

ABSTRACT

Title of Thesis: **TEMPORAL TREEMAPS FOR VISUALIZING
TIME SERIES DATA**

Gouthami Chintalapani, Master of Science, 2004

Thesis Directed By: **Professor Ben Shneiderman
Department of Computer Science**

**Dr. Catherine Plaisant
Institute for Advanced Computer Studies**

Treemap is an interactive graphical technique for visualizing large hierarchical information spaces using nested rectangles in a space-filling manner. The size and color of the rectangles show data attributes and enable users to spot trends, patterns or exceptions.

Current implementations of treemaps help explore time-invariant data. However, many real-world applications require monitoring hierarchical, time-variant data. This thesis extends treemaps to interactively explore time series data by mapping temporal changes to color attribute of treemaps. Specific contributions of this thesis include:

- Temporal treemaps for exploring time series data through visualizing absolute or relative changes, animating them over time, filtering data items, and discovering trends using time series graphs.

- The design and implementation of extensible software modules based on systems engineering methodologies and object-oriented approach.
- Validation through five case studies: health statistics, web logs, production data, birth statistics, and help-desk tickets; future improvements identified from the user feedback.

TEMPORAL TREEMAPS FOR VISUALIZING TIME SERIES DATA

By

Gouthami Chintalapani

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2004

Advisory Committee:

Professor Ben Shneiderman, Chair and Advisor
Dr. Catherine Plaisant, Co-Advisor
Professor Michael O. Ball
Professor Gregory B. Baecher

© Copyright by
Gouthami Chintalapani
2004

DEDICATION

To
My Family

ACKNOWLEDGEMENTS

I very gratefully acknowledge the guidance, support and encouragement of my advisors, Dr. Ben Shneiderman and Dr. Catherine Plaisant. I owe them many thanks for critiquing and helping me refine the work presented in this thesis. I would like to thank Dr. Gregory Baecher, Dr. Michael Ball for agreeing to serve on my committee and review the thesis.

I would like to acknowledge Cheryl Lukehart and Donald Schiro at ChevronTexaco and Michael Davidoff at March of Dimes for their support and constructive feedback. I would like to express my gratitude to ChevronTexaco for the funding support of my research work at the HCIL and I gratefully acknowledge the financial support for my Masters through a Graduate Research Assistantship from the Institute for Advanced Computer Studies.

I would also like to acknowledge the supportive roles of Niem H. Dang, Ani Jain, Nilani Aluthgedara, Aleks Aris and all my colleagues at HCIL, Ms Lee Harper, and Dr Mark Austin, systems engineering graduate program directors, and Institute for Systems Research while conducting my research and graduate studies.

Finally, I am, as always, thankful to my family for their love and support.

TABLE OF CONTENTS

List of Figures	vii
Chapter 1: Introduction	1
Chapter 2: Related work	6
2.1 Treemaps History.	6
2.2 Hierarchical Data: Visualizations	8
2.3 Temporal Data: Visualizations.	16
2.4 Summary.....	27
Chapter 3: User Requirements and Design	28
3.1 Overviews	32
3.2 Missing Data	35
3.3 Overlaying Labels at Selected Hierarchy Level	38
3.4 Other Features	39
3.5 Getting started with temporal treemaps.	42
3.5.1 Visualizing numerical attribute changes over time	42
3.5.2 Visualizing categorical attribute changes over time	44
Chapter 4: Implementation	46
4.1 A Tour of the Code	46

4.2 Data Management	48
4.2.1 Input File Format.....	48
4.2.2 Data Structures	49
4.2.3 Loading a Data File	50
4.3 Graphical User Interface	50
4.3.1 Overview of Series Tab	52
4.3.2 Display and Interaction Handling	53
4.3.3 Coupling treemaps with time series graphs	55
4.4 Performance	56
Chapter 5: Case Studies	58
5.1 NCHS Death Statistics	58
5.2 HCIL web logs	63
5.3 Daily Oil Production Data	72
5.4 NCHS Live Births Data	77
5.5 Help desk tickets	81
5.6 Feedback	88
Chapter 6: Conclusions	90
6.1 Research Contributions	90
6.2 Future Work	92
6.2.1 Scaling	92

6.2.2	Generating Textual Summaries	93
6.2.3	Coupling with TimeSearcher	94
6.2.4	Additional Features	94
Appendix A A Sample Time Series Data File		96
Appendix B A Sample List of Settings File		97
Appendix C TTSGenerator		98
References		100

LIST OF FIGURES

1.1	Temporal treemap showing %preterm births in 2001. Each rectangle is a state, grouped by geographical region. A shade of green shows states with preterm birth rate less than 5% and a shade of red indicates states with preterm birth rate greater than 5%.....	3
2.1.1	Visualizing 62 projects, each rectangle represents a project, size of the rectangle is proportional to the budget allocated, color is proportional to budget balance, a shade of red-white represents over spent projects, and a shade of white-green represents under spent projects, projects are grouped by region and then by department	7
2.2.1	Hyperbolic browser showing an organization chart, with root at the center [25]	9
2.2.2	Cone tree visualization of a Unix file directory system [31]	10
2.2.3	Visualization of JDK 1.2 distribution file directory using Information Slice technique, with the whole directory tree on the left hand side and a sub directory zoomed onto another slice on the right hand side [2]	11
2.2.4	Magic Eye view visualization technique showing a derived tree of ontology with 1100 nodes [24].....	12
2.2.5	Cheops display showing the www information with 9 hierarchy levels (grayed out), blue-highlighted nodes indicate the sub category “Health/Medicine” (fifth child in the Natural sciences branch) in the “Natural Sciences” (third child from the root) category in the whole tree [3].....	13

2.2.6	Table lens visualization of Olympic diving medal results, columns sorted first by Gender, then by Events etc. which in a way shows the hierarchical structure formed by Gender as the first level, Events as the second level etc. [30]	14
2.2.7	Infozoom visualization of two animal classifications (Infovis 2003, Contest data) as table, highlighting the differences between two classification trees in black [37].....	15
2.2.8	A Mosaic display showing the eye and hair color statistical data of 696 subjects, with hair color mapped to columns and eye color mapped to rows and the color shade showing the standard residuals from the statistical model [15]	16
2.3.1	Pixel based recursive pattern visualization showing the DOWJONES, GOLD, IBM, and DOLLAR stock prices for 7 years and 12 months. Each vertical bar represents a year, subdivided into 12 months. Color shows the stock price, light color shows high stock prices and dark color shows low stock prices. Users map the color changes to those of stock prices, for example IBM stock price has fell sharply after the first one and half month [27].....	17
2.3.2	A spiral display showing the monthly consumption for Baphia Capparidifolia by chimpanzees during 1980-1988. Each lap corresponds to a year and spokes correspond to the months in the year. Area of the blot shows the actual consumption percentage. Users follow the spiral laps or spokes and observe the areas of blots for tracking consumption trends, for example the highest percentage consumption occurred in a particular month for all the years [9]	18
2.3.3	Time tubes and disk trees showing the WWW web ecology and page visitation counts for four weeks [11].....	20

2.3.4	Coordinated visualizations for context: Visualizing the WWW log data for HCIL website (www.cs.umd.edu/hcil) for 2 months [18].....	21
2.3.5	TimeSearcher showing the stock data for 12 months [17].....	22
2.3.6	SmartMoney.com’s Map of the Market showing the % change in stock prices on Feb 10, 2004 since the past 26 weeks [35].....	23
2.3.7	Cluster and Calendar based visualization of number of employees and their working hours/patterns in ECN. Patterns are shown as time series graphs with number of employees on y-axis and daily working hours on x-axis. Clusters are shown on the calendars. Color shows the corresponding clusters and patterns. For example, cluster 722 shows that there are fewer people in summer [42]	25
3.1	Temporal treemap application window: treemap window, time series graph window, histogram window, details table and series tab	30
3.2	Timeseries graph overview of oil production data collected from 373 oil wells, showing the graph envelope	33
3.3	Histogram Overview of Oil Production data showing the amount of lost production for 4 days	34
3.4	Monitoring Oil production data, showing the missing data replaced with a value of “-1”, shown on the lower right corner; light blue colored rectangles represents the items (oil wells) with missing data on day_2	36
3.5	Visualizing HCIL web logs grouped by directory structure, without borders and with overlay labels at center of node, showing the labels at hierarchy level 4	38

3.6	Visualizing Oil production data, grouped by Business center and asset team, size is proportional to amount of estimated production of oil, color proportional to amount of lost production, grayed out rectangles show the wells whose lost production on day_2 is less than that on day_1	40
3.7	Visualization of web logs for HCIL, grouped by directory structure. Search for “gif” images, colored rectangles represent gif images, grayed out rectangles are non-search results	41
3.8	User interface for specifying sub string search based filtering	42
4.1	A Schematic overview of classes in temporal treemaps	50
4.3.1	Controls in Series Tab	53
5.1.1	Visualization of 43 causes of death statistics over 18 years grouped by disease category. Each rectangle represents a cause of death with size proportional to number of deaths in 1998 and a shade of white –yellow-purple showing increasing number of deaths in 1981 .	59
5.1.2	Visualization of same 43 causes of deaths statistics in 1998	60
5.1.3	Time series graph overview of death statistics of 43 diseases	61
5.1.4	Visualization of relative changes in death rates in 1998 with respect to 1981, grouped by disease categories. Each rectangle represents a disease; size is proportional to number of deaths in 1998 and color is proportional to % change in number of deaths from 1981 to 1998, with green-white color showing % decrease and white –red showing %increase in death rate	62

5.2.1	Visualization of HCIL web log during week_1, grouped by directory structure. Each rectangle represents a file/web page, size proportional to file size in bytes, and color proportional to hitcount, a color shade of white -yellow- purple represents 0 to 4298 hitcount	64
5.2.2	Visualization of HCIL web logs during week_1, grouped by directory structure. Each rectangle represents a web page and color represents hitcount. White color shows missing pages, yellow – purple gradient shows 0 to 4300 hits	65
5.2.3	Visualizing HCIL web logs during week_2	66
5.2.4	Visualization of HCIL web logs during Week_3	67
5.2.5	Visualization of HCIL web logs in week_4	68
5.2.6	Visualization of HCIL web logs in Week_2 without borders and with overlaid labels	69
5.2.7	Visualization of HCIL web logs in week_4 without borders	70
5.3.1	Monitoring Oil well data on Day_1, grouped by Profit center and then Asset team. Each rectangle represents an oil well, size proportional to amount of oil produced and color proportional to lost production, white represent zero oil lost, shade of yellow to black is proportional to maximum oil lost. Light gray color represents oil wells which did not report data	72
5.3.2	Visualizing same oil well data on Day_2	74
5.3.3	Visualizing same oil well data on Day_3	75
5.3.4	Visualizing same oil well data on Day_4	76

5.4.1	Visualizing percentage of live births that were born preterm in 1990. Each rectangle is a state; grouped by geographical region; layout is slice and dice; colored by %preterm in live births	78
5.4.2	Visualizing %preterm births for each state in 2001	79
5.4.3	Visualizing %changes in live births that were born preterm from 1990 to 2001	80
5.4.4	Visualizing %preterm in live births in 2001. Each rectangle is a state, grouped by geographical region and race. Color shows the %preterm in live births in 2001. A shade of yellow shows %preterm births less than 10% and shade of red shows 10% to 36%	81
5.5.1	Visualizing help desk tickets in December. Each rectangle is a ticket, grouped by Functional Data Area and First App. Colored by Functional Data Area	82
5.5.2	Visualizing help desk tickets in June. Each rectangle is a ticket, grouped by Functional Data Area and First App attributes. Color shows the Functional Data Area	84
5.5.3	Visualizing help desk tickets in July	85
5.5.4	Visualization of help desk tickets, aggregated by number of tickets in each Functional Data Area, and number of tickets as the time series attribute.....	86
5.5.5	Visualizing aggregated number of tickets in each area in December	87
A.1	A sample timeseries (.tts) data file	96
B.1	A sample list of settings (.lst) file	97

Chapter 1

Introduction

Information visualization can be defined as the use of computer-supported interactive visual representation of *abstract* data to amplify cognition [8]. Information visualization is more likely to be used to display database content (e.g. recorded stock values, health statistics) than output of models or simulations. It aims to provide compact graphical presentations and user interfaces for interactively manipulating large numbers of items (10^2 - 10^6), possibly extracted from far larger datasets [40, 36]. Information visualization, sometimes called visual data mining, uses the remarkable human visual system to enable users to make discoveries or decisions, or propose explanations about patterns or exceptions. Perceptual psychologists and graphic designers (e.g. Tufte, 1983) provide guidance about presenting static information, while the growing power of desktop computers allows user-interface designers to propose rich interaction mechanisms to manipulate the data in real time.

Treemap [32] is an interactive information visualization tool for visualizing hierarchical data. It makes effective use of display space and represents hierarchical data by a set of nested rectangles generated by a recursive space-partitioning algorithm. The focus is at leaf level where the actual data items are shown. The data attributes are shown by size and color of the rectangles. Treemap enables users to compare data items as well as sub-trees and helps to identify the patterns or exceptions. It also facilitates users in grouping the data by creating new hierarchies

through dynamically selecting a series of data attributes, modifying the existing hierarchies and saving the hierarchies.

However, many real world applications involve monitoring and analyzing time series data. Time series data can be defined as an ordered sequence of values of a variable or an attribute at equal points in time (daily, weekly, monthly etc). Examples of time series data include demographic analysis to study the birth rates, stock market analysis to spot market trends, sales forecasting to examine sales patterns and predict future trends, health statistics to identify disease trends, work load projections, weather forecasting, etc.

In general, a time-series graph, plotted with time stamps on the x-axis and attribute values on the y-axis, is used to represent this data. There have been many interactive visualization techniques that support display and exploration of time series data [9]. Recently, focus has shifted to developing tools for specifying dynamic queries for identifying time-dependent patterns [17].

Analyzing time series data, however, is not trivial. Many real world phenomena show information at different levels of detail. Analyzing each individual data item's time series as a separate graph or a record may not be sufficient to understand the overall trends in the data. It might be useful to analyze the data at different levels of detail to spot the overall trends and patterns in the data. For example, consider stock market data where attributes like stock price and number of stocks traded constantly change with time. A common trend is to study whether a stock's market value has increased or decreased and which stock has the highest/lowest value. A hierarchical

structure can be imposed on stock market data, as defined by the industry groups or other industry-specific attributes. This hierarchical organization of stock market data helps the users to identify the performance of an industry group as a whole as well as the individual companies in that industry and enables the users to visualize the market shifts in a broader perspective.



Figure 1.1 Temporal treemap showing %preterm births in 2001. Each rectangle is a state, grouped by geographical region. A shade of green shows states with preterm birth rate less than 5% and a shade of red indicates states with preterm birth rate greater than 5%.

Current implementations of treemaps support data that is static or fixed. Making use of the existing treemap technique, temporal treemaps introduce a new time dimension to visualize timeseries data.

Temporal treemaps are developed for visualizing hierarchical datasets where one or more data attribute values change over time. The temporal changes can be monitored by mapping the attribute value changes over time to the color attribute in treemap. Experience has demonstrated that changes to sizes are difficult for users to follow.

Figure 1.1 is an example of temporal treemaps for visualizing percentage of live births that were born preterm in 1990. The data is collected for each state for 12 years (1990 to 2001). Each rectangle in Figure 1.1 represents a state, grouped by region. The color of the rectangle (a shade of green and red) is mapped to the %preterm births in 2001. The time step slider on the right side of Figure 1.1 is used to navigate thru time periods to identify the trends as observed by the color changes in the treemap. It has been observed that District of Columbia showed the highest %preterm birth rate of all the states in US.

Users can view data by navigating through each time period or animate over a range of time periods. Temporal treemaps also provide a graphic overview consisting of timeseries graphs of the data items, tightly coupled with the treemap overview as shown in the lower corner of Figure 1.1.

This thesis discusses the related work, preliminary design issues and implementation details, case studies, research contributions and future possible extensions to the current work.

