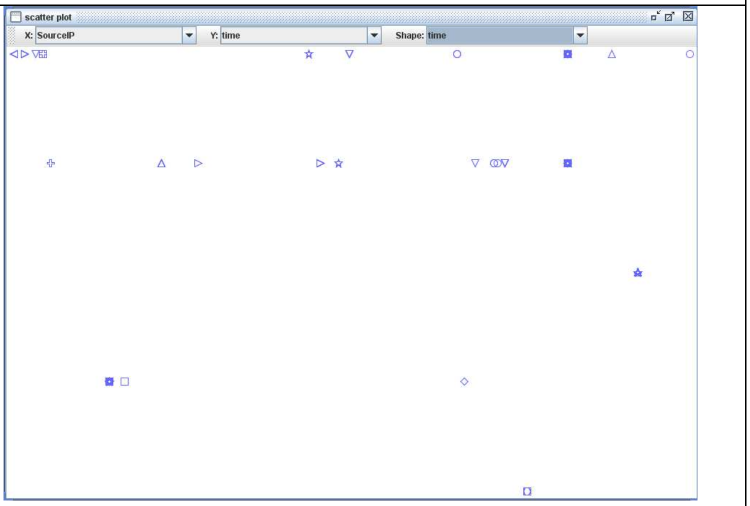
Figure 34: Controlling tasks for scatter plot.

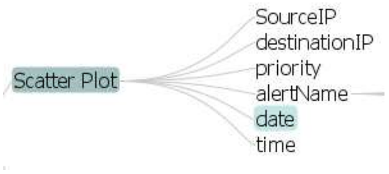
6.2 Visualization Control

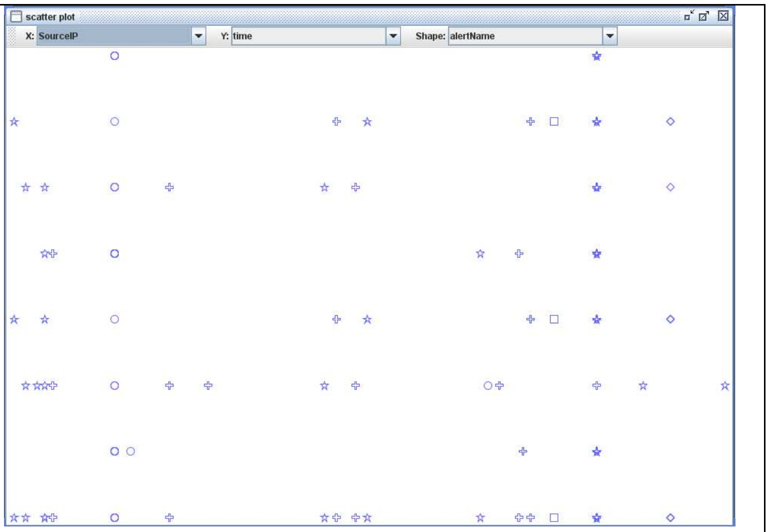
Opened visualizations can be directly manipulated by the controlling task. The controlling mechanisms are pre-defined in this case. Controlling tasks are optional for all visualizations; they are also the leaf nodes of the task tree. In the example of Scatter-plot from the previous section, we list a few controlling task as shown in the figure below.

The table below shows the several types of control-tasks and the corresponding outcome. In other types of designs, user may define other types of control-tasks and hard code them into the program. In our example, six types of control tasks are shown for the visualization: scatter plot. The controlling mechanisms in this case are all pre-defined, and they have full control over the display.

Table 10: Control tasks vs. control of the visualization.

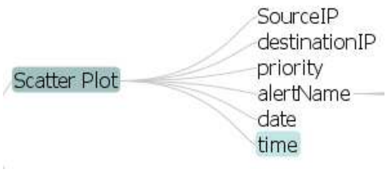
alert name clicked, visualization shows source IP vs. alert names	
 Scatter Plot SourceIP destinationIP priority alertName date time	 scatter plot X: SourceIP Y: time Shape: time
Data clicked, Visualization shows source IP vs. date.	

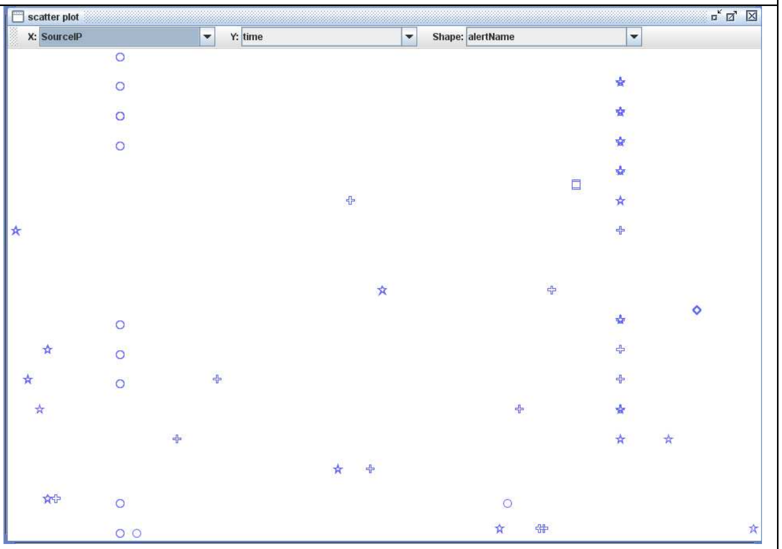



scatter plot

X: SourceIP Y: time Shape: alertName

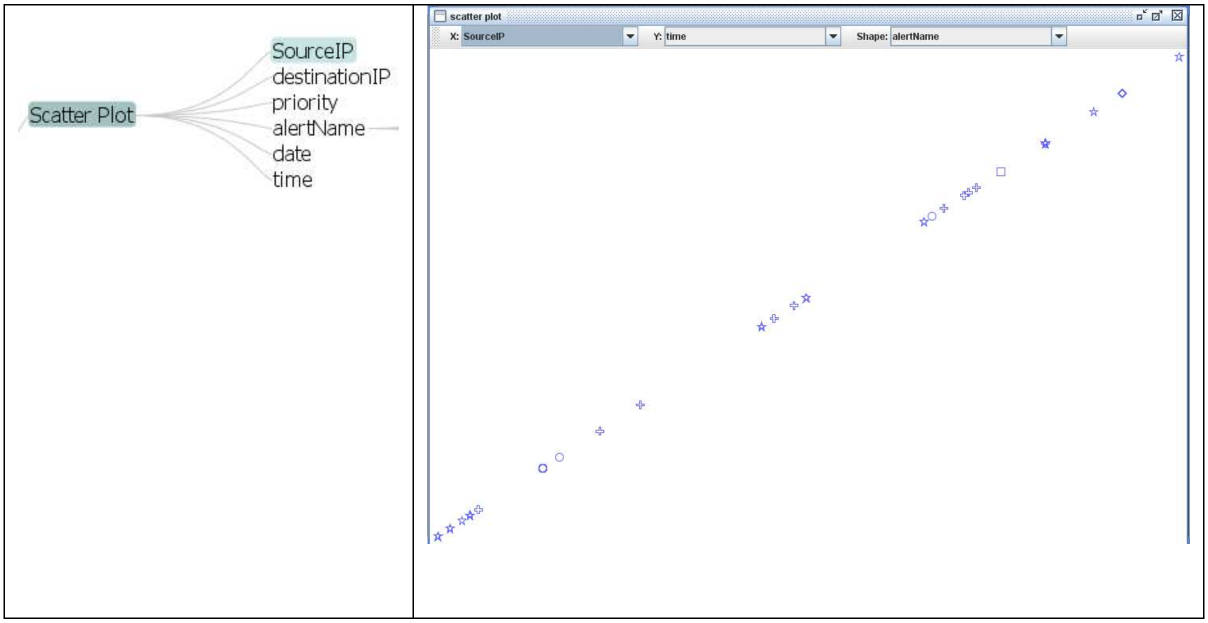
Data clicked, Visualization shows source IP vs. date.




scatter plot

X: SourceIP Y: time Shape: alertName

Source IP clicked. Visualization shows source IP vs. source IP.



X: SourceIP Y: time Shape: alertName

6.3 Visual Integration Tree

The semi-automatic integration tree is generated when the menu item "view visual integration tree" is clicked from the data table view. The integration tree first counts the numbers of frames that have been opened, and records their names. It then calculates the number of components on each visualization frames, such as JMenuBar, JTable. These components as well as the shapes displayed in each visualization frame are the visual patterns for the visual frame. As a result, in the example we have shown in the previous section, the number of visual patterns is 141. The number deviates a little, since Prefuse tree allows partial hiding for un-used tree nodes. In reality, less than 33 tree nodes are displayed for task tree; in turn the design allows less cognitive load on user. Therefore, the main visualization has slightly less than 141 visual patterns.

The figure below shows the integration tree when it is clicked to be opened, this integration tree has the same structure of task tree. Therefore only selected tree nodes will be expanded.