- Chapter 2 discusses the background work in treemaps and other related interactive techniques for visualizing hierarchical and time series data.
- Chapter 3 describes the user requirements and design issues that led to the implementation of temporal treemaps.
- Chapter 4 illustrates the implementation details of temporal treemaps.
- Chapter 5 presents five case studies and explains how temporal treemaps aid in visualizing time series data. This chapter also discusses the feedback collected from the users.
- Chapter 6 describes the research contributions of this thesis and future work.

A brief introduction of temporal treemaps is available at

<http://www.cs.umd.edu/hcil/treemap/timeseries.html>

Temporal treemaps will be included in Treemap 4.2 and the online documentation provides a detailed explanation of all the supported features and their usage.

Chapter 2

Related work

The main focus of this thesis is to enhance treemaps to visualize time series of data which is hierarchical in nature to monitor temporal changes at a broader level. A brief discussion of interactive techniques that support hierarchical and/or temporal data visualization is presented in this chapter.

2.1 Treemaps History

Treemaps [32] were first developed at the Human-Computer Interaction Laboratory (HCIL) of the University of Maryland during the 1990s. Treemap is a visualization tool that uses 100% of the available display space by mapping the hierarchy onto a rectangular region in a space-filling manner. It allows rapid comparison of the size of nodes or the shape of sub-trees and creates a display of leaf node values based on size and color [21]. Dynamic query filters [1] were added to facilitate the exploration of data. Users can filter out unwanted or uninteresting items from the display.

The basic idea of flexible hierarchy was first implemented in *CatTrees* [23], a class project prototype, which allows interactive manipulation to the hierarchy. *CatTrees* was an enhancement to treemaps, and was built on earlier versions of treemaps. *CatTrees* was limited to categorical data. Later, the idea of *CatTrees* was extended to handle both categorical and numerical data and a new prototype interface for creating and manipulating flexible hierarchies was developed in another class

project [34]. Flexible hierarchy facilitates users to categorize the data, define new meaningful hierarchies by selecting the attributes and adding them to the hierarchy, and save the hierarchies. Numerical data is categorized by defining bins or numeric ranges in the data. The interface was refined and integrated in Treemap 4.0 (www.cs.umd.edu/hcil/treemap).

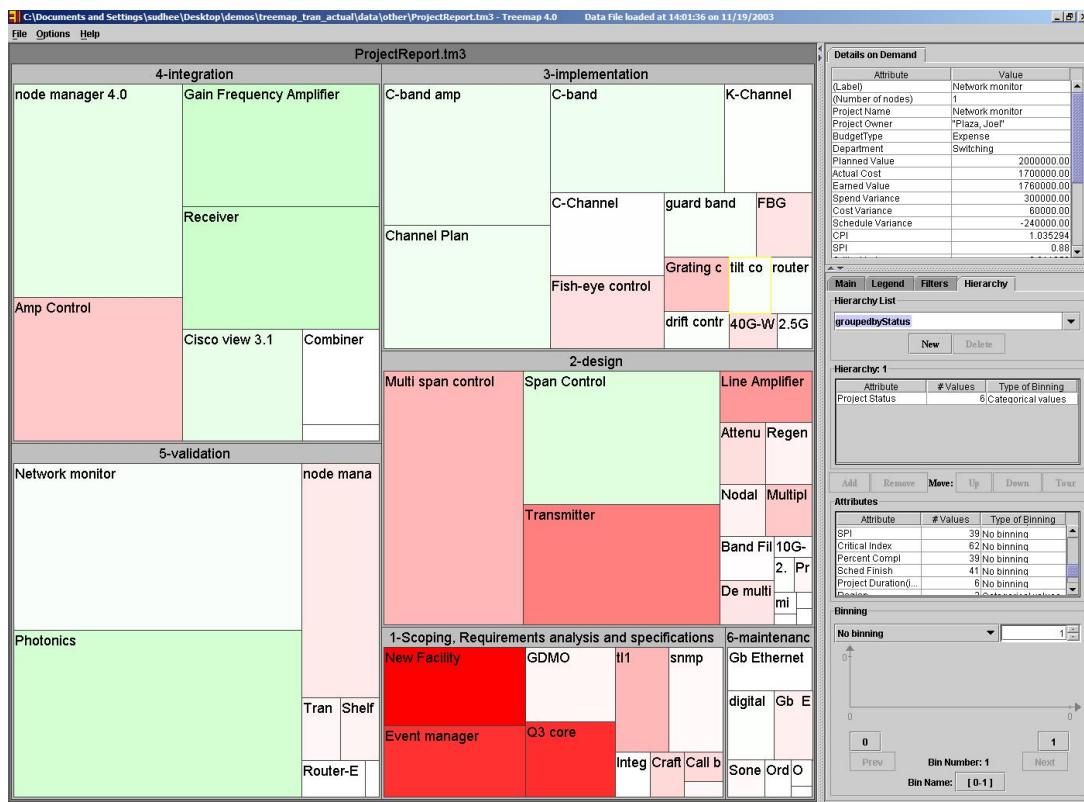


Figure 2.1.1 Visualizing 62 projects, each rectangle represents a project, size of the rectangle is proportional to the budget allocated, color is proportional to budget balance, a shade of red-white represents over spent projects, and a shade of white-green represents under spent projects, projects are grouped by region and then by department.

Figure 2.1.1 shows an abstract visual representation of project portfolio data, which consists of 62 projects, grouped by *project status* attribute. Each rectangle represents a project. The hierarchical structure in Figure 2.1.1 is obtained by imposing a new hierarchy of attribute *project status*. The size of each rectangle is proportional to the *budget allocated* attribute and color is related to the *budget balance* attribute, light colored rectangles represent under-spent projects, dark colored rectangles represent over-spent projects.

Users can identify that the three projects (dark rectangles) have run out of budget in the *scoping and requirements analysis* phase which is the first phase in the life cycle. Users can see that the large projects, *Network Monitor* and *Photonics* (large rectangles) are under-spent (light color).

More recently treemaps are used extensively in wide variety of applications from monitoring stock market data [35] to production management [28]. The historical summary of treemaps and related research can be found at <http://www.cs.umd.edu/hcil/treemap-history>.

2.2 Hierarchical Data: Visualizations

Extensive research is being done in interactive visualization of hierarchical data. Several focus+context techniques have been specified to view large hierarchies in their entirety without losing context such as the *hyperbolic tree browser* [25]. The hierarchy is laid out uniformly on the hyperbolic plane and this plane is mapped onto a circular display region thus effectively using display space. The hyperbolic browser

initially displays a tree with its root at the center, and the focus can be shifted to other nodes of interest using smooth transformations and animation.

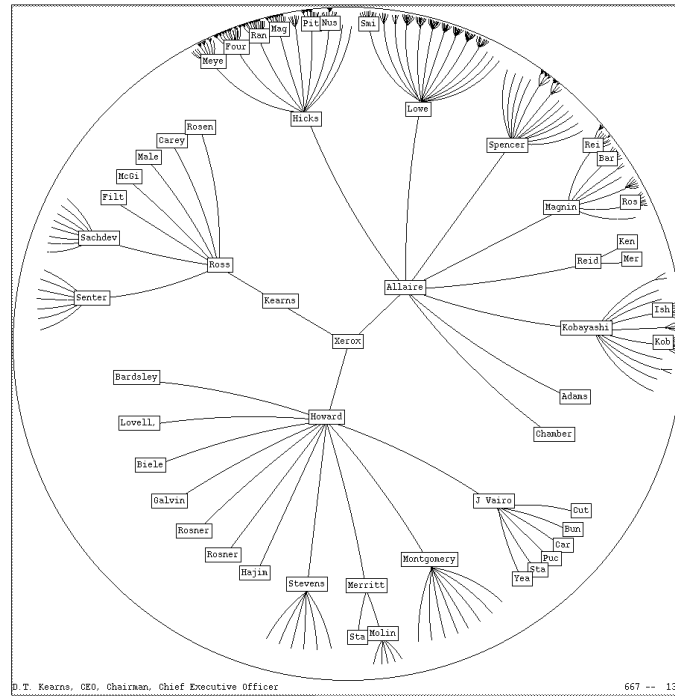


Figure 2.2.1 Hyperbolic browser showing an organization chart, with root at the center [25].

Cone trees [31] present a 3D representation of hierarchical information to maximize effective use of the available screen space and enable visualization of the whole structure. The root of the tree is located at the apex of the cone and all its children are arranged around the circular base of the cone in 3D. Interaction is achieved by rotating the 3D representation to reveal the hidden parts of the tree and zooming.

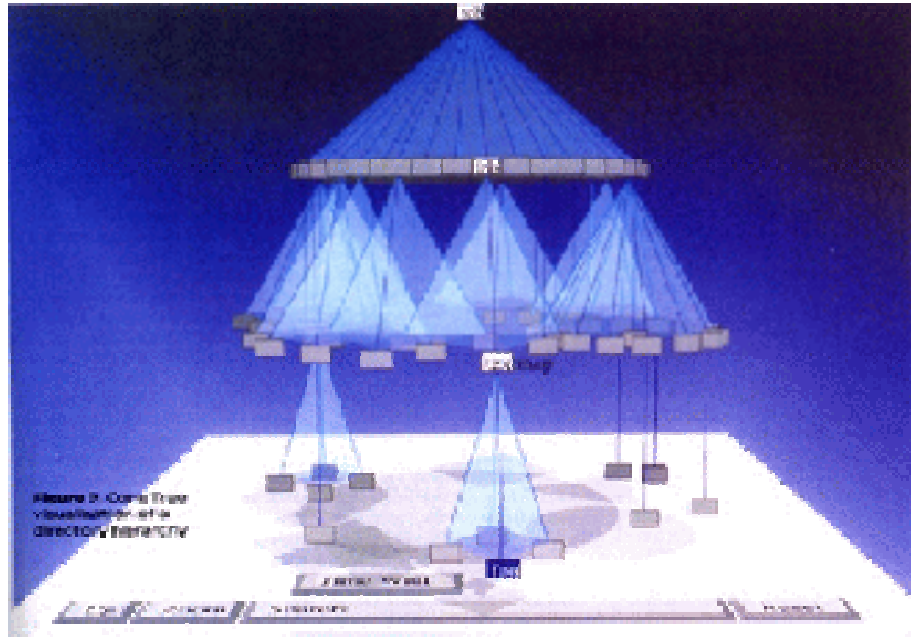


Figure 2.2.2 Cone tree visualization of a Unix file directory system [31].

Reconfigurable disc trees (RDT) [20] is a new visualization technique which can alleviate the disadvantages of cone trees significantly for large hierarchies while maintaining its context of using 3D depth by using disc instead of cones as the basic shape. In RDT, each node is associated with a disc around which its children are placed.

Hierarchical Flip Zooming [6] is another focus+context technique for visualizing hierarchical information sets. It allows for independent focus+context views at each node of the hierarchy and enables parallel exploration of different branches of the hierarchy.

Bubble trees [7] present a tree visualization mechanism based on the natural property of trees to recursively sub-categorize themselves into sub-trees. Each sub-tree is graphically represented as a bubble, which aggregates detail by enclosing

lower level information. Navigation and information retrieval are facilitated through an elegant set of browsing interactions.

Information slices [2] present a new visualization technique for visualizing and manipulating large hierarchies using one or more semi-circular discs. Each disc accommodates 5 to 10 levels of hierarchy with deeper hierarchies shown by a series of cascaded discs. The attribute values are shown at leaf level with leaves framed out in each disc depending on the size of each leaf. Several options are provided to interactively explore the data by zooming on to a part of the tree and expanding the sub tree, controlling the level of detail on each disc etc.

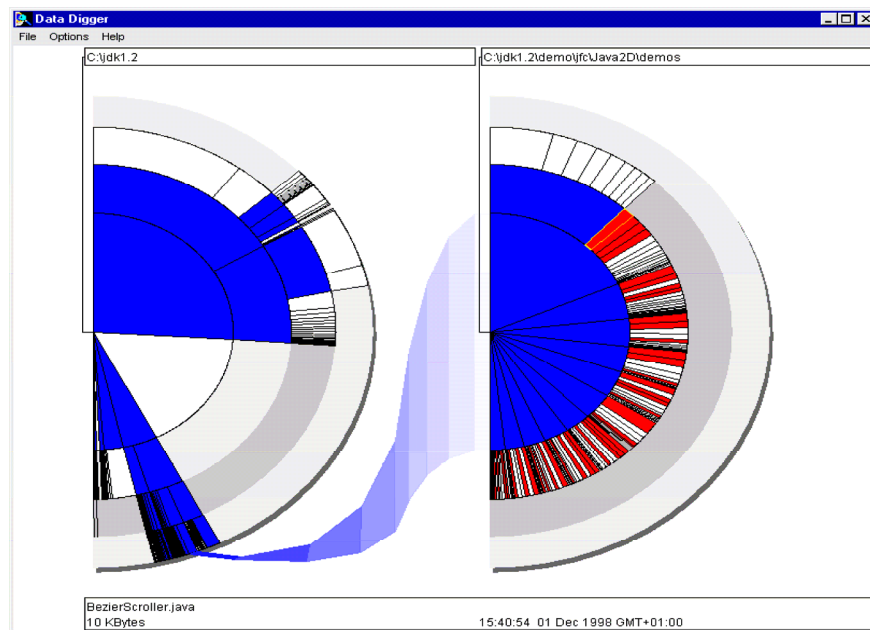


Figure 2.2.3 Visualization of JDK 1.2 distribution file directory using Information Slice technique, with the whole directory tree on the left hand side and a sub directory zoomed onto another slice on the right hand side [2].

The *Magic Eye View* [24] technique for visualizing hierarchies presents another focus+context approach that maps a hierarchy graph onto the surface of a hemisphere, and then applies a projection in order to change the focus area interactively by moving the center of projection.

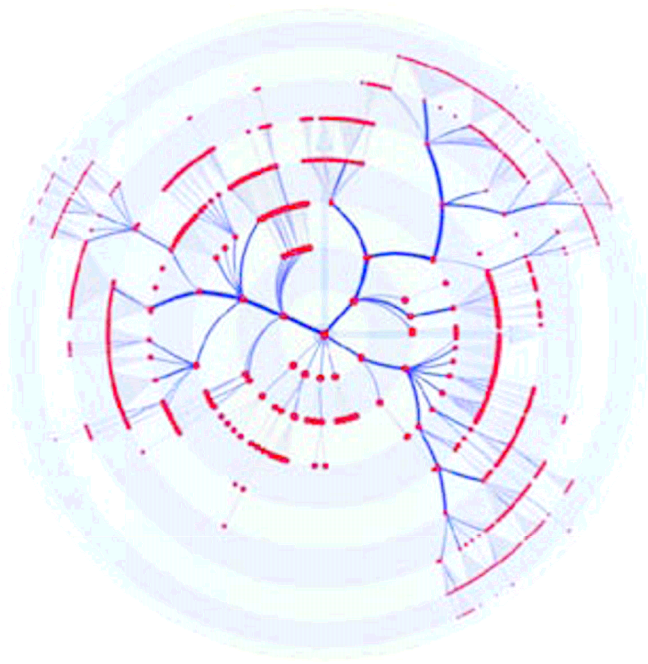


Figure 2.2.4 Magic Eye view visualization technique showing a derived tree of ontology with 1100 nodes [24].

Cheops [3] presents another novel approach to the representation, browsing and exploration of huge, complex information hierarchies. The Cheops method is based on compressed visualization of a hierarchical data set and maintains context within a complex hierarchy while providing easy access to details.

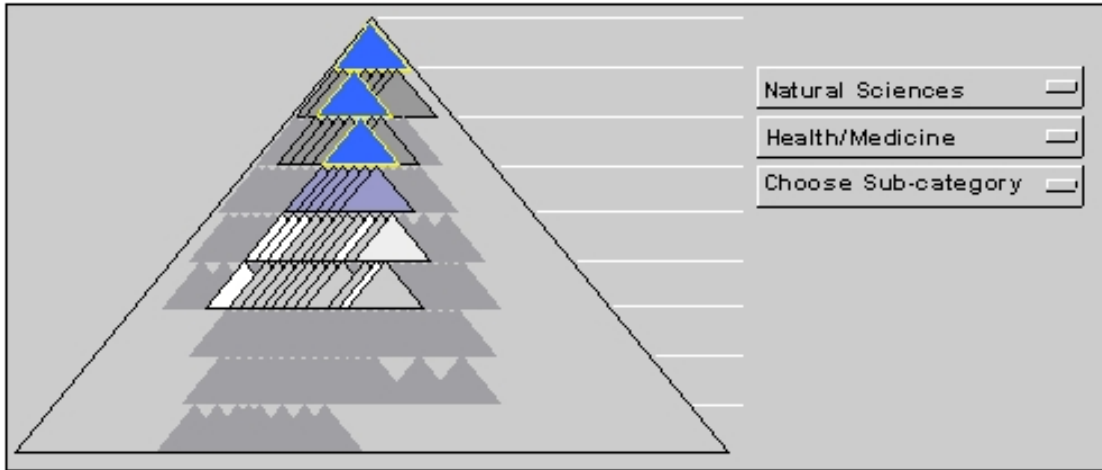


Figure 2.2.5 Cheops display showing the www information with 9 hierarchy levels (grayed out), blue-highlighted nodes indicate the sub category “Health/Medicine” (fifth child in the Natural sciences branch) in the “Natural Sciences” (third child from the root) category in the whole tree [3].

Tablelens [30] is a focus+context (fisheye) technique for visualizing and making sense of large tables. Tablelens fuses symbolic and graphical representations into a single coherent view that can be fluidly adjusted by the user. It allows the users to sort and filter data based on values of individual columns. Users can isolate a single variable or group of variable using row focusing techniques and sort successively on those variables thus creating a virtual hierarchy.

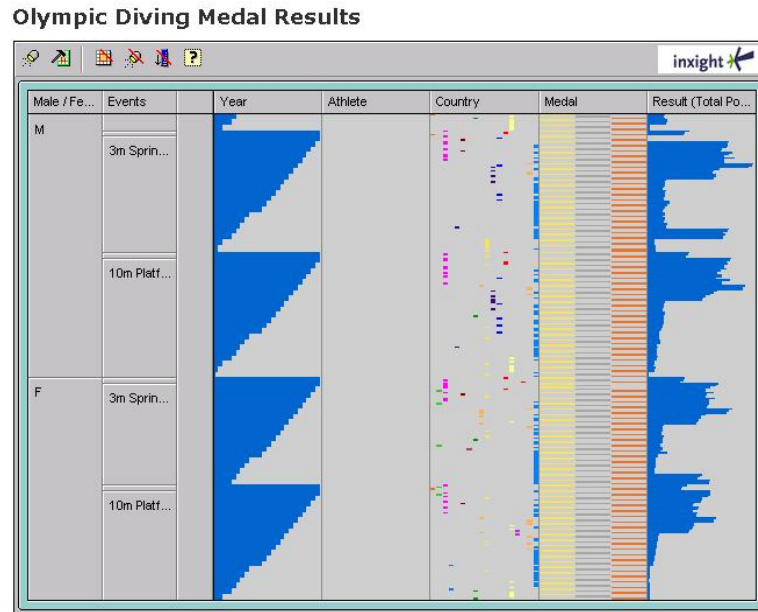


Figure 2.2.6 Table lens visualization of Olympic diving medal results, columns sorted first by Gender, then by Events etc. which in a way shows the hierarchical structure formed by Gender as the first level, Events as the second level etc. [30].

Infozoom [37] is another tool for visualization and exploration of tabular databases. During the Infovis 2003 contest (<http://infovis.org/infovis2003/>) it was demonstrated that Infozoom can be used for visualization, analysis and pair-wise comparison of trees. It displays data sets in tables with attributes as rows and objects as columns.

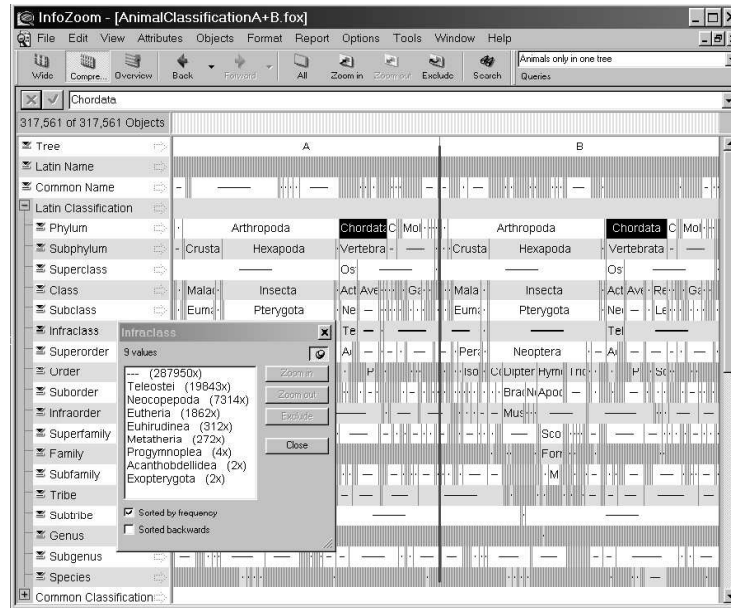


Figure 2.2.7 Infozoom visualization of two animal classifications (Infovis 2003, Contest data) as table, highlighting the differences between two classification trees in black [37].

Each column shows the leaf of the tree and the path from the leaf to the root is stored in attributes (rows) of the table. The attributes can be hierarchically ordered.

Mosaics [15] are space-filling designs composed of contiguous rectangles (“tiles”). Mosaic display is a graphical method for visualizing n-way contingency tables, where the area of the rectangle represents the cell frequencies in the contingency table. The rectangles can be shaded or colored depending on the statistical model used.

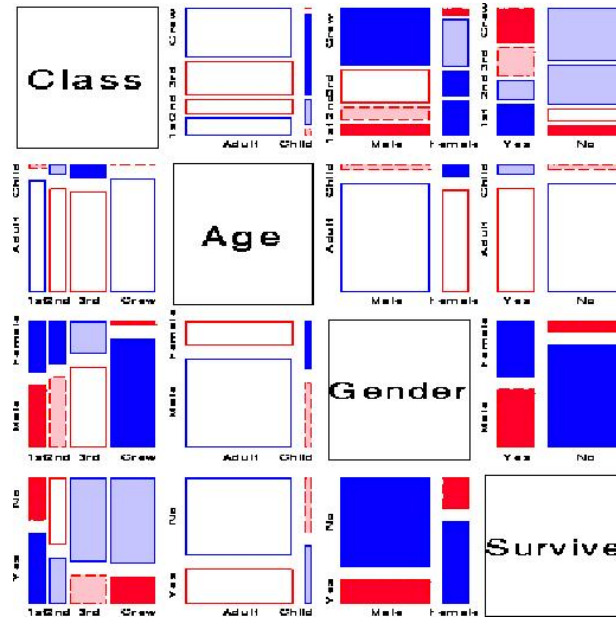


Figure 2.2.8 A Mosaic display showing the eye and hair color statistical data of 696 subjects, with hair color mapped to columns and eye color mapped to rows and the color shade showing the standard residuals from the statistical model [15].

A collection of related mosaics (also known as mosaic matrix) can be used to show all pair-wise relationships of a set of elements in a multi-way contingency table of categorical variables. Mosaic displays are static visualizations created with preprocessed data, which resemble treemaps for fixed level hierarchies.

2.3 Temporal Data: Visualizations

Many applications deal with the data that has temporal dimension for example medical, demographic, production, inventory management, and financial applications. Typically, time-series graphs are used to represent time series data by plotting time on x-axis and value on y-axis. Interest in time-related data has led to many new

visualization techniques that facilitate interactive exploration of time-oriented and visualization of large data sets. Several interactive techniques for visualizing time-oriented information are found in [33]

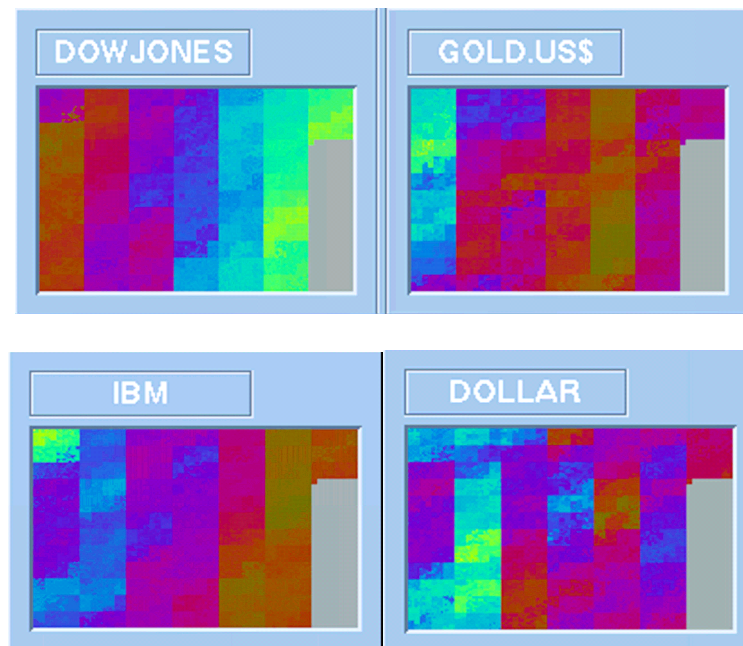


Figure 2.3.1 Pixel based recursive pattern visualization showing the DOWJONES, GOLD, IBM, and DOLLAR stock prices for 7 years and 12 months. Each vertical bar represents a year, subdivided into 12 months. Color shows the stock price, light color shows high stock prices and dark color shows low stock prices. Users map the color changes to those of stock prices, for example IBM stock price has fell sharply after the first one and half month [27].

Recursive pattern visualization [22] allows users to arrange data by some attribute, which is semantically meaningful as shown in Figure 2.3.1. Recursive pattern visualization technique provides a display of time series data divided

hierarchically into different time periods (day, week, month, etc.). For example, it groups all the data items belonging to one day in a first level pattern, those belonging to same week in second level pattern and those belonging to the same month in third level pattern. Each data item is mapped to a colored pixel, color showing the attribute value of the data item, thus enabling the users to identify patterns.

Spiral visualizations [9] uses Archimedes spirals where time progresses along each spiral lap with time stamps aligned to spokes on the spirals.

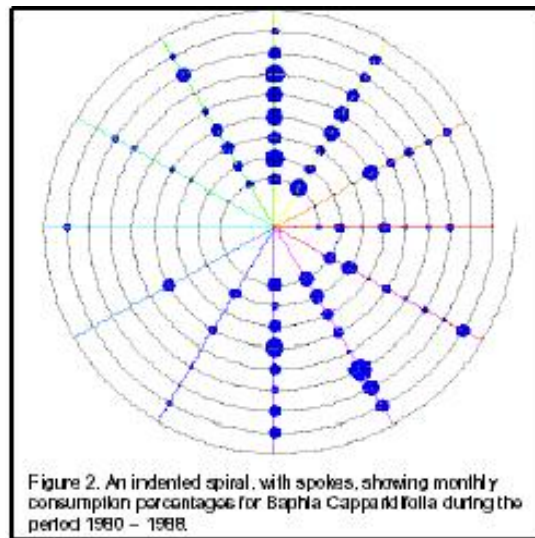


Figure 2.3.2 A spiral display showing the monthly consumption for Baphia Capparidifolia by chimpanzees during 1980-1988. Each lap corresponds to a year and spokes correspond to the months in the year. Area of the blot shows the actual consumption percentage. Users follow the spiral laps or spokes and observe the areas of blots for tracking consumption trends, for example the highest percentage consumption occurred in a particular month for all the years [9].

Bars, blots etc., are used to represent data at a specific time period, and color and area/height show other attributes. Zooming in and out on subsets of the spiral, animating and controlling the duration between each spiral achieve interaction. For data sets with multiple time series attributes, spiral displays show each attribute with a different marking at each time point in a 3D projected view.

Disk Tree and Time Tubes [11] are examples of Web Ecology and Evolution Visualization (WEEV) techniques for monitoring WWW web log data. Disk Trees represent the web site hierarchy by concentric circles, where each circle corresponds to one level of hierarchy, showing web pages as points, links as lines between points, number of page requests by line thickness and brightness of the line going from the referring page to the destination and page life cycle stage by color (red for new, green for continued and yellow for deleted pages). A new dimension of time is added to Disk Trees, by laying out several Disk Trees on a time axis, resulting in Time Tubes showing the evolution of the web over longer time periods. For each time period of web ecology tube, a disk tree showing that week's data is inserted in the tube. Any week's data can be viewed by clicking on the slice representing that week, or multiple week data can be viewed with all the slices in a row, mapped on to a time axis. Interaction is achieved by mouse-over, zooming etc.

All the disk trees can be stacked up in a time tube so that users can animate over time, automatically or in response to their input, and view the data over all time periods. However, identifying the changes and interpreting the series of changes depends on the users' ability to do comparisons between different time points.

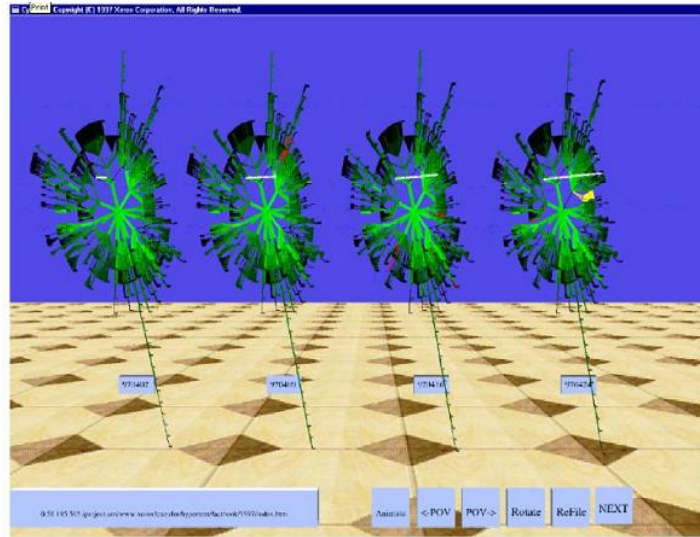


Figure 2.3.3 Time tubes and disk trees showing the WWW web ecology and page visitation counts for four weeks [11].

Disk trees [12] of page visit counts are arranged in visualization spread sheet. The concept of visualization spreadsheet is based on the tabular layout where each cell shows abstract visual representation of large data set, the ability to modify cell contents using operators and the automatic update of cells, based on their dependencies, when they are manipulated. Users can perform visual operations that are synchronized across rows and columns, such as “visual usage pattern subtraction”, for example column one shows the result of subtracting week one from week two, while column two is week three subtracted from week two. This lets the users quickly understand the usage patterns over time which would other wise be complex to notice.

Information visualizations with multiple coordinated views enable users to rapidly explore complex data and discover relationships [18]. Initially interactive StarField [1] visualizations, similar to a scatter plot, are used for exploring web log data [19] where individual page access requests are shown on two-dimensional displays, with size and color showing some other attribute. Zooming and filtering options are used to support the interactive exploration of data. Different visualizations (time vs url (macro/micro), time vs hostname, client host vs url, referrer vs url, referrer vs time, etc.) showing different patterns over time are presented. The difference in web page requests during weekends and regular working days was clearly identified using those visualizations.