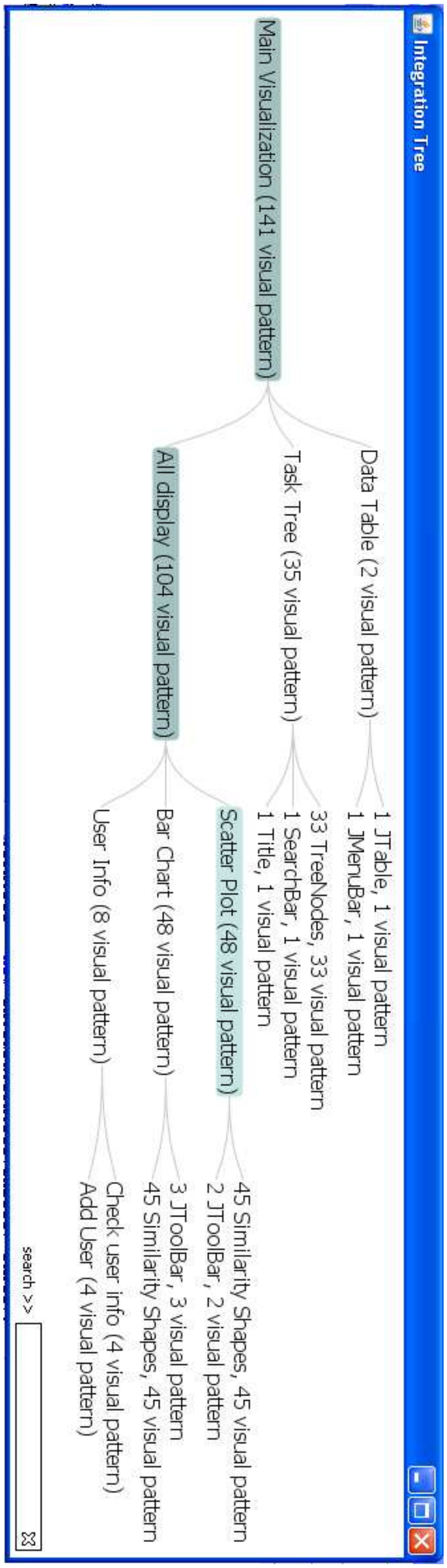

Integration Tree
Main Visualization (141 visual pattern)
Data Table (2 visual pattern)
Task Tree (35 visual pattern)
All display (104 visual pattern)
1 JTable, 1 visual pattern
1 JMenuBar, 1 visual pattern
33 TreeNodes, 33 visual pattern
1 SearchBar, 1 visual pattern
1 Title, 1 visual pattern
Scatter Plot (48 visual pattern)
Bar Chart (48 visual pattern)
User Info (8 visual pattern)
45 Similarity Shapes, 45 visual pattern
2 JToolBar, 2 visual pattern
3 JToolBar, 3 visual pattern
45 Similarity Shapes, 45 visual pattern
Check user info (4 visual pattern)
Add User (4 visual pattern)
search >>

Figure 35: Visual integration tree for the case study.

The integration tree is still “partially” automatic at present stage; it only calculates and displays all the critical visual factors for visualization. Whether the visualization has been properly designed, integration simply provides little connection with the design. Ideally, we would like to make the integration tree to be the ultimate ruler for all visual frames generated under the same main visualization. In a way, such as when one visual frame has exceeded the limit of visual patterns, it will be warned or automatically modified by the integration tree. More details will be discussed in the future work section of this thesis.

6.4 Visual Mapping Complexity Scoring System

Visual mapping scores are assigned not by the system, but rather by the users, or the evaluators. Therefore the system do not provide any hard-coded complexity mapping scoring system, it will show the standards for building complexity scoring tree, then leave the rest to the users. The scoring code and standard can be access from menu bar options, as shown in the figure below.

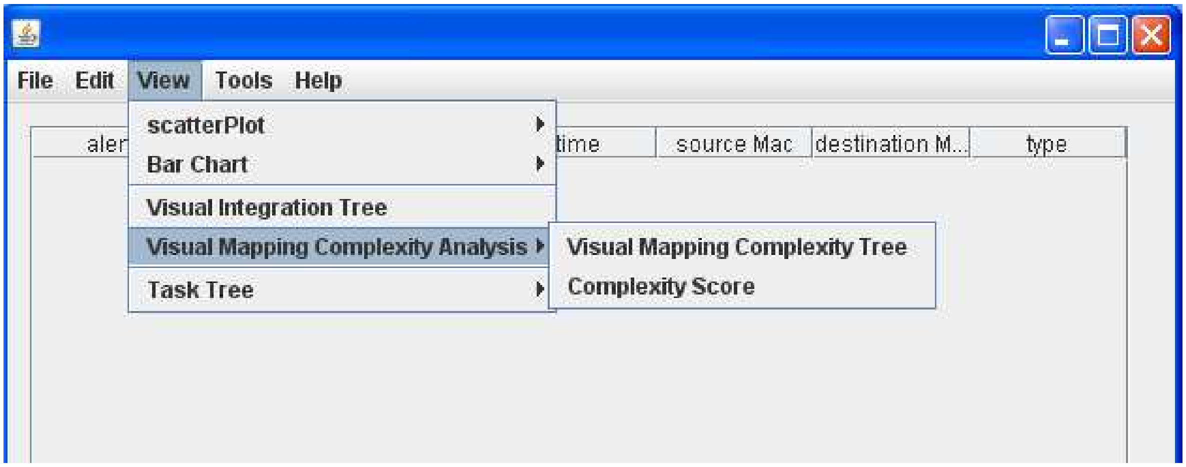
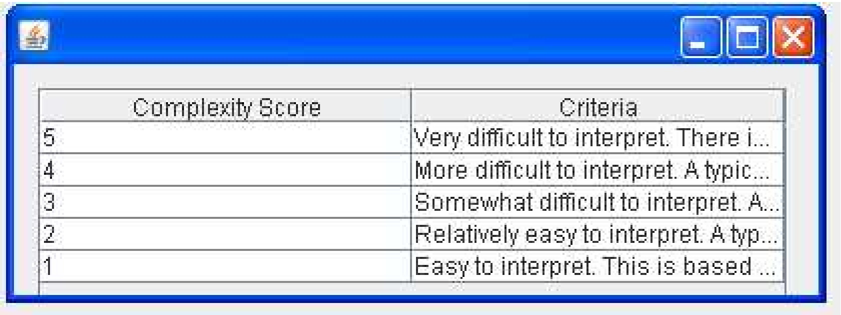


Figure 36: Accessing complexity score system.

Once the visual complexity tables are built, they can be compared against with similar well established designs, such as TNV. The visual complexity system is designed to serve the designer, to have a better sense of their design in comparison with the other comparable ones.[111]



Complexity Score	Criteria
5	Very difficult to interpret. There i...
4	More difficult to interpret. A typic...
3	Somewhat difficult to interpret. A...
2	Relatively easy to interpret. A typ...
1	Easy to interpret. This is based ...

Figure 37: Visual complexity score table.

Visual scoring system is rather a user-study type of tool. Visual scoring system can be used either during the design process or when the design is completed.

7 Comparisons with TNV and RUMINT

Using the visual integration tree discussed in the previous sections, we evaluated our system and two other established works in the field. The works we listed here are TNV [38] and RUMINT [112]; they are both very well designed networking security monitoring tools.

In TNV [38] system, there are 5 different dimensions in the main visual matrix frame; and 3 different dimensions in the port visualization frame (figure below).

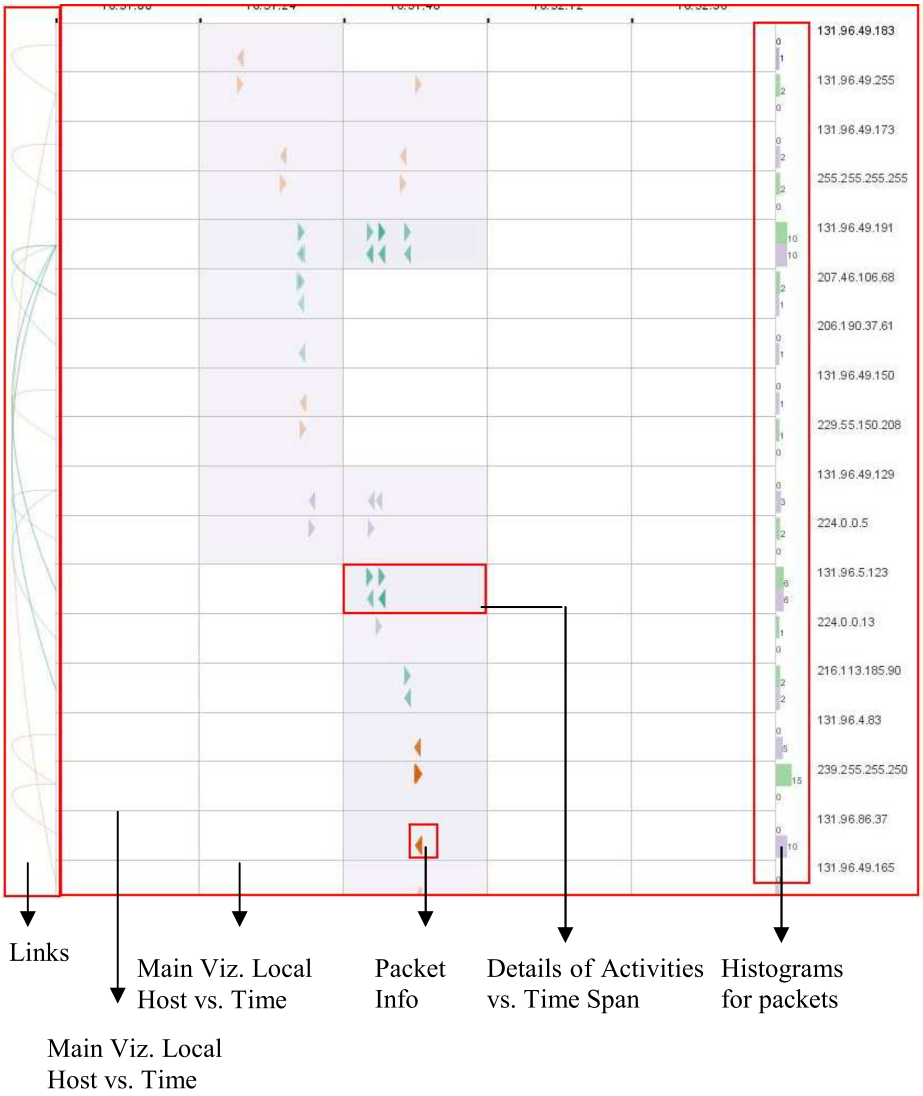


Figure 38: Five different dimensions in TNV main visualization matrix.