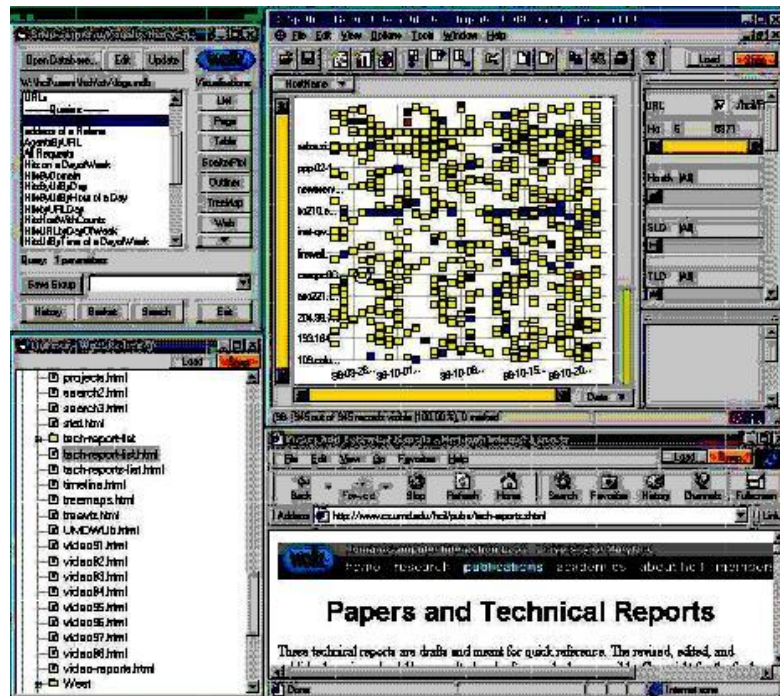


Figure 2.3.4 Coordinated visualizations for context: Visualizing the WWW log data for HCIL website (www.cs.umd.edu/hcil) for 2 months [18].

Since the web log data is highly context dependent, it might be helpful to consider web site topology for interpretation of the log data. The coordinated visualization [18] in Figure 2.3.4 has an outline window on the left hand side showing the hierarchical view of URL's on the site web browser window in the lower right corner displays the selected web page and the Spotfire [38] display spots requests for a given URL, with time on x-axis and hostname on y-axis. All this added context helps the users understand the over all patterns in the data set.

Time Searcher [17] is an interactive visualization tool for querying and exploration of time series data. It introduces a new graphical technique for querying time series data, known as Timeboxes. Timeboxes are rectangular regions that are placed and directly manipulated on a timeline, with the boundaries of the region providing the relevant query parameters.

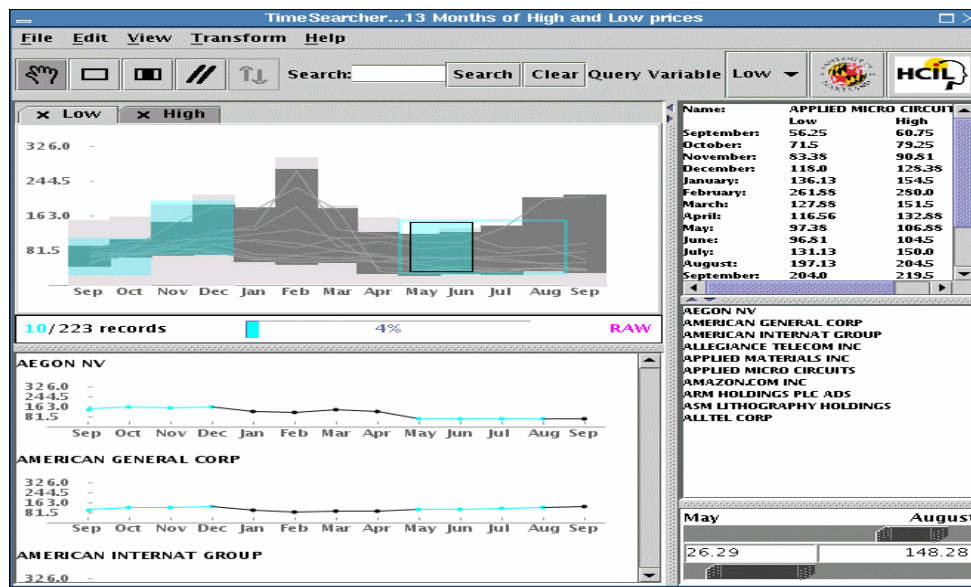


Figure 2.3.5 TimeSearcher showing the stock data for 12 months [17].

TimeSearcher provides a graphic overview of all the data items on the left top window along with each data items graph on the left bottom window. These two windows are synchronized: clicking on an item in one window will lead to updating of the other. The details of the data item can be read from the details window on the right top, list of items can be read from the right bottom window. A variety of options are provided to specify and modify different queries.

The financial website, smartmoney.com [35], provides various maps to observe the performance of several hundred stocks at once, thereby enabling the users to spot the investment trends and opportunities. Its Map of the Market presents a graphical overview of the price performance of more than 600 stocks at once (Figure 2.3.3).

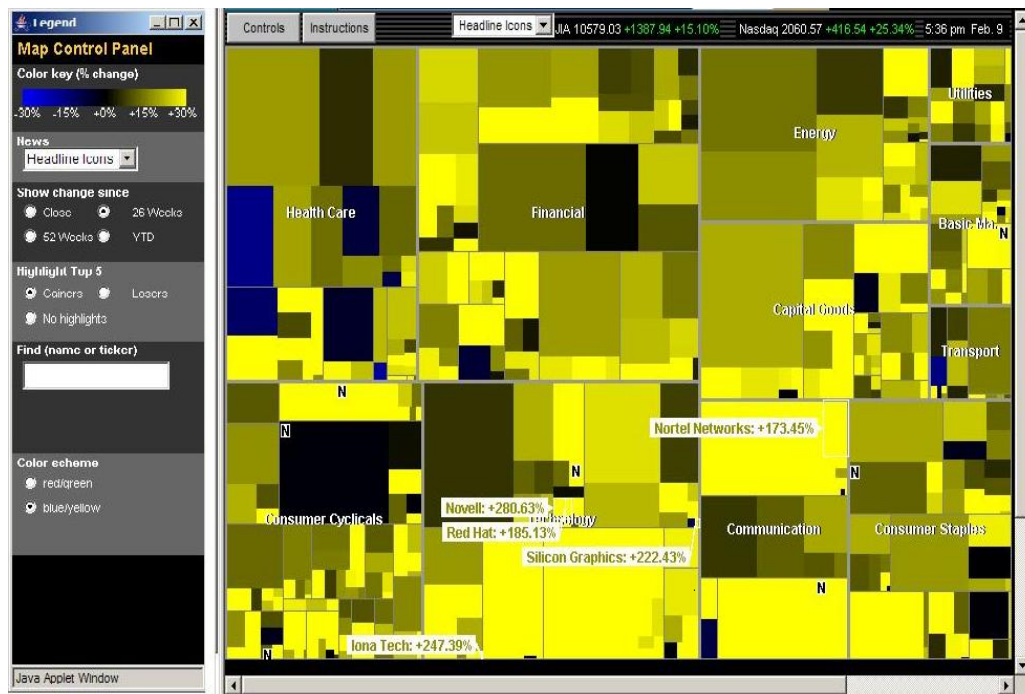


Figure 2.3.6 SmartMoney.com's Map of the Market showing the % change in stock prices on Feb 10, 2004 since the past 26 weeks [35].

The visualization in the Map of the Market is a customized version of traditional treemap display. Each rectangle in the map represents an individual company, with size of the rectangle proportional to the company's market capitalization and color proportional to the price performance, light colored rectangles indicate the %increase in stock prices and dark colored rectangles indicate %decrease in stock prices. Also, a three level hierarchy is imposed, defined by industry, sector and the company itself at the lowest level. The market shifts for an industry or a sector can be studied by zooming in/out in the map. The Map of the Market provides four different time periods: previous market close, 26 weeks, 52 weeks, and year to date. Any one-time period can be selected by clicking on the radio buttons. The default view gives the stock performance of the company since the previous market close. By fixing the end time period, it calculates the %changes in stock values from the start period (selected by the viewer) to that end period and colors the map accordingly. Also, the Map of the Market highlights the top 6 gainers or losers depending on the users' choice, thus enabling the users to quickly identify the leaders and laggards in all the time periods.

The stocks in Figure 2.3.3 are light colored indicating that the stock prices have increased from the past 26 weeks. The highlighted companies (rectangles with the popup showing the company name) are the top 6 gainers from the past 26 weeks.

This is the closest related work to the study described in this thesis. The SmartMoney application is designed for monitoring stock data and hence has specific significant pre-determined time periods.

Calendar and time schedules are the most familiar time lines. Calendar visualizers [26] use several simultaneous displays, zooming and fisheye like quality to display detailed appointment information and to coordinate appointments and meetings. In cluster and calendar based visualization methods, temporal patterns and trends on multiple time scales (days, weeks, seasons) are identified simultaneously by clustering similar daily patterns and visualizing the average patterns as graphs and the corresponding days on a calendar [42].

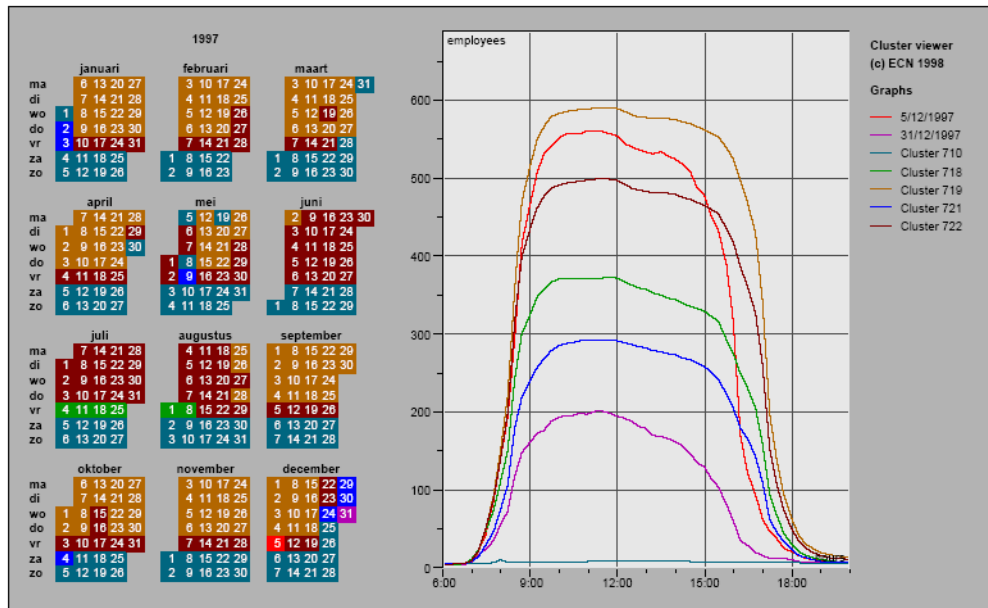


Figure 2.3.7 Cluster and Calendar based visualization of number of employees and their working hours/patterns in ECN. Patterns are shown as time series graphs with number of employees on y-axis and daily working hours on x-axis. Clusters are shown on the calendars. Color shows the corresponding clusters and patterns. For example, cluster 722 shows that there are fewer people in summer [42].

Patterns are shown as graphs and clusters on the calendar. Colors are used to indicate corresponding clusters and patterns. The average value per cluster is shown as a colored graph adjacent to the calendar view in which each day is colored according to the cluster to which it belongs. The calendar view and the graphical view of patterns are tightly coupled. Clicking on a day in the calendar would highlight the remaining days, which belong to the same cluster.

Clinical data consists of treatment information, laboratory data, and physical findings observed/collected over time. Cousins and Kahn [13] have developed a visualization technique based on the “time line” concept, an abstract entity to represent a sequence of events ordered by time. The user interface supports several options (slice, filter, overlay, new, and add) to modify timelines.

Lifelines [29] present an interactive technique for visualizing individual’s medical history data over time. The screen is horizontally divided to represent various time periods under which a person’s medical history is graphically depicted. Symbols such as horizontal lines, color of the lines, icons are used to represent time-oriented events in a person’s life. Lifelines give an overview of the entire medical record, as the users zoom in to subsequent smaller time intervals the display is updated automatically showing the details. Additional information like ultrasound scan images can be displayed to get more context information.

Nowell et al., [27] discusses shortfalls of certain techniques for visualizing temporal data. In particular, users may not be able to recognize major changes

between two displays if the change is abrupt. They recommend that visualizations for temporal data include the information from the preceding or following period in a way that is distinct from the present period.

2.4 Summary

This survey of related work illustrates issues related to time series and/or temporal data. Visualization of hierarchical data illustrates various perspectives that are appropriate for these data sets. Factors like layout, zooming, and interaction lead to a variety of ways to display such data sets and interpret these data sets [2, 3, 7, 15, 20, 25, 31, and 37]. Time series and temporal data visualizations present various techniques to address issues like periodicity [19, 26, 29, and 42], support multiple time series attributes [9], and querying [17, 33]. Visualizations with multiple coordinated views or context information help users explore the data rapidly [11, 12, 18, and 19]. Even though applications like SmartMoney help visualize stock market data over time, design and implementation of general purpose systems that rapidly combine hierarchical data visualization with the time series visualization may be difficult.

Chapter 3

User Requirements and Design

Treemaps are extensively used for visualizing hierarchical data. Earlier implementations were limited to visualizing predefined hierarchies. Later, flexible hierarchy was introduced in Treemap 4.0 (www.cs.umd.edu/hcil/treemaps) which enables the users to impose new hierarchical structures with the available data attributes. Current implementations are time invariant. Most of the applications of treemaps involve monitoring data which is often time oriented. For example, the stock prices in stock market data, production data for monitoring amount produced or lost production etc. Traditionally, such time oriented data is collected separately and is viewed separately in treemap with the visualization showing data at any one time period. Or a new attribute showing the %change between first and last time periods is added manually to the data set. Although it gives a summary of changes over time, it is only a part of the time series data analysis.

To address this problem, a new temporal dimension is introduced to visualize the time series data in a single display. Time series data is a series of real value measurements taken at consecutive points of time. The goal of visually exploring this time series data is to understand how data is distributed at any time point t , how it varies over time from t_i to t_j and to identify patterns and/or similarities.

Temporal treemaps are developed to support time series data sets, which is only one component of “time-oriented” data [33]. Other types of time-oriented data

include existential changes (creation and deletion of data items) and spatio-temporal changes (changes in location, position, etc.). Data items have one or more static attributes and one or more time-varying attributes, where the number of time points and the interpretation of those points remain the same for every data item in the data set. These time-varying attributes could be either categorical or numerical. Categorical time series data sets involve attributes with discrete values that change over time. On the other hand the hierarchical structure of the data could also change over time due to existential changes. New data items can be added and existing data items can be removed from the data set. These aspects of time oriented changes are not supported by temporal treemaps. For categorical attribute changes, if the focus is on monitoring number of items in each category, temporal treemaps provides a way to track this number of items over time. This process is discussed in detail in Section 5.6 and is demonstrated in Chapter 5, Section 5.5.

The hierarchical structure of data is represented by a set of nested rectangles where the attributes are mapped to the size and color of the rectangles. Temporal treemaps map time series data attributes to color attributes of treemaps. The time varying data attribute could be mapped to the size of treemap nodes. As the attribute values change over time, the size of the rectangles changes, resulting in different aspect ratios, which makes it hard to compare the rectangles. Experience has shown that changes in size of the rectangles are hard to follow and hence size attribute is not used for showing temporal attribute changes.

Figure 3.1 shows the temporal treemap visualization of oil production data. When the data set is loaded, data items are displayed as a set of nested rectangles showing the hierarchical structure of the data in the top left corner of the application. These rectangles are generated by a recursive space-partitioning algorithm (one of the three algorithms: squarified, slice and dice, and strip) [5]. Each rectangle at the leaf level (lowest level) is a data item and is labeled by its name.

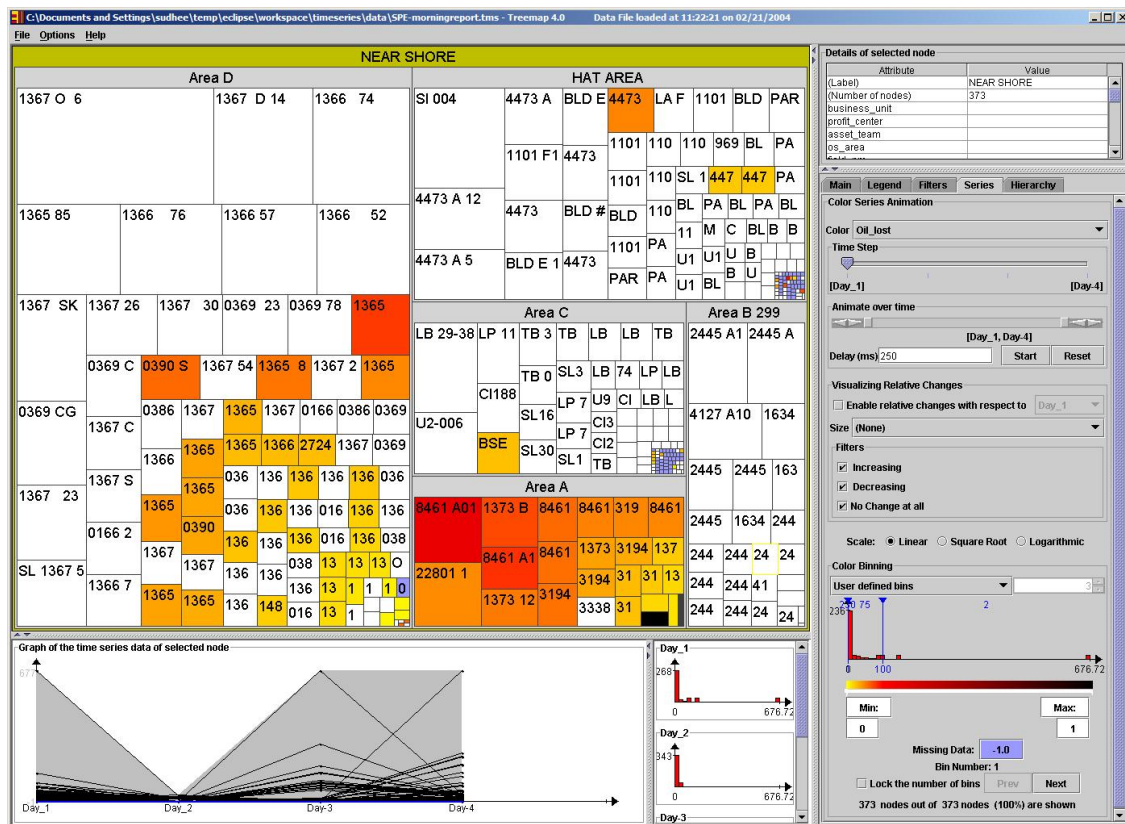


Figure 3.1 Temporal treemap application window: treemap window, time series graph window, histogram window, details table and series tab.

The position of that item in the hierarchy can be identified from the labels, from the item to the root. Detailed information of any data item can be obtained by simply

clicking on the rectangle for that item; this would display the details in the upper right-hand window (Figure 3.1).

All the options that support interactive exploration of data are distributed into five tabs: main, legend, filters, series, and hierarchy tabs. The main tab provides the options for choosing partitioning algorithm, font size, border size, and overlay label option, legend tab consists of label, size and color options. Any available data attribute can be used for color coding the treemap where as only numerical attributes are used for size coding. Dynamic query filters in the filters tab can be used to filter out unwanted or uninteresting data items from the display. When users select a range of values, the data items that fall outside the range are grayed out and can be hidden using “hide filtered” option. A new search option is added to filters tab, which enables the users to search the treemap based on a substring. The non-search results can be hidden from treemap by filtering them. The hierarchy tab consists of the controls for specifying and changing flexible hierarchies.

The lower left corner of temporal treemaps shows the time series graph overview of all data items. Initially the time series graphs of all the data items are displayed. Individual data item’s time series graph can be displayed by clicking on that item in the treemap. Two or more data items can be compared using the graph overview by selecting (Ctrl+click on the rectangle) them in the treemap. The graph scale can be normalized depending on user needs.

The lower middle area shows the data distribution in the form of histograms for all time periods. They are displayed in a vertical linear fashion.

The series tab on the lower right corner consists of the options for exploring the time series data. Any time varying attribute can be selected from a scroll down list of all time series attributes. The binning widget shows the series minimum and maximum. Users can bin the data and assign colors to each bin. Users have to group the data depending on its significance, specify the colors either discrete or continuous spectrum of colors, and interpret the resulting visualization by mapping attribute values to the color of the rectangles in the treemap.

Users can navigate through the time periods using the time slider or animate over a range of time periods. As the slider moves, data is updated and color of the rectangles is changed in the treemap accordingly.

3.1 Overviews

Temporal treemaps provide three overviews: treemap overview showing the hierarchical structure of the data set, graphic overview showing the time series data of selected series attribute, and histogram overview showing the data distribution over time.

The treemap display utilizes all the available space and displays large hierarchical data sets up to 100,000 nodes on a single screen. This display enables the users to zoom in to a particular sub tree, explore the details of the data items in that sub tree. Double-clicking on a rectangle would zoom in on the rectangle to occupy the entire display space while mouse right-click would zoom out level by level. The size and color of the rectangles show other attributes. Here color shows the time series attribute values at any time point.

The graph window provides another form of overview by displaying the extreme values that can be found in the data set at each time point known as a “data envelope”, this overview is optionally shown in the background of the time series graphs. The graph overview provides further support for browsing the data set.

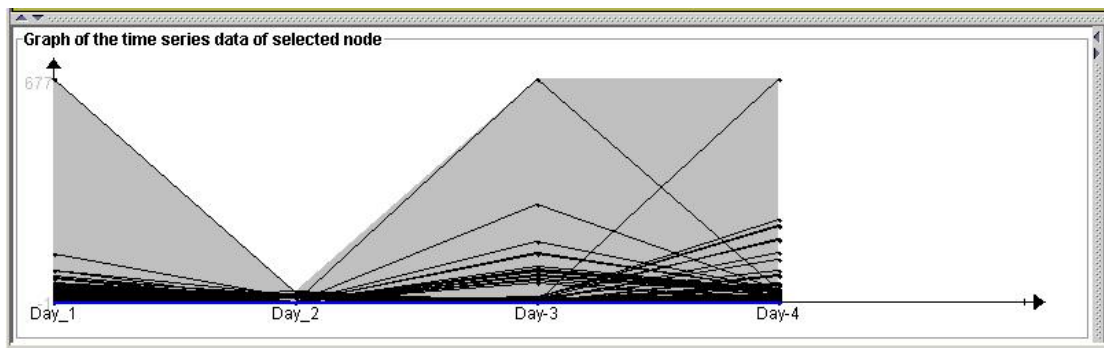


Figure 3.2 Timeseries graph overview of oil production data collected from 373 oil wells, showing the graph envelope.

Mouse over on a graph envelope line would highlight that line thus displaying the individual item in the context of the larger data set. Also the name of the item is displayed in a pop up, along with the value of the item at the time point closest to the point where the mouse-over occurred and the corresponding rectangle in the treemap is highlighted along with the item details. Mouse over on a rectangle in treemap would highlight its corresponding time series graph in the graph overview, with the item name shown in a pop up window. This tight coupling in response to light weight mouse movement will encourage exploration based on visual examination of the graph overview.

Overdrawing and clutter might cause the graph overview display to become less useful for large data sets. Furthermore, the computational overhead of drawing and updating the graph overviews and processing the mouse over handling can lead to substantial performance degradation with large number of data items ($10^4 - 10^6$).

The histogram representation provides a limited overview by displaying the data distribution over time in a linear vertical fashion. As this display shows a small number of time points, users often have to scroll to effectively use the display. Another possible overview would be to display the histograms in a separate visualization spreadsheet where each cell would show the data distribution at any time point.

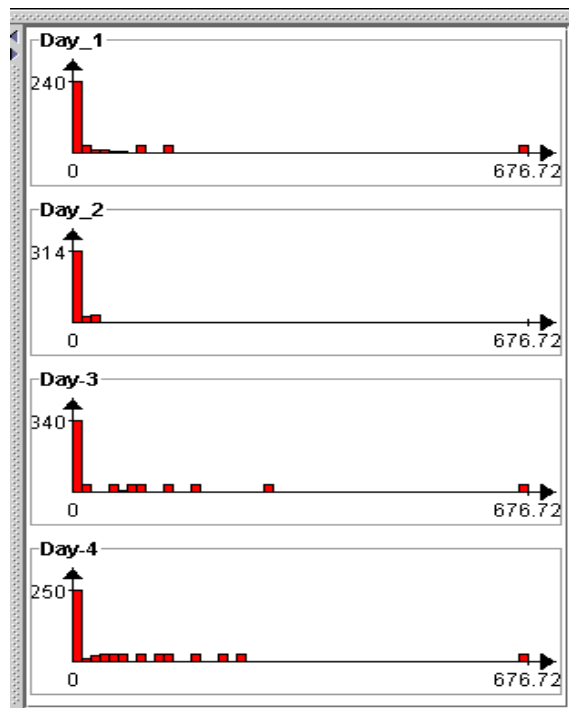


Figure 3.3 Histogram Overview of Oil Production data showing the amount of lost production for 4 days.

The problem with this approach is that displaying them in a separate window would involve switching between two windows. This window switching would result in flow interruption for the users and the interaction may not be utilized to the full extent.

3.2 Missing data

Missing data is a form of uncertainty which is the common problem in time series data that can be attributed to either measurement failure or error [10]. The simplest method for treating this uncertainty is to use some noticeable supplemental value. Twiddy [41] proposed a way of handling missing data by using gray shades called a “restorer” technique. The display combines the exact data and missing data through visual blending but can be distinguishable on close examination. The user distraction is minimal in restorer.

Temporal treemaps treat missing values by using a supplemental value and assigning a color to that value. The value chosen must be different from the actual data values. Users can specify this value in the data file (See Appendix A) which is plugged in for the blank cells. This value is excluded from the relative change calculations and aggregation.

Figure 3.4 shows the missing data value and color options. Users can click on the button and select a color to identify the missing data points. The color selected for missing data value should be different from the normal color gradient chosen for time series values to avoid distraction. As the users navigate from one time period to other

time period, they can easily identify the data items with missing values by their specific color.

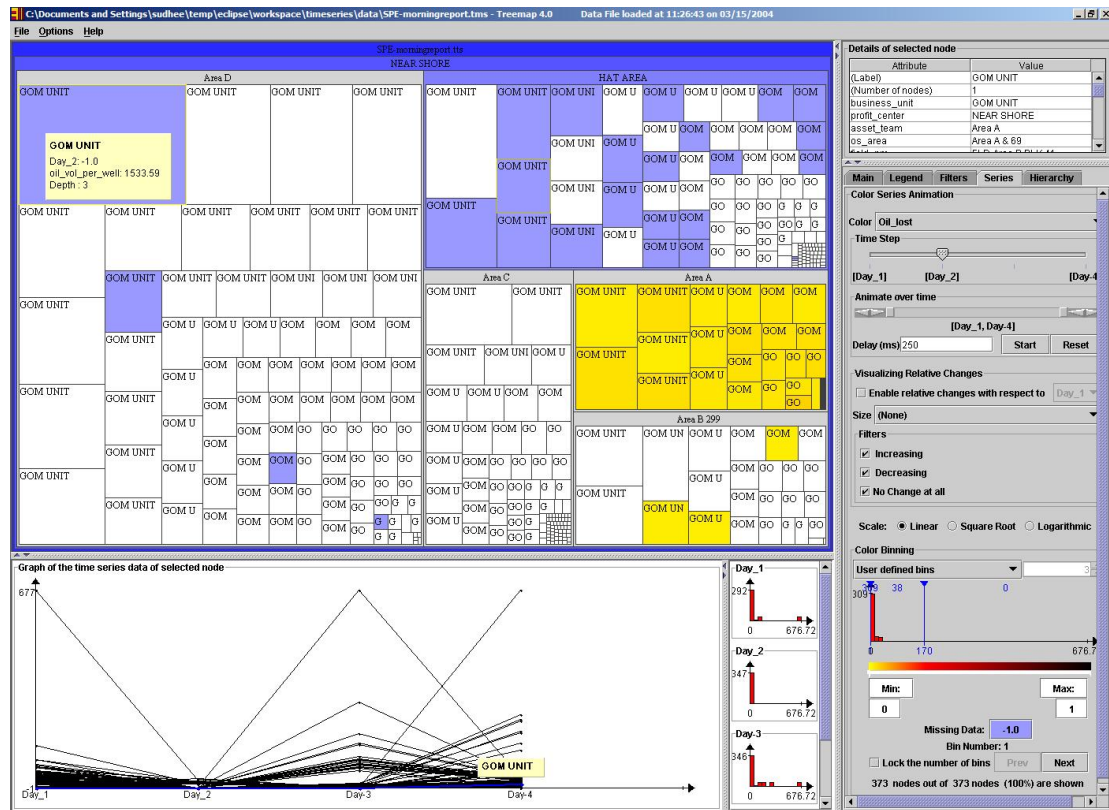


Figure 3.4 Monitoring Oil production data, showing the missing data replaced with a value of “-1”, shown on the lower right corner; light blue colored rectangles represents the items (oil wells) with missing data on day_2.

Treemaps facilitate the aggregation of lower level data items by controlling the visible hierarchy level. An aggregate value (average, maximum, or minimum) is calculated and assigned to the rectangle representing the sub-tree. Missing data items are not considered in calculating this aggregate value. It might be helpful to show the missing data items in the aggregated rectangle. One option would be to aggregate the

missing data items separately and show them as a portion of the actual aggregated rectangle. Another possibility would be to show the missing data items in their assigned positions, but aggregate only the values of the data items that are present. These two ideas are not yet implemented.

This approach of using supplemental values for missing data has two problems. First, if the supplemental value used is 0 or any negative value, then these nodes would be ignored by treemap if size coded with any time period. This problem is solved by adding a value of “1” to zero or less than zero valued data items, so that all the data items are retained in the display. Since we are using a unit value for all the nodes, the significance of the rectangle sizes is not preserved, as the rectangles with values -100 and -1 would be equal in size. A future possibility to address this problem would be to provide a menu of options: 1) Use absolute value, 2) Ignore negative values, and 3) Add the absolute value of the most negative number +1 to all the values. Second, time series graphs of the data items are misleading in understanding the trends. The supplemental values must be excluded from the time series graph display. This feature is not yet implemented.

The %changes cannot be calculated if the data item in the reference time period has a zero value; it would result in divide by zero error. For such data items, the %change is shown as +100% or -100% depending on its attribute value. Another possible option would be to have a special value that implies “Cannot calculate the %value for this item”.

3.3 Overlaying labels at selected hierarchy level

When the data set is large and deep, data items may not be obvious as the space is lost in displaying the hierarchy levels. More over while observing the changes over time focus is on leaf level items. In such cases borders may be removed and the pixels can be reclaimed for displaying data items. Context information (hierarchical structure and labels of items in hierarchy levels) is lost when the borders are removed. Text labels are important to understand the context in which visualized data appear.