In the figure shown above, there are five different dimensions: (a) Histograms for packets: categorized based on their shape/size/color. (b) Details of activities; categorized based on color/shape. (c) Main visualization matrix: categorized based on X-Y coordinate; in this case, they are local host vs. time. (d) Package information triangles: categorized based on shape/size/color. (e) Links: categorized based on shape/size/color.

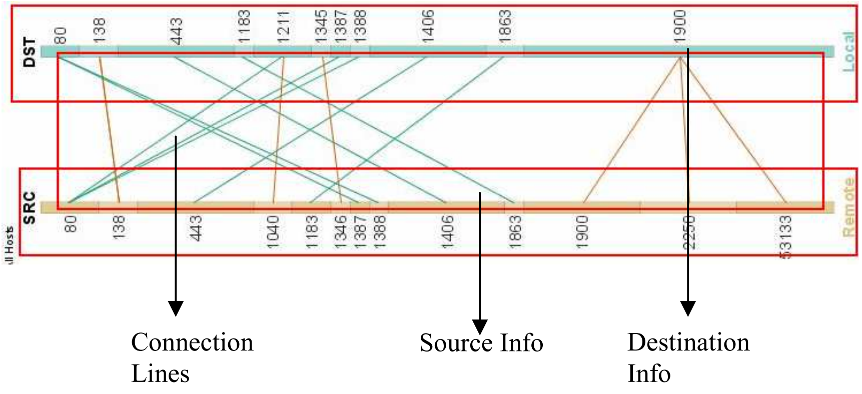


Figure 39: Three different dimensions in port visualization.

The three different dimensions are: (a) Source and destination information: categorized based on the coordinate (vertical axis). (b) Connection lines between the two axes: categorized based on shape/color.

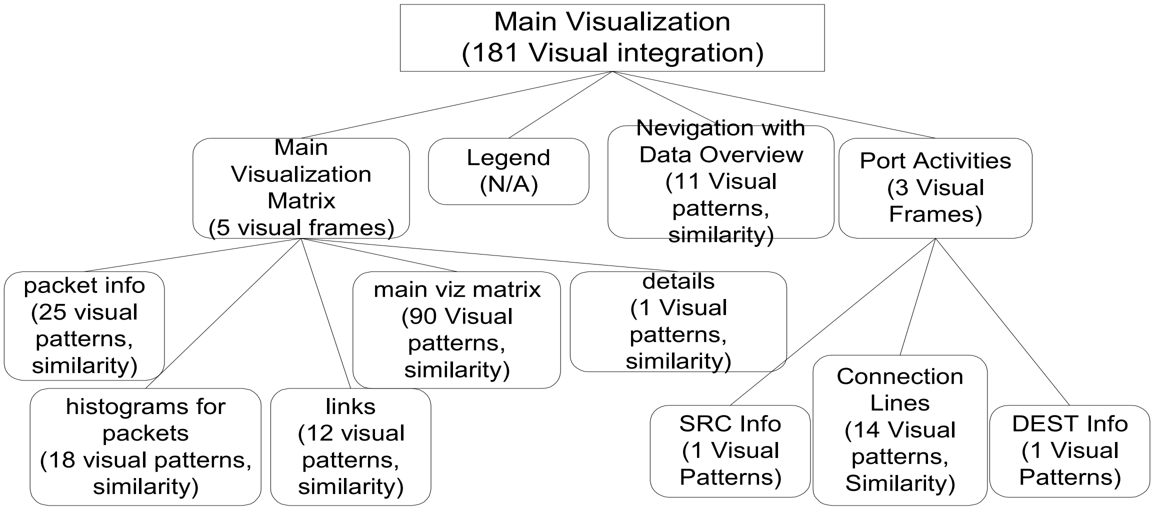

Main Visualization
(181 Visual integration)
Main Visualization Matrix
(5 visual frames)
Legend
(N/A)
Nevigation with Data Overview
(11 Visual patterns, similarity)
Port Activities
(3 Visual Frames)
packet info
(25 visual patterns, similarity)
main viz matrix
(90 Visual patterns, similarity)
details
(1 Visual patterns, similarity)
histograms for packets
(18 visual patterns, similarity)
links
(12 visual patterns, similarity)
SRC Info
(1 Visual Patterns)
Connection Lines
(14 Visual patterns, Similarity)
DEST Info
(1 Visual Patterns)

Figure 40: Visual integration tree for TNV.

In the integration tree shown above: each of the child nodes has a different number of visual patterns and the different Gestalt laws they are based on. Every parent node has a different frame number; the number of visual integration associated with the parent node is calculated by multiplying the number of visual frames and the sum of the visual patterns.

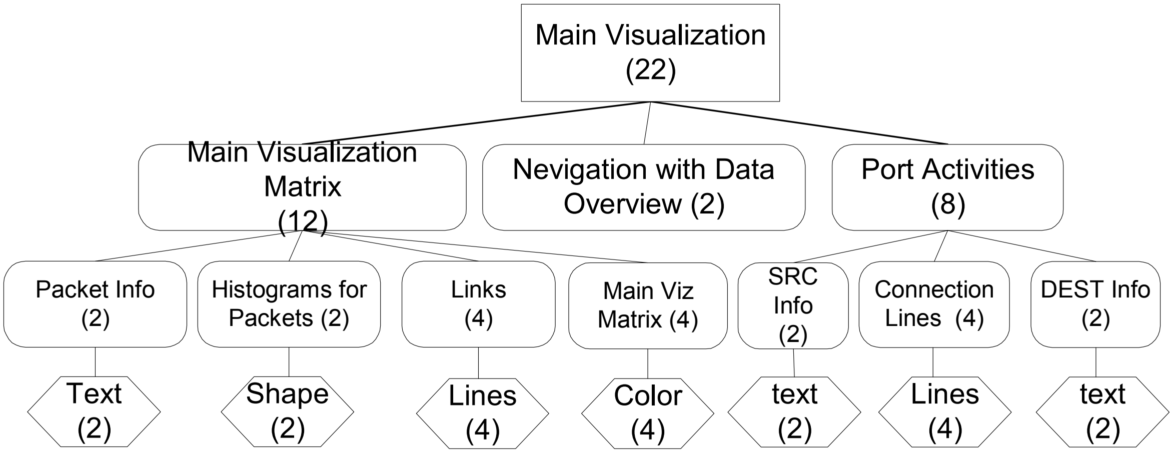


Figure 41: Visual mapping complexity tree for TNV.

In the visual mapping complexity tree for TNV, each number below the visual units are the complexity scores based on table 2. Every parent node's score is calculated as the sum of its children's score.

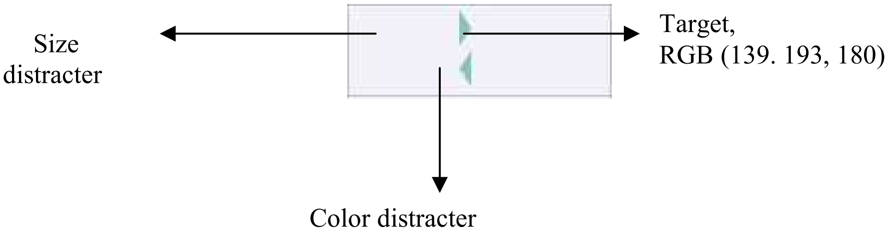


Figure 42: A portion of the main TNV visualization.

In the TNV example we mentioned before, we took a portion of the main visualization and calculated the resultant color, motion, size and orientation values; as table 4 shows. The color distracter is the background color (against the targeted object's color), and the size distracter is the similar object presented in the very adjacent location.

Table 11: Target-distracter difference scores for TNV.

	Color	Motion	Size	Orientation
Target-distracter difference scores	0.2850	N/A	0	1

Rumint is an open source network and security visualization tool developed by Gregory Conti et al. [112] Rumint accomplishes the security visualizing tasks by loading pcap datasets and capture live traffic.

In the following example, we took two screen shots from Rumint after it finishes capturing the 949 packets from the sample dataset.

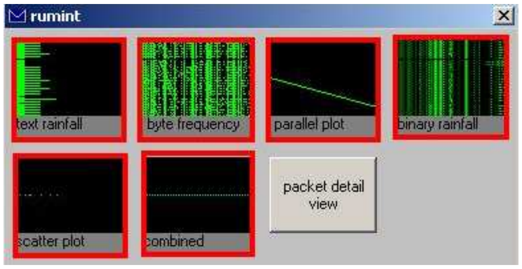


Figure 43: Rumint thumbnail overview.

The overview screenshot contains 6 different visual frames, using the methods we mentioned in the previous section; we produce the following complexity tree for Rumint. Since Rumint is a rather complicated visualization system with considerable amount of frames and dimensions, we minimized our trees to fit into this paper.

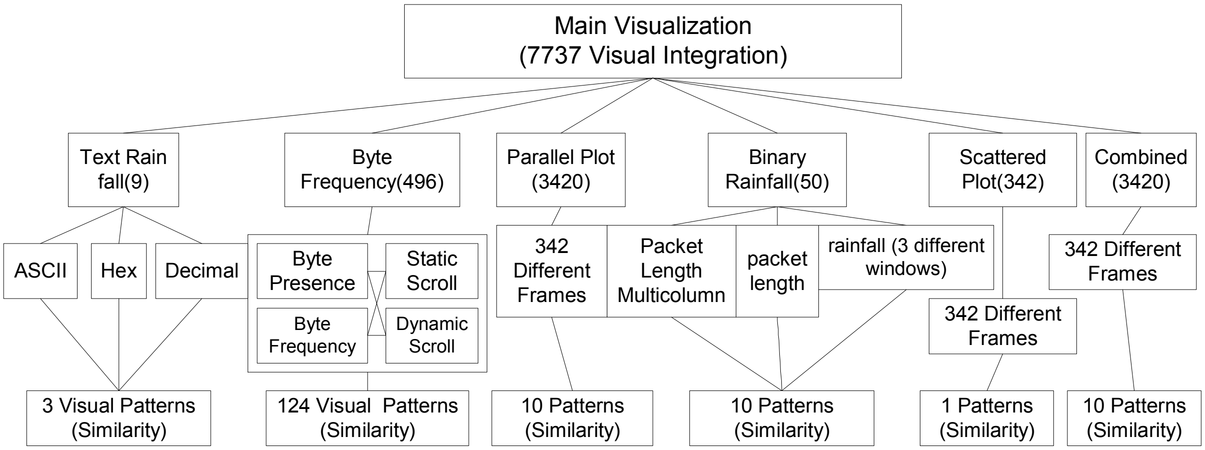


Figure 44: Visual integration tree for Rumint.

The simplified visual complexity tree is shown in figure 12.

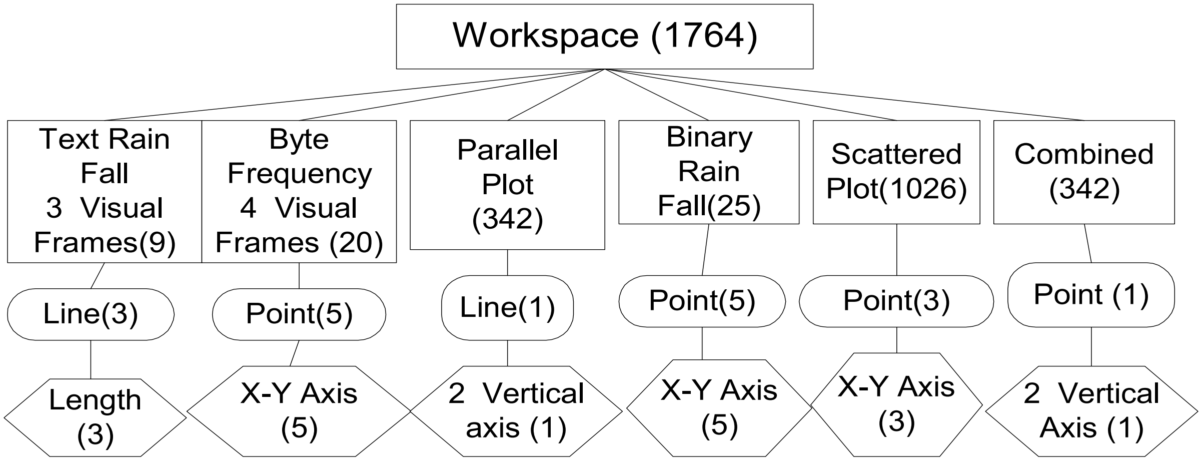


Figure 45: Visual mapping complexity tree for Rumint.

No visual mapping complexity tree will be provided for our work; we will leave this part to the user studies.

Table 12: Evaluation results in metrics format for RUMINT.

Visualization	Security Architecture	Visual Search Guidance				Number of Data Points
		Color	Motion	Size (pix^2)	Orientation	
RUMINT	overview	0.33333	n/a	0.43064	0	1797
	Byte Frequency	0.33333	n/a	0.32169	0	421

A detailed visual integration tree for our case study has been discussed in the previous sections. The figure below showed a manually calculated visual integration tree; the results are the same as the automated calculated one.

The scatter plot we provided uses shapes to represents different alert names, instead of color. Since in reality, different systems are built for different purposes; typical designers and users have different intentions. It's hard to rely on numerical systems to rank the visualization systems. Our method of evaluation only provides general guidelines to designers; the method should co-exist with traditional user studies and other methods.

The figure below (figure 46) is a manually generated visual integration tree for our case study, the results showed are exactly the same as the machine calculated visual integration tree.

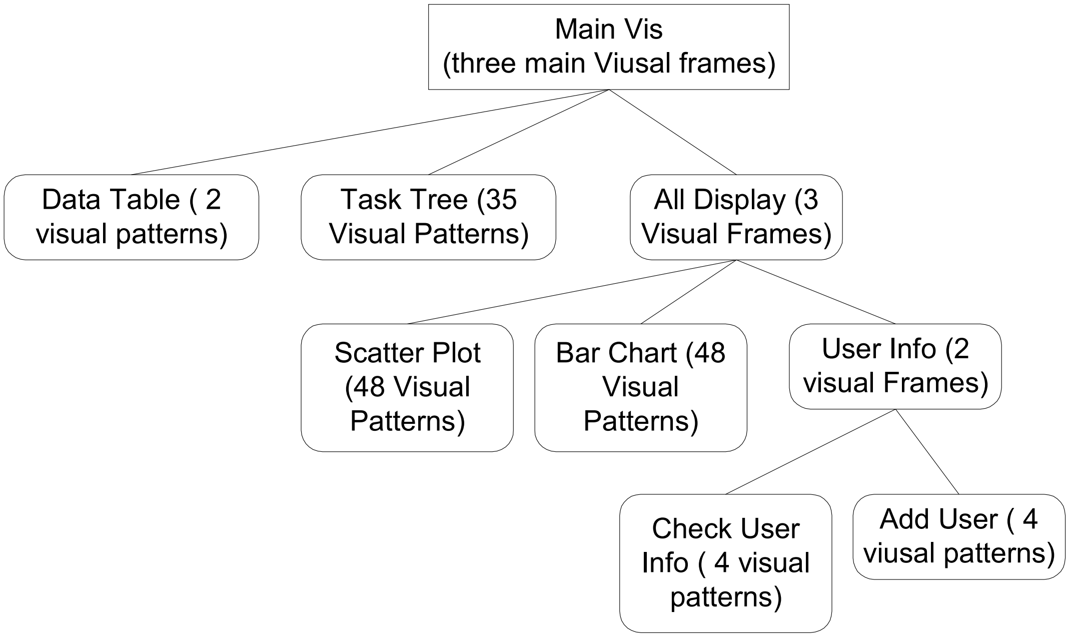

Main Vis
(three main Viusal frames)
Data Table (2 visual patterns)
Task Tree (35 Visual Patterns)
All Display (3 Visual Frames)
Scatter Plot (48 Visual Patterns)
Bar Chart (48 Visual Patterns)
User Info (2 visual Frames)
Check User Info (4 visual patterns)
Add User (4 viusal patterns)

Figure 46: Visual integration tree for our case study.