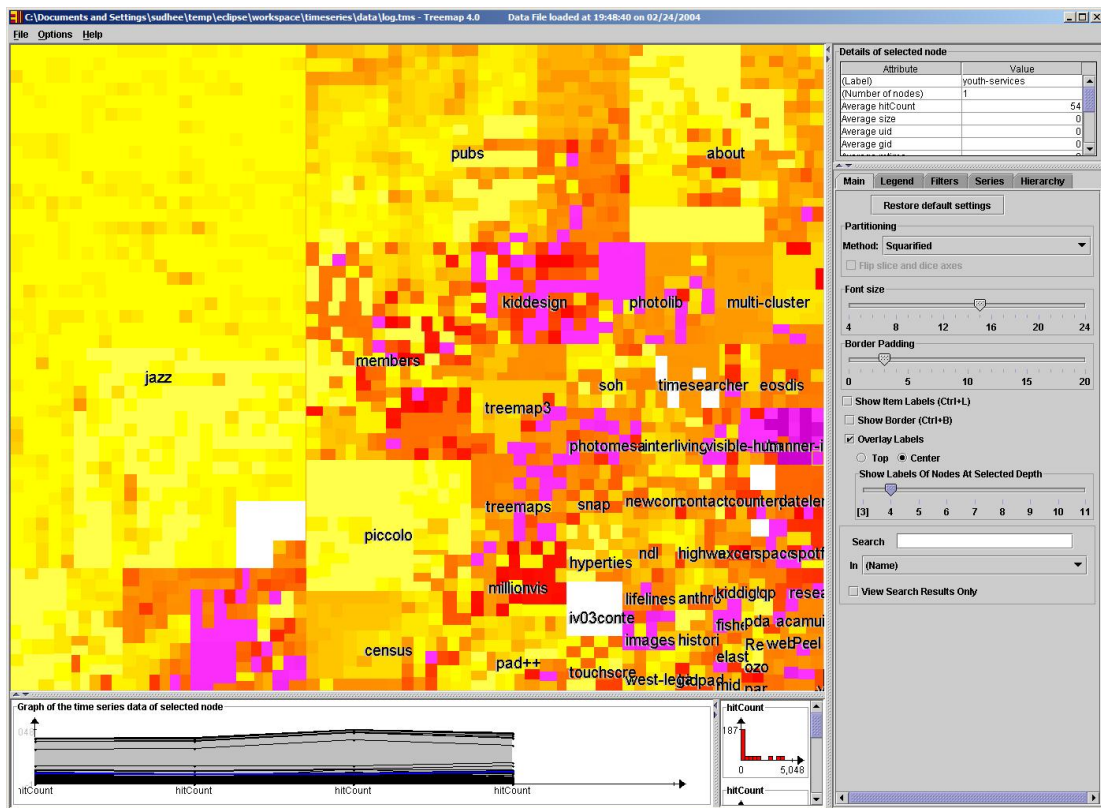


Figure 3.5 Visualizing HCIL web logs grouped by directory structure, without borders and with overlay labels at center of node, showing the labels at hierarchy level 4.

In such cases labels are overlaid on top of the rectangles by extending the use of excentric labeling [14] dynamic technique for the scatter plot. Users have the flexibility to control the level of detail by selecting the hierarchy level so that the labels of items at that level are shown.

The overlaid labels are displayed either at the top or center of the corresponding rectangle, depending on the user's choice. The labels are displayed in such a way that they are visible on a light and black background. The labels are first painted with black ink and then painted with white ink, with one pixel shifted up.

The label overlaying is important for visualizing time series data. When the users navigate from one time period to another, their focus is on the data items with an intent to identify the color changes over time. In such cases, one possibility to facilitate easy exploration would be to remove the borders/labels when the users grab the time slider and repaint the borders/labels when the users release the time slider. Since the slider can be operated with the key board shortcuts, repainting the borders/labels caused a flickering effect when key board shortcuts are used to operate slider, which otherwise worked well with the mouse operation.

3.4. Other Features

Filters and Search

Users can filter the unwanted/uninterested items from the display using dynamic query sliders [1]. As users select a range of values for any numeric attribute, the items that fall outside the selected range are grayed out dynamically and can be hidden from the display. In visualizing time series data, it might be interesting to filter the data

items that have increased, decreased, or remained same in value from the previous time period. Temporal treemaps provide these filters in the form of three check boxes as shown in middle right corner of Figure 3.6.

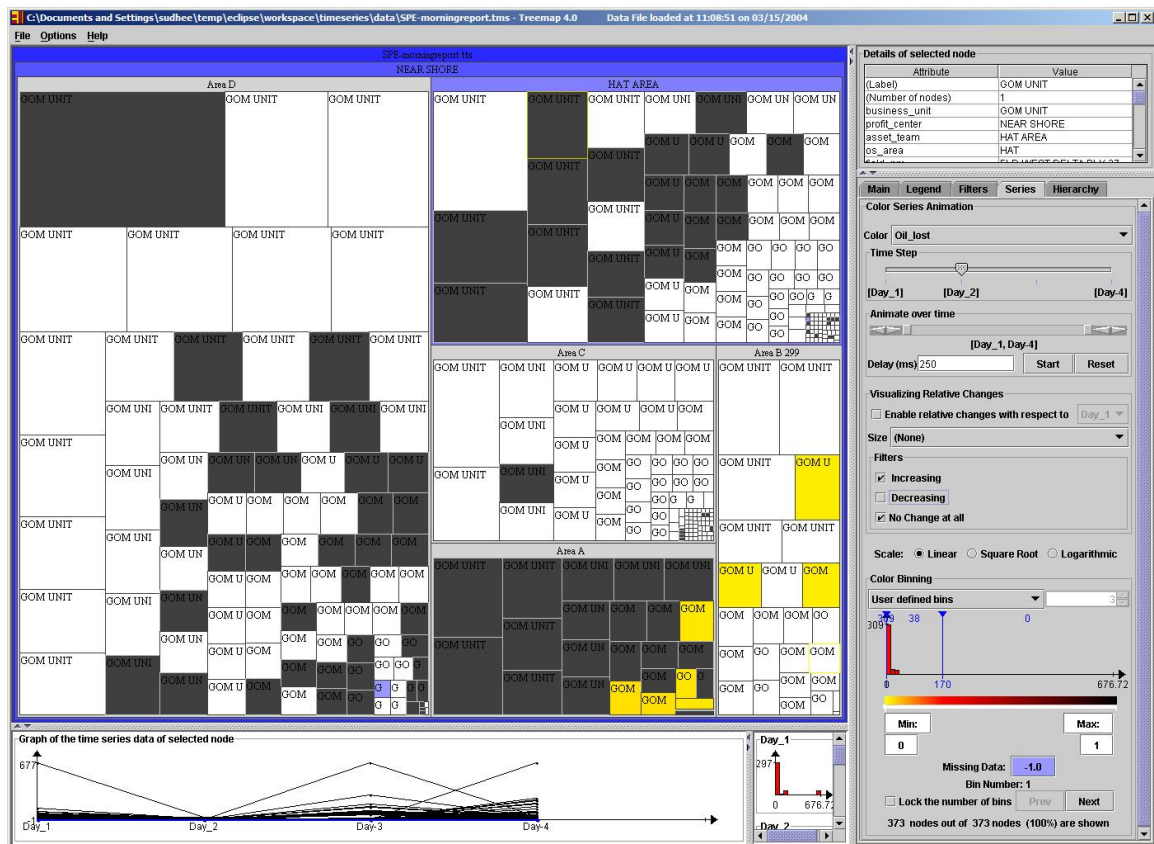


Figure 3.6 Visualizing Oil production data, grouped by Business center and asset team, size is proportional to amount of estimated production of oil, color proportional to amount of lost production, grayed out rectangles show the wells whose lost production on day_2 is less than that on day_1.

The three check boxes are selected by default showing all the data items. Unselecting any of the filters would gray out the items, which fail to pass filters. For example “decreasing” filter is unselected in Figure 3.6. The gray color rectangles in

Figure 3.6 show the oil wells whose delta production loss has decreased on day_2 compared to that on day_1. Users can get a quick overview of what has increased, decreased, and remained constant from the previous day with these filters. Filtering the data items using the dynamic query sliders or time series filters will also filter the corresponding time series graphs from the display.

Among the filtering features, it is helpful to be able to do a selection or a filtering with a sub string search or eventually a regular expression search.

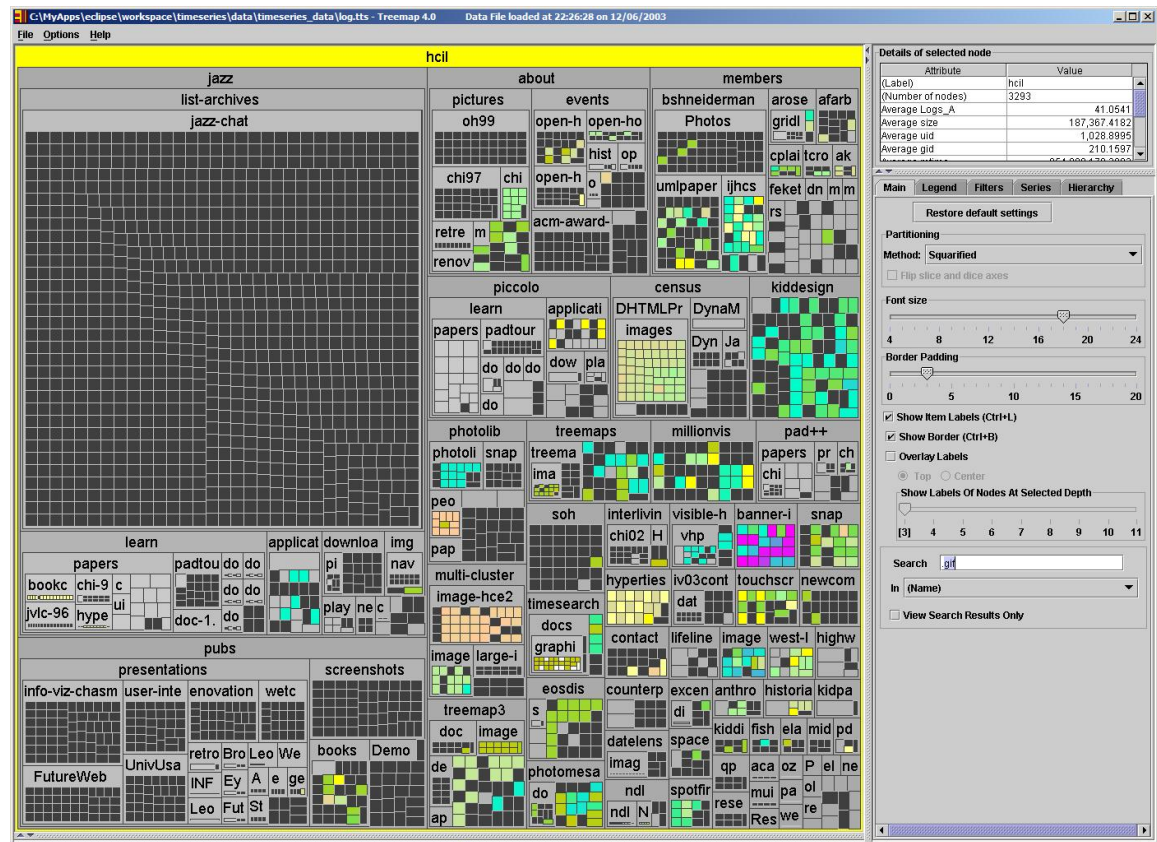


Figure 3.7 Visualization of web logs for HCIL, grouped by directory structure. Search for “gif” images, colored rectangles represent gif images, grayed out rectangles are non-search results.

Users can search for a sub string and filter out all the non-search results. An item is described as a search result if that item name or attribute value starts with or contains the search keyword. All the non-search results are grayed out enabling the users to identify the search results easily (Figure 3.7). In the initial implementation, search results were highlighted. This is modified as highlighting prevents the users from visualizing the item's color attribute and the color used for highlighting may be used in color coding as well.

Figure 3.8 shows the user interface for specifying the search string to search treemap in any available attribute.

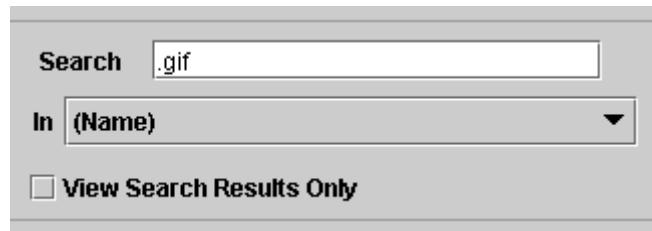
The image shows a user interface for searching within a treemap. It consists of a light gray rectangular box. Inside, there is a 'Search' label followed by a text input field containing '.gif'. Below this is an 'In' label followed by a dropdown menu showing '(Name)' with a downward arrow. At the bottom, there is a checkbox labeled 'View Search Results Only'.

Figure 3.8 User interface for specifying sub string search based filtering.

3.5. Getting started with temporal treemaps

3.5.1 Visualizing numerical attribute changes over time

The envision process for visualizing time series data presents a procedure to help the users to get started with the application and to explore the time series data. The data set can be created manually or can be generated using TTSGenerator from several treemap data sets. A brief explanation of TTSGenerator is given in Appendix C.

After loading a treemap time series data file, users can select any time series attribute from the list of available attributes. This would display all the controls and color the treemap with some default gradient. Users then have to create groups/bins based on the significance of data attribute and specify colors to each group. The colors can be discrete or they can form a continuous spectrum. It is helpful to create a spectrum of colors within the group and discrete colors between the groups. Colors should be chosen in such a way that the first and last groups can be easily distinguished from other groups to observe the minimum and maximum values in the series. Users can select to view %changes with respect to a reference time period. These %changes can be divided into at least two groups showing the %decrease (minimum to 0) and %increase (0 to maximum) in data items. These groups can be further divided into smaller groups (-100%, -50%, 0, 50%, 100%) depending on the user needs.

Once the colors are specified, users can move the slider to see the changes over time. The slider can be operated with key board shortcuts (left and right arrow keys). Users can specify a range of time periods with the range slider and select to animate over that range with the specified delay. They can switch between border and no border and choose to overlay labels when borders are removed. These settings can be saved and reloaded for later use. Users can select search and filter options and graphic overview to interactively explore the time series data.

The time series graph overview can be used to see the trending information for one or more data items. Users can select the data item with dark color and see the

time series graph for that item or can select any time series graph and view its details in the treemap.

3.5.2 Visualizing categorical attribute changes over time

Temporal treemaps can be used for visualizing categorical attribute changes over time by tracking the number of data items in each category. The input data is stored in different treemap data sets with each file corresponding to one time period. Load any one file and create a settings file with the desired visualization depending on the users needs. One possibility would be to group the data by categorical attributes that user is interested in. Then use the same attribute for color coding and choose strip partitioning method to preserve the order in which groups are displayed. Create a list file consisting of all the files for all time periods. A sample list file is shown in Appendix B. Reload the settings file along with the list file. This would display the time periods on the top right corner, beside the detail on demand table. Users can view any time period data by selecting that time period from the list.

To view the number of items in each category, all the data items in each category are aggregated by their count and a new time series file is created using “Export” option in the “File” menu, with the number of items as time series attribute and categories as data items. A supplemental value “-1” is added if the category does not exist in any time period.

Load the new time series file and color the treemap with the time series attribute (number of items). Group the data with the static attribute and label it with the time

series attribute. Select colors based on the user needs and navigate thru the time periods using the slider. This process is demonstrated in Chapter 5, Section 5.5.

Chapter 4

Temporal Treemaps Implementation

Temporal treemaps was an extension to the original treemap implementation and was developed on top of the existing treemap code. Treemaps as well as temporal treemaps were implemented in Java 2, using the Java Swing toolkit for user-interface widgets.

Treemaps is a research prototype that is available for free for educational use and also licensed for commercial use. Many features were added to treemaps over time which included years of developmental work and substantial redesign. A list of people who have contributed to this research project is available at <http://www.cs.umd.edu/hcil/treemap>.

This chapter provides an overview of temporal treemaps implementation, along with a brief overview of the original treemap implementation.

4.1 A Tour of the Code

This section provides a quick overview of the current implementation of treemap code. Initially all the classes were in a single package, new packages were introduced recently. The main package consists of classes that drive the treemap application: `TreemapProgram` initializes the program, `TreemapModel` constructs the tree data structures and does the initial calculations, `Treemap` paints the nested rectangles based on the layout algorithm and `ControlPanel` initializes the available controls to

modify the display. Several other classes in the main package support additional features like binning, color binning and layout algorithms. A variety of packages provide the features of Treemap functionality.