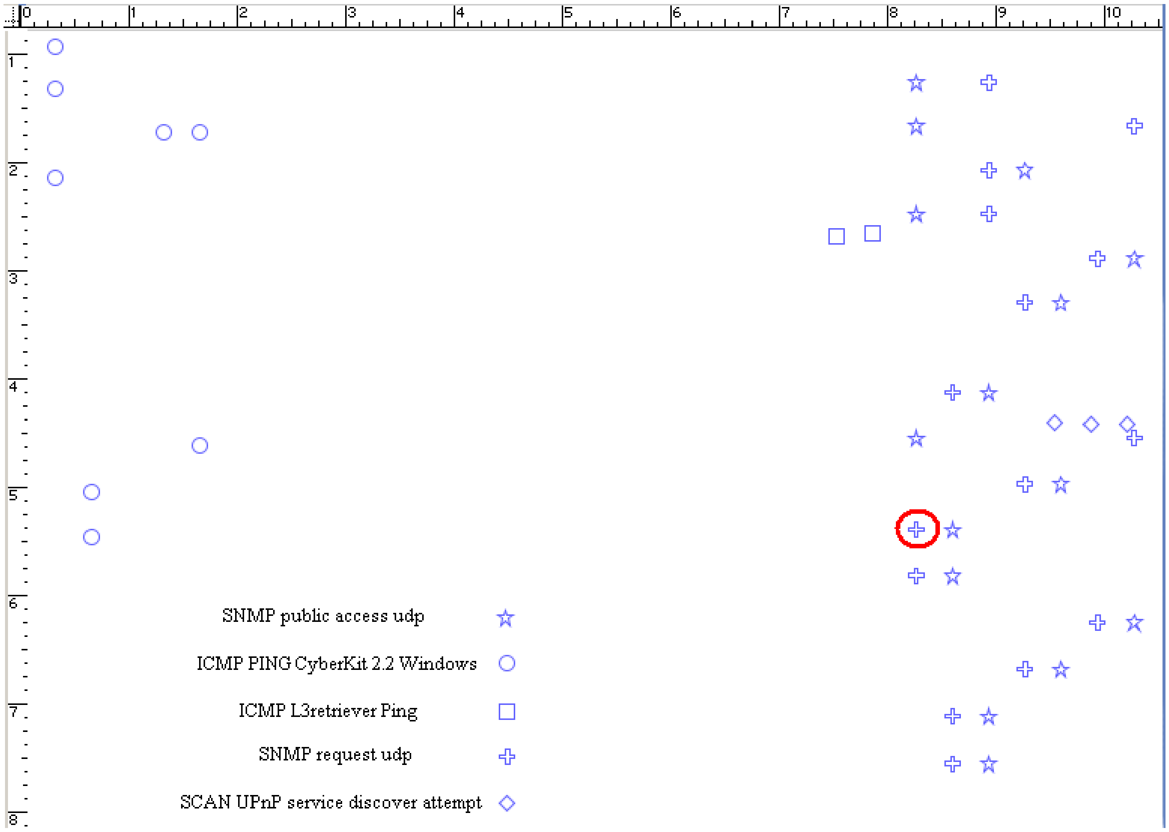


Figure 47: Target distracter analysis for our system.

Table 13: Evaluation results in metrics format for our case study.

	Color/shape	Motion	Size	Orientation
Target-distracter difference scores	1	N/A	0	1

8 Conclusion

The paper has three main elements: complexity analysis, task-centered visualization design strategies and task-centered visualization implementation.

8.1 Complexity Analysis

We have presented a systematic methodology to measure the complexity of visualization. Here the complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, number of visual units, and the efficiency of visual search. These measures are derived from well established psychological theories. Together they indicate the amount of cognitive load involved in comprehending a particular visualization.

The complexity analysis is particularly useful during the design phase before any user studies can take place. We have demonstrated the application of this method by using it to analyze two computer security visualization programs: TNV and Rumint. This method does not provide a rating system for comparing competing visualizations; it focuses on helping designers to explore different ways to reduce the complexity. For example, by increasing the target-distracter differences, or reducing the number of dimensions per visual unit.

Overall, the analysis is not a comprehensive analysis of the design, but a rather focused one. It should be combined with other heuristic evaluation methods, especially the user studies.

8.2 Task Centered Visualization Design

We also discussed a task centered visualization design framework, in which tasks are explicitly identified and organized and visualizations are constructed for specific tasks and their related data parameters. The center piece of this framework is a task tree which dynamically links the raw data with automatically generated visualization. The task tree serves as a high level interaction technique that allows users to conduct problem solving naturally at the task level, while still giving end users flexible control over the visualization construction.

The design guidelines offered a frame work of building a design gallery style visualization interface that allows users to compare and select from multiple visualizations that are automatically generated. A significant challenge is to develop a visualization engine that helps automatically generate visualizations given a task and its related parameters. The key is to codify the many design rules from the visualization research literature and to develop a systematic method to evaluate and optimize the visualization. Our previous work on visualization complexity analysis [90] can be used as the basis for the evaluation and optimization.

8.3 Implementation

The implementation focused on security visualization using SNORT data gathered during a short period of time. We showed the visualization generation, management and organization. In order to demonstrate the usage of our implementation, a case study with real-time data and the corresponding visual integration tree was provided and a brief comparison between our case study and two other established works was also provided.

9 Future Works

There are numerous potentials to perfect the current design; even though certain changes are still un-predictable at present stage. In this section we will list a few possible improvements.

Our future work includes developing a design gallery style visualization interface that allows users to compare and select from multiple visualizations that are automatically generated. A significant challenge is to develop a visualization engine that helps automatically generate visualizations given a task and its related parameters. The key is to codify the many design rules from the visualization research literature and to develop a systematic method to evaluate and optimize the visualization. Our previous work on visualization complexity analysis [113] can be used as the basis for the evaluation and optimization. Finally, we will develop an evaluation plan to test the effectiveness of the discussed framework, working with domain experts in the field of computer security.

9.1 Visualization Engine

The discussed system includes a visualization engine to automatically select a list of visualization design choices for users to choose from. Before constructing visualization, users are required to select a task and its related data parameters. With this information, the engine will search the three visualization dictionaries to find possible (visual structure, data structure, task) mappings and ((visual unit, visual attribute), data attribute, task) mappings. The mappings with high accuracy, utility, and efficiency scores will be selected. A simple weighted sum of scores can be used to calculate the final ranking. More sophisticated ranking calculation will be investigated.

The visualization engine and the design gallery interface provide the crucial connection between the visualization dictionaries and the visualization construction process. Because the scores in the visualization dictionaries are assigned based on psychological theories and empirical studies, together the engine and the interface provide the theoretical and empirical guidance for visualization design.

The long-term goal is to expand this visualization engine into an automatic visualization engine so that it can automatically generate visualizations based on user specified tasks and data parameters. Building an automatic visualization engine also has theoretical significance. As Kosslyn [89] points out, “One way to systematically develop a program of empirical research is to consider how one would program a computer to emulate an expert human graph designer.”

Using the visualization tool to collect user data and conduct empirical studies on how visualizations are constructed and used. One of the main purposes of the user studies is to refine the scores in the visualization dictionaries. Many initial scores are based on hypotheses that need to be empirically tested. To support user study, TVDA will provide extensive logging capability that can record and replay the entire visualization construction and exploration session, including visual element selection and composition, task tree configuration, task completion time, etc.

The collected empirical data will be analyzed to help answer the following research questions:

1. How does experience affect the perceived effectiveness of visualization? Do experienced users use more visualization or less visualization?

2. How are the visualizations created by experienced users different from the ones created by novice users? How to use this knowledge to train novice users to create better visualizations?
3. How do the visualizations created by users change over time as they become more experienced? Is there any pattern in such changes?
4. What are the domain specific and domain independent factors that influence the effectiveness of visualization? And how can these factors guide the creation of data visualization?
5. How do tasks relate to visualization design layout? How do tasks affect the choices of visual units and visual structures? Is there any pattern we can conclude from tasks and visualization designs?
6. In the end, we want to be able to answer the ultimate question: how do we design a user friendly yet task efficient visualization design.

9.2 User Studies

One of the main purposes of the user studies is to refine the scores in the visualization dictionaries. Many initial scores are based on hypotheses that need to be empirically tested. To support user study, TVDA will provide extensive logging capability that can record and replay the entire visualization construction and exploration session, including visual element selection and composition, task tree configuration, task completion time, etc.

The collected empirical data will be analyzed to help answer the following research questions:

1. How does experience affect the perceived effectiveness of visualization? Do experienced users use more visualization or less visualization?

2. How are the visualizations created by experienced users different from the ones created by novice users? How to use this knowledge to train novice users to create better visualizations?
3. How do the visualizations created by users change over time as they become more experienced? Is there any pattern in such changes?
4. What are the domain specific and domain independent factors that influence the effectiveness of visualization? And how can these factors guide the creation of data visualization?

Table 14: User study activities planned for the target applications.

Application	*Benchmark databases and tasks*	*User study subjects*	*User study activities*
Neural circuitry visualization	• NeuronBank [114] • May include other neuroscience databases in the future	• Neuroscientists and students at Dr. Paul Katz's lab. • NeuronBank users	• Measure interpretation errors • Measure the number of goals achieved for benchmark tasks • Record the number of times a visualization design is selected by users to conduct a task • Record task completion time • User interview • Observation • Expert/novice comparison
Computer security visualization	• DARPA Intrusion Detection Evaluation project [115] and KDD Cup 1999 contest database [116] • More data will be provided by the PI's collaborators	• IT staff members at GSU's IS & T division • Computer Science students at GSU	

9.2.1 How Do User Studies Help Our Designs

The discussed design methodology has a number of unique characteristics:

- The creation of annotated visualization dictionaries is an attempt to organize our knowledge about the effectiveness of visualization – which are currently scattered in a wide variety of psychological theories, heuristic rules, and empirical studies – into an organized and domain specific framework. Combined with the visualization engine and design gallery interface, they provide much needed theoretical and empirical guidance to visualization construction and evaluation.

- The visualization dictionaries also help separate the visualization design knowledge from visualization programs. The visualization dictionaries are meant to be shared and collaboratively edited by both the research and user community. Users can "plug in" personalized visualization dictionaries to customize a visualization tool.
- The discussed methodology promotes a task-centered visualization design that is different from the currently predominant data-centered design methodology. Tasks are explicitly identified and organized. Visualizations are constructed for specific tasks and their related data parameters.
- The discussed methodology gives domain experts greater control over the visualization construction, exploring the benefits of self-constructed visualization.

9.3 Visualization Dictionary

Current database lacks of sufficient data on visual design documentations; in another word, the visualization dictionary is still bare. Visualization dictionary can only be built based on the real designs and real user inputs. The more input we get, the more comprehensive our visualization dictionary will be.

A typical visualization dictionary is shown in the table below.

Table 15: Structure of the visualization efficiency dictionary.

Tasks	Visual Mappings	Efficiency score (complete by evaluator)	Popularity (the number of times it has been used)
	((visual unit, visual attribute), data attribute)		
	(visual structure, data structure)		
	(visual frame, data group)		

The efficiency scores mainly come from two sources: recorded task completion time and visualization complexity analysis (see table 1). In the future, the eye movement analysis and learning curve may also be considered.

The complexity analysis is carried out in the steps discussed under the Principle of Efficiency. The efficiency scores for the ((visual unit, visual attribute), data attribute, task) mappings are assessed based on perceptual efficiency theories [4, 22, 117-122]. The complexity scores for the (visual structure, data structure, task) mappings will be based on cognitive level theories [3, 105-107, 123-126]. Because of the hierarchical structure of the visualization and data classification, a complexity score can be calculated for the (visual frame, data group, task) mapping based on the perception and cognition level scores. The outcome of the complexity analysis will be a list of complexity scores for different factors that are organized in a hierarchical form.

The complexity analyses will be performed independently by several developers, and the final complexity scores are calculated by averaging the scores assigned by different people. Again, the

PI plans to create a Wiki-style online efficiency dictionary for neural circuitry and computer security visualization.

9.4 Integration Tree

As we mentioned in the previous chapter, current integration tree is still a pre-built independent evaluation tool. Ultimately, the integration tree should be modified as a quicker evaluation tool comparable to user studies.

It should be made as an interactive tool with the visualization designs. Such as, when the changes of the visualization designs have been detected, integration tree should be able to automatically recalculate the integration scores and compare the score with data size, therefore re-adjust the design efficiency by sending warning messages to the designers.

Integration tree is currently built based on theoretical values; user studies can also help to improve the accuracy of integration tree.

9.5 Long Term User Study

Rules involved in the current design are rather limited, and largely based on previously established psychological cognitive studies. The rule basis lacks of human factors, especially long term user studies. In-depth Long-term Case studies (MILCs) [28] suits our purposes perfectly. Instead of a user study alone, we also would like to see how user study may have impact on the designs and building of the design dictionary.

In-depth Long-term Case studies (MILCs) [28] contains multiple detailed steps in performing long term user studies. Our user studies are based on MILCs, but still differ from MILCs in several ways. The detailed steps are listed below:

Two sets of users may participate in this case: designers and evaluators. A few steps may be involved:

1. Identify designers and evaluators. Designers are the group of users who will be using our system to build their own task tree and desired visualization associated with their specific tasks. Evaluators are a group of people who will be evaluate the designs build by designers. Evaluators' input will be recorded, and their responses will be stored in our database for further analysis. Both groups' responses are critical in building our systems. Designers' input serves as our guidelines on relationships between visual units and visual frames vs. visualization design; while evaluators' input can help us on achieving higher universal usability of our system.

2. Record designers' problem, tasks and designs. Usability of a visualization tool is been measured Usability of information visualization tools can be measured in a laboratory however, to be convincing, utility needs to be demonstrated in a real setting. Designers' creations help us to configure the best matches between tasks and visual design. The system should not only be able to help the designers to create proper designs, but also should be able to assist the user in achieving better solutions to their problem. The recorded information can be saved into our database in a table format; the system can use the saved data to recommend proper fit for designer's tasks and problems.

3. Use visual mapping complexity tree. Visual mapping complexity tree was discussed in the previous chapter of this thesis; it requires the user to rate the system based on five different criteria on everything from visual unit to workspace. The visual mapping complexity of the design helps to monitor the design before any major user study can take place. Visual mapping complexity tree can also be used as survey for the evaluators. Evaluators may use visual mapping complexity tree to rate the system or the visual designs built by individual designers. The complexity tree will provide quantitative ratings for all the designs.

4. Open up user network. Current system is stand-alone and not available through internet access. Although depending on the designer's profession and social network, there could be numerous alternatives, we believe that deploying the visualization online and gather evolution feedback is the best approach. IBM many-eyes [50] deployed on-line a few years back, and the feedback received was enormous. By deploying our system online, the user group will be enlarged and the database could grow exponentially. The initial intention for such approach is to deploy all designers' design on web, where rating tools as well as feedback forms will be provided. Evaluators or anybody who are interested may submit their feedbacks voluntarily. The results will again be saved.

5. User studies vs. design process. Traditional user studies are done after a system or a design is complete. In our case, the data comes from user studies is the critical component in our database in order to prettify all the designs. We therefore believe that user studies and design

process should be done concurrently. Concurrent user studies will minimize the design error, and also help to gain more connections between designers and evaluators.

6. Comparisons between our system and comparable systems. While prettifying our system is clearly the top priority, encouraging the user to continue using the best possible tool toward their task is still essential. Doing so would avoid a situation where users try to please the researcher by using the new tool while another classic one would have been more appropriate [28]. At the same time, conducting user interviews on "why choose the alternative" would further help us on the design.

7. Document success and failures. Feedbacks and ratings from designers and evaluators are important in this stage. Summarized reports from both designers and evaluators based on their experiences can be stored into the database. Typical ingredients in the reports include: goals, tasks, visual frames chosen, visual units utilized, etc.

9.6 Addressing Universal Usability

Making visualization tools accessible to diverse users regardless of their backgrounds, technical disadvantages, or personal disabilities is necessary when the tools are to be used by the public, but it remains a challenge for designers [127]. Currently, our system targets mainly on network security issues, while in reality the users may come from all background with variant abilities and requirements.

One of the biggest challenges in visualization designs is to address the visually-impaired users [128]. The current design does not deal with such concerns. It comes to our recognition that people come in different ability and technology advances, accommodate their special needs is our next biggest improvement.

REFERENCES

[1] R. Clark, F. Nguyen, and J. Sweller, *Efficiency in Learning: Evidence-Based Guidelines to Manage Cognitive Load* Pfeiffer, 2006.

[2] W. R. Garner, *The processing of information and structure*: Lawrence Erlbaum, 1974.

[3] J. G. Trafton, S. S. Kirschenbaum, T. L. Tsui, R. T. Miyamoto, J. A. Ballas, and P. D. Raymond, "Turning pictures into numbers: extracting and generating information from complex visualizations," *International Journal of Human-Computer Studies,* vol. 53, pp. 827-850, 2000.

[4] M. Wertheimer and D. King, *Max Wertheimer and Gestalt Theory*: Transaction Publishers, 2004.

[5] J. M. Wolfe and T. S. Horowitz, "What attributes guide the deployment of visual attention and how do they do it?," *Nature Reviews/Neuroscience,* vol. 5, pp. 1-7, 2004.

[6] J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Transactions on Graphics,* vol. 5, pp. 110-141, 1986.

[7] W. S. Cleveland and R. McGill, "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods," *Journal of the American Statistical Association,* vol. 79, pp. 531-554, 1984.

[8] W. S. Cleveland and R. McGill, "Graphical Perception and Graphical Methods for Analyzing Scientific Data," *Science,* vol. 229, pp. 828-833, 1985.

[9] M. Dastani, "The Role of Visual Perception in DataVisualization," *Journal of Visual Languages and Computing,* vol. 13, pp. 601-622, 2002.

[10] M. Wattenberg and D. Fisher, "Analyzing perceptual organization in information graphics," *Information Visualization,* vol. 3, pp. 123-133, 2004.

[11] R. Cox, "Representation construction, externalised cognition and individual differences," *Learning and Instruction,* vol. 9, pp. 343-363, 1999.

[12] E. G. Freedman and P. Shah, "Toward a Model of Knowledge-Based Graph Comprehension," in *Second International Conference on Diagrammatic Representation and Inference; Lecture Notes in Computer Science*. vol. 2317, 2002, pp. 59-141.

[13] D. Brumley, L.-H. Liu, P. Poosankam, and D. Song, "Design space and analysis of worm defense strategies," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006.

[14] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review,* vol. 34, pp. 39-53, 2004.

[15] S. Card, J. Mackinlay, and B. Shneiderman, "Information Visualization," in *Readings in Information Visualization using Vision to think*, S. Card, J. Mackinlay, and B. Shneiderman, Eds., 1999.

[16] A. Komlodi, P. Rheingans, U. Ayachit, J. R. Goodall, and A. Joshi, "A User-centered Look at Glyph-based Security Visualization," in *Workshop on Visualization for Computer Security*, Minneapolis, MN, USA, 2005pp. 21 - 28.

[17] D. A. Keim, "Information Visualization and Visual Data Mining," *IEEE Transactions on Visualization and Computer Graphics,* vol. 8, pp. 1-8, 2002.

[18] D. Bowman, "The Science of Interaction Design," in *ACM SIGGRAPH Course Notes*, 2000.

[19] T. Zuk, L. Schlesier, P. Neumann, M. S. Hancock, and S. Carpendale, "Heuristics for Information Visualization Evaluation," in *BELIV* Venice, Italy, 2006.