- `edu.umd.cs.treemap.filefilters`: displays file dialogs depending on the file type. The dialogs let the users pick a file to load from a list of files. They display only the files that have the chosen extension and hides others from being displayed on the list. There are several types of files, `tm3`, `tts`, `tms`, `xml`, `tnv`, `lst` etc.
- `edu.umd.cs.treemap.hierarchy`: handles the flexible hierarchy feature. These classes manage the addition and deletion of attributes from the hierarchy and also notify other classes when such things have happened.
- `edu.umd.cs.treemap.filereader`: performs the basic input output operations. It has separate classes for reading data from `tm3`, `tts`, `tnv` and `xml` and for writing data in `tm3` and `xml` format.
- `edu.umd.cs.treemap.listener`: event listeners classes for displaying popup on mouse events. `NodeClick` and `NodeCursor` listeners capture the mouse click and mouse moved events on treemap where as `LineClick` and `LineCursor` listeners capture mouse clicked and mouse moved events on the time series graphs in the graphic overview window.

4.2 Data Management

4.2.1 Input File Format

Temporal treemaps uses a simple file format for input file. Data files are simple text tab delimited files and pertain to a specific format.

A valid treemap time series data file consists of series of headers that explain attribute names, series names, time stamps and attribute data types as explained below.

1. Number of time stamps for each attribute. For single valued attributes it is 1, for multi valued / series attributes it is the width of time series.
2. Time Series Name: name of the time series to be monitored. For single valued items, enter “Single”.
3. Attribute names: For static attributes, it is the attribute name. For multi valued attributes, it is the time point label. For example, 1980, 1982, etc.
4. Attribute data type: Specifies the data type of the attribute to the program. It could be one of the following; Integer, float, string, and date.
5. Individual items: Each row corresponds to one individual data item. The item’s attribute values must correspond to the column. The item’s hierarchy can be entered at the end of the line after a blank column.

A sample treemap time series (tts) data file is given in Appendix A.

4.2.2 Data Structures

TreemapModel is an abstract data type which consists of some information about the global characteristics of the data set, such as the attribute types, attribute names. This class contains all the data manipulation and data filtering aspects of the tree as well as the data structures that facilitate data filtering and manipulation.

Data from the tts (See Appendix A) data file is read by an instance of java class TreemapTTSReader. The global information such as number of static attributes, time varying attributes, length of each time series attribute, attribute types, and attribute names are initialized after reading the first four lines in the data file.

Actual data items are read into an ArrayList of TreemapNode instances, one for each item in the data set. Each of these instances contains the name of the object, the attribute value object array for static attributes and a two dimensional array for dynamic variables to store the time series values for each dynamic attribute. The hierarchical path from the root to the node is initiated recursively in TTSReader itself.

New data structures are created to store actual data items and items in the hierarchy separately. All the numerical attributes are sorted with TreemapNodeComparator; the sorted order is used in filtering and binning. Series attribute values are sorted and the minimum and maximum values in each time period are stored. These values are later used for showing the data envelope.

Finally, model contains a flag for showing %changes. Initially it is set to false. When set to true, %changes are calculated on the fly and displayed in the binning widget.

4.2.3 Loading a data file

A data file can be loaded by selecting “Open” from the File menu. This action creates a `TimeSeriesFileFilter` from the `edu.umd.cs.treemap.filefilter` package and displays a list of `tts` and `tms` files in a `JFileChooser` window. The filename is used to create root of the tree, whose title can be changed later. The `TreemapTTSReader` reads through the data file line by line, initializes the global data structures after reading the metadata in the first few lines. The individual items are read into `TreemapNode` and a tree structure is created. When `TreemapTTSReader` finishes reading the items from the file, `TreemapModel.CommonConstructor2()` method calculates the aggregate values, sorts the numerical attribute values and sets the default size and color attributes. An instance of `Treemap` class draws the treemap constructed by its model. An instance of Java class `ControlPanel` creates the controls in a `JSplitPane`. Graphic overview and histogram overview are painted when a multi-valued attribute is selected for color coding treemap.

4.3 Graphical User Interface

The temporal treemaps graphical user interface consists of several Swing windows.

- `TreemapProgram` is a `JFrame` that acts as the main application window. It contains a menubar, a vertical and a horizontal split pane. It contains `Treemap` on its left top area, `AttributeTable` and `ControlPanel` on its right top area and `GraphPanel` and `HistogramPanel` on its left bottom.

- Treemap is JPanel which paints the data items as nested rectangles based on the layout algorithm and handles Mouse Listeners, KeyListeners, TreemapNodeClick and TreemapNodeCursor Listeners.
- AttributeTable is a JTable that displays item details on demand and implements TreemapNodeClick listener.
- ControlPanel is a JTabbedPane with the controls being distributed in five tabs: main, legend, filters, series and hierarchy.

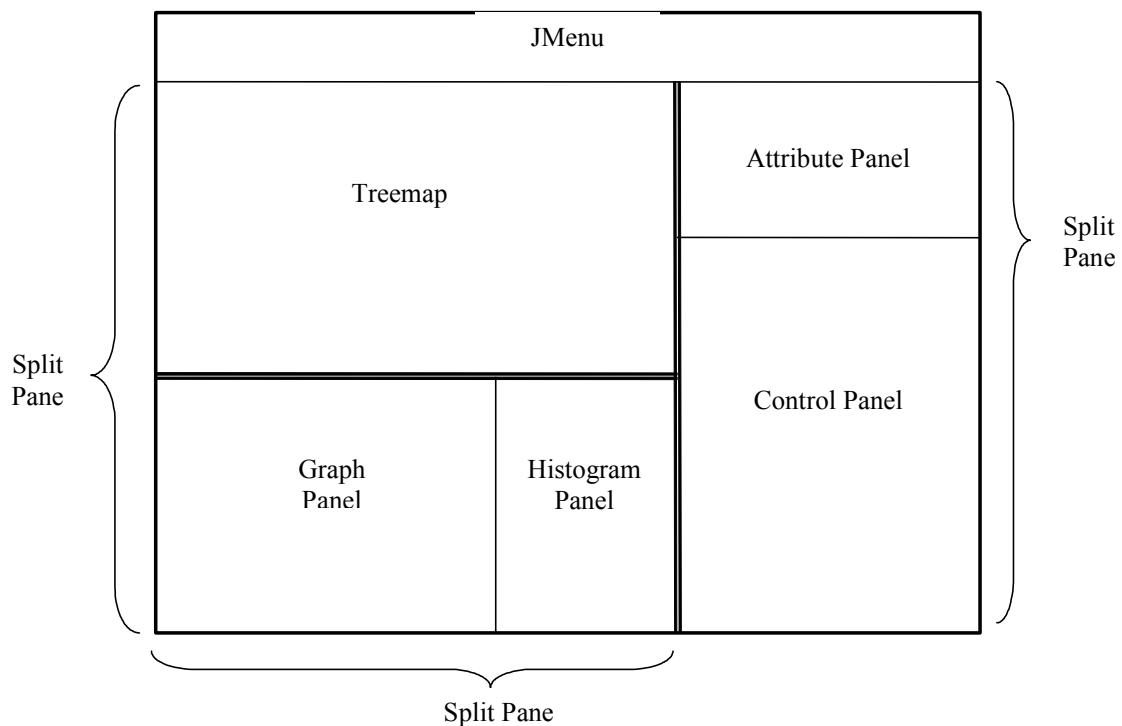


Figure 4.1A Schematic overview of classes in temporal treemaps.

- GrpahPanel is another JPanel with all line graphs.
- HistogramPanel is a JPanel with histogram display of data distribution in all time periods.

A schematic overview of the classes involved is given in Figure 4.1.

Typical treemaps have a treemap display and a split pane consisting of attribute table and control panel with four tabs (main, legend, filters and hierarchy). Graph panel and histogram panel are newly added in temporal treemaps.

4.3.1 Overview of Series Tab

Initially Series tab has a “Color” combo box with all the series or multi valued attributes, with “None” as selected index.

Upon selecting a series attribute from the combo box, the following swing components are displayed as shown in Figure 4.2.

- JSlider to navigate through time periods
- TMRangeSlider for selecting a range of time stamps and “Start” and “Reset” JButtons to animate the display with a “Delay” of specified milliseconds.
- JCheckBox to change from absolute value to relative changes mode.
- JComboBox with all the time periods, any one time period can be selected as a reference point.
- JComboBox with all the time periods.
- JCheckboxes for “increasing”, “decreasing” and “no change at all”
- Binning widget with missing data option.

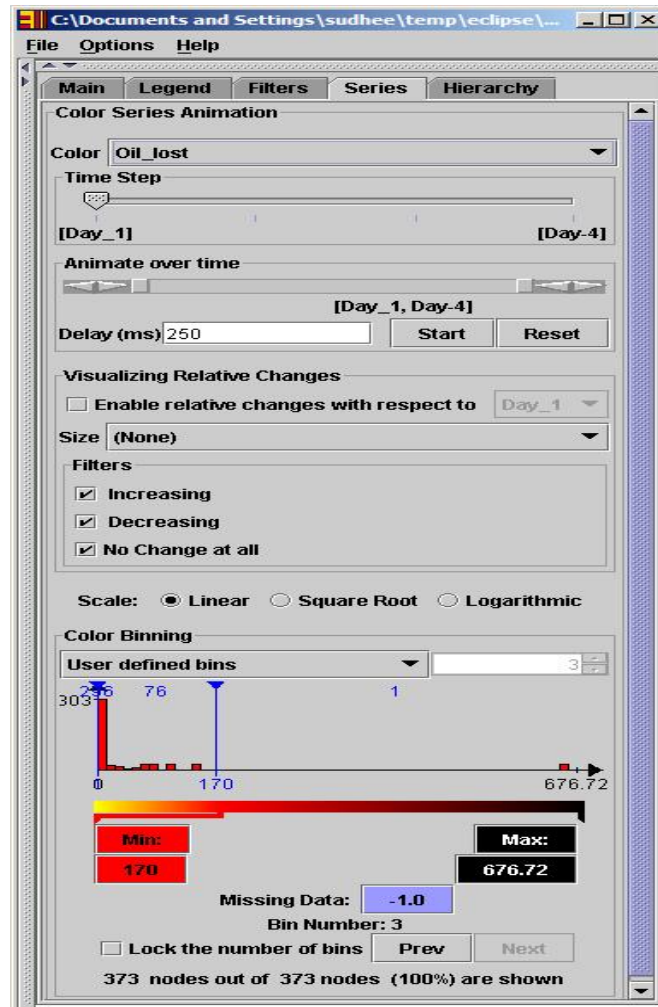


Figure 4.2 Controls in Series Tab

4.3.2 Display and Interaction handling

The display space consists of a JSplitPane which accommodates a Treemap and another JSplitPane consisting of GraphPanel and HistogramPanel. Treemap paints nodes, instances of TreemapNode in a recursive fashion based on the layout algorithm. GraphPanel displays the axis. When a time series attribute is selected, the axis, labels on the axis, and lines that plot the values of all items are drawn with Graph.PaintComponent method.

The display window has mouse event handlers and event listeners which facilitate the interaction between different views. Mouse click event in Treemap registers a `NodeClickListener` and implements `nodeClick()` method in `AttributeTable` and `GraphPanel`. The `GraphPanel.nodeClick` method draws the line plot of the node on which mouse click occurred. Mouse click event is handled in three different ways in Treemap, a single left mouse click would highlight the path, double click would zoom the display and CTRL+click would select the node by setting flag to true. This action draws graphs of all selected nodes.

The mouse Click event on an internal node (node in the hierarchical structure) would retrieve all its leaves or all data items under that sub tree and display the time series graphs of these items only.

The mouse motion event handler captures the current position of the mouse, computes the `TreemapNode` at that position with `findNodeContainingPoint()` method. If this is a new item under the cursor, then `NodeCursorListener` interface implements `nodeCursor`. The `nodeCursor` method in Treemap displays the popup and `nodeCursor` method in `GraphPanel` displays the popup in the graph overview.

The Java class `Graph` implements two Mouse listeners which register mouse motion with `LineCursorListener` and mouse click event with `LineClickListener`. Mouse over on the time series graphs in `GraphPanel` would highlight the line graph, show the item name, nearest (x,y) ordered pair, highlight the corresponding `TreemapNode` in Treemap, show the popup and update the detail table.

4.3.3 Coupling treemap with time series graphs

Tight coupling between treemap and time series graphs is achieved through coordinated highlighting, updating textual labels, and the details in the detail on demand table. When the user clicks on a treemap node, `Treemap.nodeCursor` method highlights that particular node and `GraphPanel.nodeCursor` highlights the corresponding time series graph.

The java class `Graph` uses a global variable `LineUnderCursor` to store the corresponding time series graph index. Each line has an index which is equal to the order in which they are painted initially. A hash table `idToIndexHash` stores the treemap node id and time series index entries. `GraphPanel.nodeCursor` passed the node id as an argument to `Graph.setNodeCursor` method. This method checks whether that node's time series graph exists or not. If it does not exist it returns null. If it exists, it sets `LineUnderCursor` and paints the pop-up at the end of the time series graph.

Similarly, when the users do a mouse over in the time series graph overview, the nearest time series graph and the corresponding treemap node is highlighted. `Graph` uses three arrays of hash tables: `indexToVal[numTimeSteps]` – stores an index and series attribute value for each time period; `indexToNodeId[numTimeSteps]` – stores the same index and node id for each time period; `valToPixel[numTimeSteps]` – stores the series attribute value and the corresponding pixel value on the panel for each time period. When the users move the mouse, the x,y coordinates are captured and the two time periods t_1 , t_2 are determined so that the point $t_1 \leq x \leq t_2$. Determining t_1 and

t2 limits the search to the y-values in these two time periods. These y-values are retrieved from the valToPixel [t1] and valToPixel[t2] hash tables. Then a linear search is performed in the key set of these two hash tables and the index of the key for which the difference in the pixels is minimum is returned. From this index, the node and its name are determined using the indexToNodeId hash table. Once the corresponding treemap node is identified, it is highlighted, popups are displayed and the detail on demand table is updated. If the user clicks on this line, then only that time series graph is displayed.

4.4 Performance

Performance of temporal treemaps depends on the number of data items, number of data attributes, number of hierarchy levels, and number of time periods. Treemaps can handle datasets consisting of up to 20,000 data items, 10 data attributes and with 18 levels of hierarchy. Much larger data sets can be handled by increasing the memory allocated to run treemaps. This can be done thru command line using the option `<-Xmx[v]M>`. But for such large data sets, it takes several seconds to update the screen when data items are filtered or hierarchy is changed.

Based on the available data sets, temporal treemaps can handle up to 20 time periods, a few thousands of data items with fast interaction ($\ll 1s$). It is observed that for the HCIL web logs data consisting of 3300 data items, 7 attributes, 12 hierarchy levels, and four time periods, the interaction is sluggish (1–2s). The time taken to read this data file is approximately 1-2s, but to load the settings it took 3-4s. The bottleneck in the performance is in the method that implements coupling of the

treemap with the time series graph. Also, as the number of time periods increase (>20), the slider will become cluttered with the overlapping ticks and labels. This would also have an effect on the performance as the array size will be increased and performing linear search in array will increase the time to retrieve the data.

The input data sets are stored in the memory as simple text tab delimited files. For each time period, the data is retrieved from an array of values and set to the corresponding nodes and then corresponding colors are applied. For visualizing percentage changes, the values are calculated each time when the time period is changed. An alternative would be to have a buffer of multiple preprocessed (with all settings applied) treemaps in the main memory and load the corresponding treemap for each time period. In this case performance depends on how fast the treemap can be accessed from the main memory, but once it is displayed the interaction can be guaranteed to be fast.

Chapter 5

Case studies

In this section five case studies are presented to demonstrate the use of temporal treemaps.

5.1 NCHS Death Statistics

The National Center for Health Statistics (NCHS) publishes death statistics which consists of number of deaths due to 43 different diseases collected over 18 years (1981 – 1998), per 100,000 people of age 65 plus. The hierarchical structure shows the disease categories. The main goal of visualizing this data is to identify the disease trends.

The original data consisted of statistical information of the number of deaths due to each disease and the total number of deaths in each disease category. The latter was greater than the sum of all diseases in that category. The difference is shown in treemap by imposing a new disease/disease category “other” at each level to indicate deaths due to unidentified diseases. For example, *Other diseases of heart* in the *Diseases of heart* category represents people who died of unidentified heart diseases.