[20] T. Zuk, M. S. Hancock, and S. Carpendale, "Theoretical Analysis of Uncertainty Visualizations," in *SPIE & IS&T Conf. Electronic Imaging* 2006.

[21] J. Bertin, *Semiology of Graphics*: University of Wisconsin Press, 1983.

[22] C. Ware, *Information Visualization: Perception for Design*, 2nd ed.: Morgan Kaufmann, 2004.

[23] B. Tognazzini, "First Principles of Interaction Design http://www.asktog.com/basics/firstPrinciples.html." vol. 2006, 2006.

[24] C. G. Healey, "On the Use of Perceptual Cues and Data Mining for Effective Visualization of Scientific Datasets," in *GI*, 1998, pp. 177-184.

[25] E. R. Tufte, *The Visual Display of Quantitative Information* Cheshire, CT: Graphics Press, 2001.

[26] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in *Symposium on Visual Languages*, 1996, pp. 336-343.

[27] R. Amar and J. Stasko, "A Knowledge Task-Based Framework for Design and Evaluation of Information Visualizations," in *InfoVis*, Los Alamitos, USA, 2004, pp. 143-149.

[28] B. Shneiderman and C. Plaisant, "Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies," in *AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, Venice, Italy 2006.

[29] M. Tory and T. Möller, "Evaluating visualizations: do expert reviews work?," *Computer Graphics and Applications,* vol. 25, pp. 8-11, 2005.

[30] R. Brath, "Metrics for effective information visualization," in *Symposium on Information Visualization*, AZ, USA, 1997, pp. 108-111.

[31] B. Craft and P. Cairns, "Beyond guidelines: what can we learn from the visual information seeking mantra?," in *Proceedings of the 9th IEEE International Conference on Information Visualization (IV)*, 2005.

[32] E. Morse, M. Lewis, and K. A. Olsen, "Evaluating visualizations: using a taxonomic guide," *International Journal of Human-Computer Studies,* vol. 53, pp. 637-662, 2000.

[33] M. Tory and T. Moller, "Evaluating Visualizations: Do Expert Reviews Work?," *IEEE Computer Graphics and Applications,* vol. 25, pp. 8-11, 2005.

[34] T. Zuk, L. Schlesier, P. Neumann, M. S. Hancock, and S. Carpendale, "Heuristics for Information Visualization Evaluation," in *Proceedings of the Working Conference on Advanced User Interface (AVI)*, 2006.

[35] M. Scaife and Y. Rogers, "External cognition : how do graphical representations work?," *International Journal of Human-Computer Studies,* vol. 45, pp. 185-213, 1996.

[36] I. Vessey, "Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature," *Decision Sciences,* vol. 22, p. 219, 1991.

[37] T. M. Shaft and I. Vessey, "The Role of Cognitive Fit in the Relationship between Software Comprehension and Modification," *MIS Quarterly,* vol. 30, 2006.

[38] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi, "Preserving the Big Picture: Visual Network Traffic Analysis with TNV," in *Workshop on Visualization for Computer Security*, Minneapolis, MN, USA, 2005, pp. 47 - 54

[39] K. Abdullah, C. Lee, G. Conti, J. A. Copeland, and J. Stasko, "IDS RainStorm: Visualizing IDS Alarms," in *IEEE Symposium on Information Visualization's Workshop on Visualization for Computer Security (VizSEC)*, 2005.

[40] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "PortVis: a tool for port-based detection of security events," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* Washington DC, USA: ACM Press, 2004.

[41] G. Conti, *Security Data Visualization: Graphical Techniques for Network Analysis*: No Starch Press, 2007.

[42] F. B. Viegas, M. Wattenberg, F. v. Ham, J. Kriss, and M. McKeon, "Many Eyes: A Site for Visualization at Internet Scale," in *Proceedings of the IEEE Symposium on Information Visualization*, 2007.

[43] G. Conti, "http://www.rumint.org/."

[44] S. Lok and S. Feiner, "A Survey of Automated Layout Techniques for Information," in *SmartGraphics*, 2001.

[45] S. K. Feiner and K. R. McKeown, "Automating the Generation of Coordinated Multimedia Explanations," *Computer,* vol. 24, pp. 33 - 41, 1991.

[46] C. Beshers and S. Feiner, " AutoVisual: rule-based design of interactive multivariatevisualizations," *Computer Graphics and Applications, IEEE,* vol. 13, pp. 41-49, 1993.

[47] A. Komlodi, J. R. Goodall, and W. G. Lutters, "An Information Visualization Framework for Intrusion Detection," in *CHI*, Vienna, Austria, 2004.

[48] F. B. Viégas, M. Wattenberg, F. v. Ham, J. Kriss, and M. McKeon, "Many Eyes: A Site for Visualization at Internet Scale," *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS,,* vol. 13, 2007.

[49] C. M. Danis, F. B. Viegas, M. Wattenberg, and J. Kriss, "Your Place or Mine? Visualization as a Community Component," in *CHI*, Florence, Italy, 2008.

[50] IBM, "http://manyeyes.alphaworks.ibm.com/manyeyes/."

[51] S. M. Casner, "A Task-Analytic Approach to the Automated Design of Graphic Presentations," *ACM Transactions on Graphics,* vol. 10, pp. 111-151, April, 1991 1991.

[52] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases " *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS,,* vol. 8, pp. 52-65, 17 Apr. 2001 2001.

[53] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2005.

[54] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* Washington DC, USA: ACM Press, 2004.

[55] C. Muelder, K.-L. Ma, and T. Bartoletti, "A visualization methodology for characterization of network scans " in *Visualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on* Minneapolis, MN, USA 2005, pp. 29 - 38.

[56] H. Koike, K. Ohno, and K. Koizumi, "Visualizing cyber attacks using IP matrix," in *Proceedings of the Workshop on Visualization for Computer Security (VisSEC)*: IEEE, 2005.

[57] A. Komlodi, P. Rheingans, U. Ayachit, J. R. Goodall, and A. Joshi, "A User-centered Look at Glyph-based Security Visualization," in *Proceedings of the Workshop on Visualization for Computer Security*: IEEE, 2005.

[58] A. D'Amico and M. Kocka, "Information assurance visualizations for specific stages of situational awareness and intended uses: lessons learned," in *Workshop on Visualization for Computer Security*, Minneapolis, MN, USA, 2005, pp. 107 - 112

[59] A. Komlodi, J. Goodall, and W. Lutters, "An information visualization framework for intrusion detection," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (ACM CHI)*, 2004.

[60] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A System for Query, Analysis and Visualization of Multi-dimensional Relational Databases," *IEEE Transactions on Visualization and Computer Graphics,* vol. 8, 2002.

[61] C. North and B. Shneiderman, "Snap-together visualization: can users construct and operate coordinated visualizations?," *International Journal of Human-Computer Studies,* vol. 53, pp. 715-739, 2000.

[62] C. North and B. Shneiderman, "Snap-Together Visualization: A User Interface for Coordinating Visualization via Relational Schemata," in *Proceedings of the working conference on Advanced visual interfaces*: ACM Press, 2000.

[63] J. C. Roberts, "On Encouraging Multiple Views for Visualization," in *Proceedings of the International Conference on Information Visualization (IV)*: IEEE, 1998.

[64] J. C. Roberts, "On Encouraging Coupled Views for Visualization Exploration," in *Proceedings of SPIE Conference on Visual Data Exploration and Analysis*: IS&T and SPIE, 1999.

[65] G. Fink, P. Muessig, and C. North, "Visual Correlation of Host Processes and Network Traffic," in *IEEE Visualization 2005, Workshop on Visualization for Computer Security (VizSEC 2005)*, 2005, p. 8.

[66] G. Fink and C. North, "Root Polar Layout of Internet Address Data for Security Administration," in *IEEE Visualization 2005, Workshop on Visualization for Computer Security (VizSEC 2005)*, 2005, p. 8.

[67] X. Yin, W. Yurcik, and A. Slagell, "VisFlowConnect-IP: An Animated Link Analysis Tool For Visualizing Netflows," in *FLOCON - Network Flow Analysis Workshop (Network Flow Analysis for Security Situational Awareness)* Baltimore, MD, 2005.

[68] D. Yao, M. Shin, R. Tamassia, and W. H. Winsborough, "Visualization of Automated Trust Negotiation," in *Workshop on Visualization for Computer Security*, Minneapolis, MN, USA, 2005, pp. 65 - 74

[69] Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher, and S. Foresti, "A Visualization Paradigm for Network Intrusion Detection," in *IEEE Information Asssurance Workshop*, West Point, NY, 2005, pp. 92-99.

[70] S. T. Teoh, T. Jankun-Kelly, K.-L. Ma, and F. Wu, "Visual Data Analysis for Detecting Flaws and Intruders in Computer Network Systems " *IEEE Computer Graphics and Applications,* vol. September/October 2004.

[71] H. Chernoff, "Using faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association,* vol. 68, pp. 361-368, 1973.

[72] M. X. Zhou and S. K. Feiner, "Visual Task Characterization for Automated Visual Discourse Synthesis," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 1998.

[73] M. X. Zhou and S. K. Feiner, "Top-Down Hierarchical Planning of Coherent Visual Discourse," in *Proceeding of International Conference on Intelligent User Interface (IUI)*: ACM, 1997.

[74] J. Lohse, H. Rueter, K. Biolsi, and N. Walker, "Classifying Visual Knowledge Representations: A Foundation for Visualization Research," in *Proceedings of IEEE Visualization Conference (VIS)*, 1990.

[75] S. F. Roth and J. Mattis, "Data Characterization for Intelligent Graphics Presentation," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*: ACM, 1990.

[76] S. Wehrend and C. Lewis, "A Problem-oriented Classification of Visualization Techniques," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*: IEEE, 1990.

[77] G. L. Lohse, K. Biolsi, N. Walker, and H. H. Rueter, "A Classification of Visual Representations," *Communications of the ACM,* vol. 37, pp. 36-49, 1995.

[78] M. C. Chuah and S. F. Roth, "On the Semantics of Interactive Visualizations," in *Proceeding of IEEE Symposium on Information Visualization (InfoVis)*: IEEE, 1996.

[79] C. North and B. Schneiderman, "A Taxonomy of Multiple Window Coordinations," Technical Report CS-TR-3854, Department of Computer Science, University of Maryland1998.

[80] J. C. Roberts, "Display Models for Visualization," in *Proceedings of IEEE Conference on Information Visualization (IV)*, 1999.

[81] D. A. Keim, "Visual Exploration of Large Data Sets," *Communications of the ACM,* vol. 44, pp. 39-44, 2001.

[82] M. Tory and T. Möller, "Rethinking Visualization: A High-Level Taxonomy," in *Proceeding of the IEEE Symposium on Information Visualization (InfoVis)*, 2004.

[83] D. Tang, C. Stolte, and R. Bosch, "Design choices when architecting visualizations," *Information Visualization,* vol. 3, pp. 65-79, 2004.

[84] R. Amar, J. Eagan, and J. Stasko, "Low-Level Components of Analytic Activity in Information Visualization," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*: IEEE, 2005.

[85] R. Amar and J. Stasko, "A Knowledge Task-Based Framework for Design and Evaluation of Information Visualizations," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*: IEEE, 2004.

[86] R. A. Amar and J. T. Stasko, "Knowledge Precepts for Design and Evaluation of Information Visualizations," *IEEE Transactions on Visualization and Computer Graphics,* vol. 11, pp. 432-442, 2005.

[87] E. Valiati, M. Pimenta, and C. M. D. S. Freitas, "A Taxonomy of Tasks for Guiding the Evaluation of Multidimensional Visualizations," in *Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV06), A workshop of the AVI 2006 International Working Conference* Venezia, Italy: ACM, 2006.

[88] G. Klein, R. Pliske, B. Crandall, and D. D. Woods, "Problem detection," *Cognition, Technology & Work,* vol. 7, pp. 14-28, 2005.

[89] S. M. Kosslyn, "Graphics and Human Information Processing: A Review of Five Books," *Journal of the American Statistical Association,* vol. 80, pp. 499-512, 1985.

[90] J. H. Larkin and H. A. Simon, "Why A Diagram is (Sometimes) Worth Ten Thousand Words," *Cognitive Science,* vol. 11, pp. 65-99, 1987.

[91] S. Kosslyn, "Graphics and Human Information," *Journal of the American Statistical Association,* vol. 80, p. 13, September 1985.

[92] M. Petre and T. R. G. Green, "Learning to Read Graphics: Some Evidence that 'Seeing' an Information Display is an Acquired Skill," *Journal of Visual Languages and Computing,* vol. 4, pp. 55-70, 1993.

[93] R. Cox and P. Brna, "Supporting the use of external representation in problem solving: the need for flexible learning environments," *Journal of Artificial Intelligence in Education,* vol. 6, pp. 239-302, 1995.

[94] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, and T. Kang, "Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation," in *Proceedings of ACM SIGGRAPH Conference*, 1997.

[95] M. Terry, "Set-Based User Interface," in *PhD Thesis, School of Computing, Georgia Institute of Technology, Atlanta, Georgia*, 2005.

[96] I. Bratko, *PROLOG Programming for Artificial Intelligence*, 2 ed.: Addison-Wesley Longman Publishing Co., Inc., 1990.

[97] H. Pain and A. Bundy, "What stories should we tell novice PROLOG programmers?," in *Artificial intelligence programming environments* New York, NY: John Wiley & Sons, 1987, pp. 119-130.

[98] R. Simmons and D. Apfelbaum, "A Task Description Language for Robot Control," in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems* Victoria, B.C., Canada, 1998.

[99] G. L. Lohse, "The role of working memory on graphical information processing," *Behaviour & Information Technology,* vol. 16, pp. 297-308, 1997.

[100] N. Marcus, M. Cooper, and J. Sweller, "Understanding Instructions," *Journal of Educational Psychology,* vol. 88, pp. 49-63, 1996.

[101] J. Sweller, "Visualisation and Instructional Design," in *Proceedigns of the International Workshop on Dynamic Visualizations and Learning*, 2002.

[102] S. Casner and J. Bonar, "Using the expert's diagram as a specification of expertise," in *Proceedings of IEEE Symposium on Visual Languages*, 1988.

[103] J. Davies and A. K. Goel, "Transfer of problem-solving strategy using Covlan," *Journal of Visual Languages and Computing,* vol. 18, pp. 149-164, 2007.

[104] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in *Proceedings of the IEEE Conference on Visual Languages*: IEEE, 1996.

[105] R. M. Ratwani and J. G. Trafton, "Making Graphical Inferences: A Hierarchical Framework," in *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*, 2004.

[106] R. M. Ratwani, J. G. Trafton, and D. A. Boehm-Davis, "Thinking Graphically: Extracting Local and Global Information," in *Proceedings of the Annual Meeting of Cognitive Science Society*, 2003.

[107] J. G. Trafton and S. B. Trickett, "A New Model of Graph and Visualization Usage," in *Proceedings of the Annual Meeting of Cognitive Science Society*, 2001.

[108] A. Paivio, *Mental representations: a dual coding approach*: Oxford University Press, 1986.

[109] J. M. Clark and A. Paivio, "Dual Coding Theory and Education," *Educational Psychology Review,* vol. 3, pp. 149-210, 1991.

[110] B. Johnson, "TreeViz: treemap visualization of hierarchically structured information," in *Conference on Human Factors in Computing Systems*, Monterey, California, United States 1992.

[111] R. Bearavolu, K. Lakkaraju, and W. Yurcik, "NVisionIP: An Animated State Analysis Tool for Visualizing NetFlows " in *FLOCON - Network Flow Analysis Workshop (Network Flow Analysis for Security Situational Awareness)*, 2005.

[112] G. Conti, J. Grizzard, M. Ahamad, and H. Owen, "Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries," in *IEEE Symposium on Information Visualization's Workshop on Visualization for Computer Security (VizSEC)*, 2005.

[113] X. Suo, Y. Zhu, and G. S. Owen, "Measuring the Complexity of Visualization Design," in *Proceedings of the 2007 Workshop on Visualization for Computer Security (VizSEC)*, 2007.

[114] P. Katz, S. Prasad, R. Sunderraman, and Y. Zhu, "NeuronBank," http://www.neuronbank.org, 2007.

[115] "DARPA Intrusion Detection Evaluation," MIT Lincoln Laboratory, http://www.ll.mit.edu/IST/ideval/, 2001.

[116] "ACM SIGKDD: KDD Cup Contest," http://www.acm.org/sigs/sigkdd/kddcup/index.php?section=1999&method=info, 1999.

[117] S. M. Casner and J. H. Larkin, "Cognitive Efficiency Considerations for Good Graphic Design," in *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* Ann Arbor, MI, 1989.

[118] C. G. Healey, "Building a Perceptual Visualization Architecture," *Behavior & Information Technology,* vol. 19, pp. 349-366, 2000.

[119] S. M. Kosslyn, "Understanding Charts and Graphs," *Applied Cognitive Psychology* vol. 3, pp. 185-226, 1989.

[120] G. L. Lohse, "A Cognitive Model for Understanding Graphical Perception," *Human-Computer Interaction,* vol. 8, pp. 353-388, 1993.

[121] J. M. Wolfe, "Visual Search," in *Attention*, H. Pashler, Ed.: University College London Press, 1998.

[122] J. M. Wolfe, "How might the rules that govern visual search constrain the design of visual displays?," *Society for Information Display (SID) Symposium Digest of Technical Papers,* vol. 36, pp. 1395-1397, 2005.

[123] G. S. Halford, W. H. Wilson, and S. Phillips, "Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology," *Behavioral and Brain Sciences,* vol. 21, pp. 803-865, 1998.

[124] M. Hegarty, "Capacity Limits in Diagrammatic Reasoning," in *Proceedings of the First International Conference on Theory and Application of Diagrams (Diagrams), Lecture Notes in Artificial Intelligence*: Springer, 2000.

[125] S. B. Trickett, R. M. Ratwani, and J. G. Trafton, "Real-World Graph Comprehension: High-Level Questions, Complex Graphs, and Spatial Cognition," (under review)2006.

[126] J. R. Anderson, M. Matessa, and C. Lebiere, "ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention," *Human-Computer Interaction,* vol. 12, pp. 439-462, 1997.

[127] C. Plaisant, "The challenge of information visualization evaluation," in *the working conference on Advanced visual interfaces*, Gallipoli, Italy 2004.

[128] J. Dykes, A. Maceachren, and M.-J. Kraak, *Exploring Geovisualization*: Elsevier, 2005.