Figure 5.1.1 shows an abstract visualization of 43 causes of deaths grouped by disease categories. The hierarchical structure in Figure 5.1.1 exemplifies pre-defined variable depth hierarchy. Each rectangle represents a disease; size of the rectangle is proportional to the number of deaths due to that disease in 1998 and color is

proportional to the number of deaths in 1981, with light to dark shade showing a minimum to maximum number of deaths.

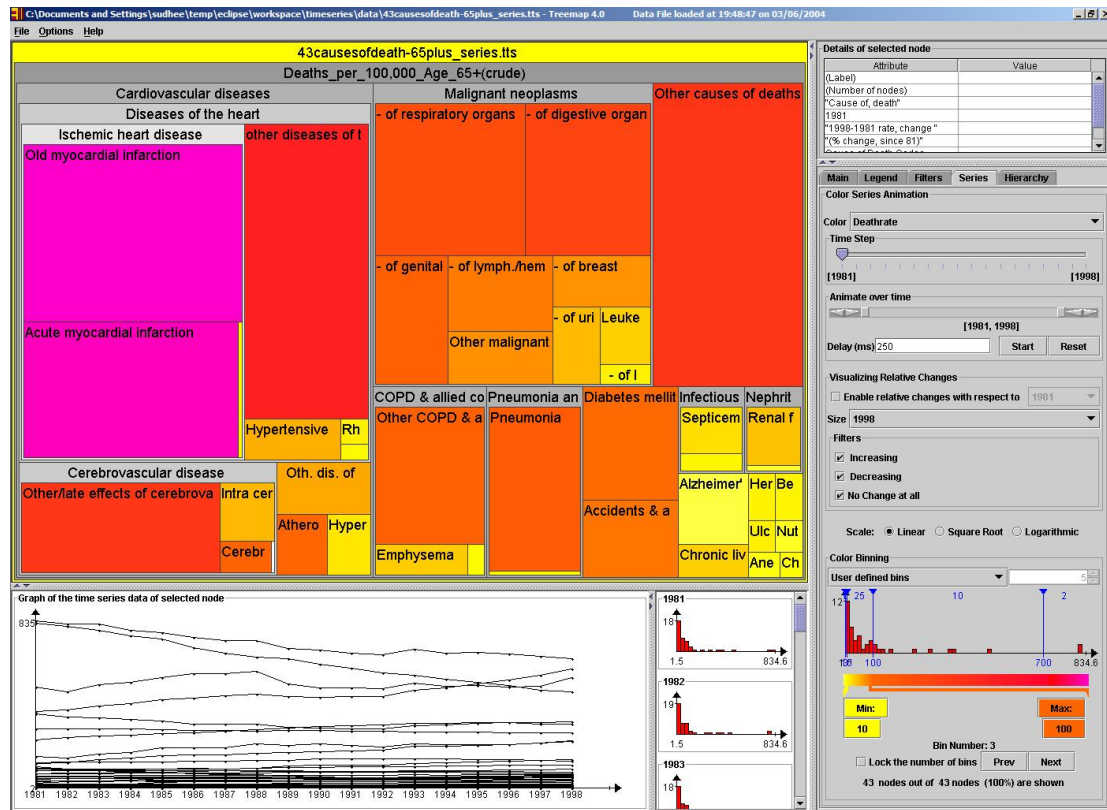


Figure 5.1.1 Visualization of 43 causes of death statistics over 18 years grouped by disease category. Each rectangle represents a cause of death with size proportional to number of deaths in 1998 and a shade of white –yellow-purple showing increasing number of deaths in 1981.

Users can see that *Old myocardial infarction* and *Acute myocardial infarction* are two major causes of deaths in 1998 (large rectangles) as well in 1981 (dark rectangles).

Figure 5.1.2 shows the number of deaths in 1998. Users can see that *Old myocardial infarction* and *Acute myocardial infarction* are still the major causes of deaths though the number of deaths due to these diseases having decreased as inferred from the color change (purple to red).

Figure 5.1.2 Visualization of same 43 causes of deaths statistics in 1998.

Alzheimers disease, Other (non-pneumonia) and Other causes of deaths. For such changes the time series graph display helps users better understand the trends (Figure 5.1.3).

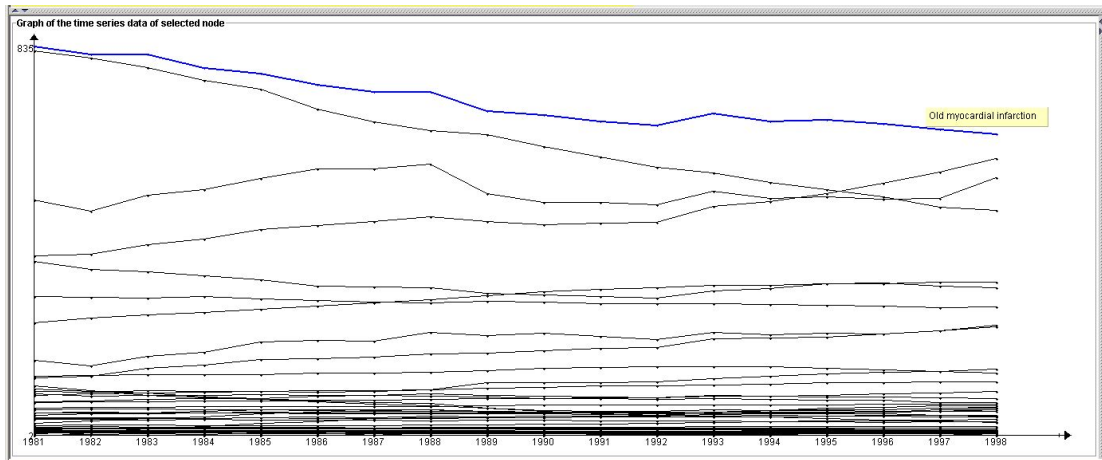


Figure 5.1.3 Time series graph overview of death statistics of 43 diseases.

From the time series graph overview in Figure 5.1.3 users can see that there are four diseases with large number of deaths (top 4 time series graphs). Users can observe that *Old myocardial infarction* (highlighted in blue with the name shown in popup) is the major cause of death and has been the major cause for all the eighteen years. Most of the disease trends are monotonous. There are about 30 diseases for which the number of deaths is less than 100 (cluttered time series graphs at the bottom of Figure 5.1.3).

Figure 5.1.4 shows the visualization of same disease statistics with color of the rectangle proportional to the %change in death rates from 1981 to 1998. A shade of green indicates %decrease in the death rate and shade of red indicates %increase in death rates from 1981 to 1998.

Users can see that *Alzheimer's* disease has increased by 1085%. This might be because *Alzheimer's* disease has been diagnosed recently and the cause of such deaths in earlier years was not classified as *Alzheimer's*.

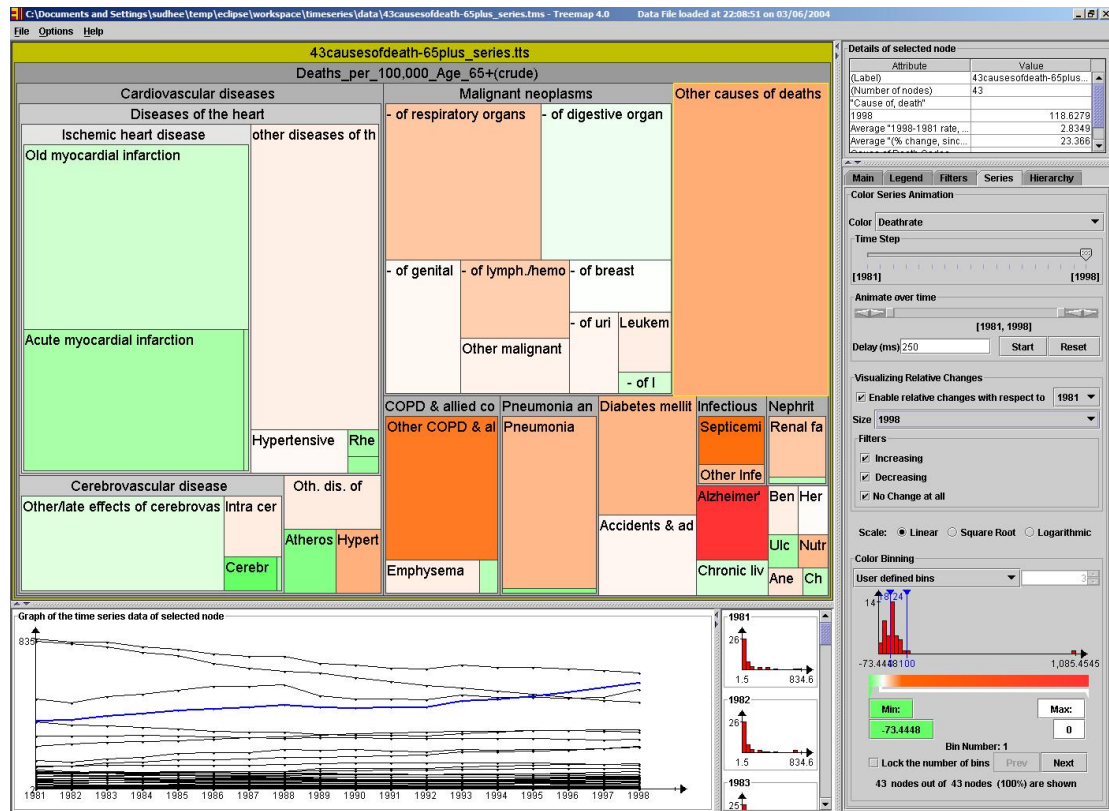


Figure 5.1.4 Visualization of relative changes in death rates in 1998 with respect to 1981, grouped by disease categories. Each rectangle represents a disease; size is proportional to number of deaths in 1998 and color is proportional to % change in number of deaths from 1981 to 1998, with green-white color showing % decrease and white –red showing %increase in death rate.

There is a significant drop in death rates due to heart diseases except *Hypertension* whose death rate has increased by 50%. The death rate due to

Malignant neoplasms does not show much variation relative to 1981, fluctuated by about 5 to 10%.

Visualizing the %changes in 1998 with respect to 1990 showed an increase in death rates for about 23 causes of deaths and a decrease in death rate for about 18 causes of death. Death rates are increasing at a higher rate till 1990 compared to the death rates from 1990 to 1998.

5.2 HCIL Web logs

HCIL web log data consists of the HCIL (www.cs.umd.edu/hcil) web statistics like *page name*, *user id*, *group id*, *modification time*, *creation time*, *size* and *hitcount*. These web logs are collected over four weeks, labeled as *Week_1*, *Week_2*, *Week_3*, and *Week_4*. The time series attribute is *hitcount*.

From these given four log files, a single time series dataset is generated using a routine known as TTSGenerator (See Appendix c). Users can specify a missing value to fill in the data value for missing pages (pages that don't exist during some time period).

Figure 5.2.1 shows the treemap visualization of web directory structure during *Week_1*. Users can see that there are a total of 3293 files in the HCIL, 4% of which have a size of 0 bytes. It might be because either the file content is zero or the access is restricted. This file directory tree is 9 levels deep and the deepest sub-tree is *hcil/members/arose/gridl/edu*.



Figure 5.2.1 Visualization of HCIL web log during week_1, grouped by directory structure. Each rectangle represents a file/web page, size proportional to file size in bytes, and color proportional to hitcount, a color shade of white -yellow- purple represents 0 to 4298 hitcount.

The largest directory in HCIL is *millionVis* (largest rectangle), the next large directories are *kiddesign* and *datelens*, and the largest file is *UIST 2002 – fishcal.mpg(55MB)* in *hcil/datelens* directory and the other big file is *precslider.gif(35MB)* in *hcil/millionvis*.

Figure 5.2.2 shows the temporal visualization of HCIL web hitcounts during week_1. Users can see that few html files in *Jazz-chat* and *iv03contest* directory does not exist in week_1.



Figure 5.2.2 Visualization of HCIL web logs during week_1, grouped by directory structure. Each rectangle represents a web page and color represents hitcount. White color shows missing pages, yellow – purple gradient shows 0 to 4300 hits. The highest hitcount is registered for *banner-images* (from the dark color). *Jazz-chat* has 0 to 1 hits during week_1. *Kiddesign*, *Photolib* and *Treemaps* were popular with more than 100 hits. In *hcil/jazz*, 100 to 400 hits have been registered for

hcil/jazz/img/nav files, the *index.shtml* files, and the few image files in *jazz/applications*.

In the *hcil/pubs* directory a total of 1254 hits have been registered and *index.html* was visited 124 times. The *tech-reports.html* has 397 hits and *index.html* and *products.shtml* have more than 100 hits. The files in *hcil/pubs/books* have a wide range of hit count from 0 to 100. All the files in *hcil/pubs/screenshots* directory have been visited an average of 10 times.

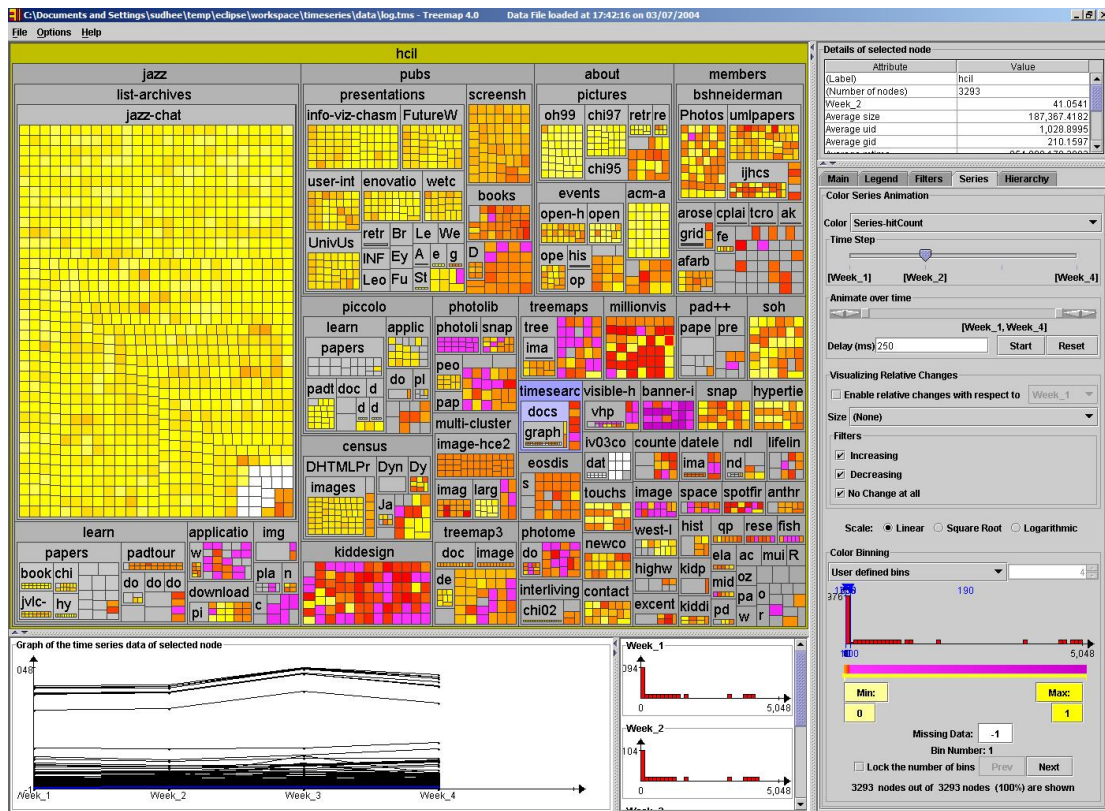


Figure 5.2.3 Visualizing HCIL web logs during week_2.

The *hcil/members/* gif images are visited an average of 30 to 300 times among which *ben-formal.gif* has the highest hitcount of 371 hits. HCIL faculty and research

staff images have been visited 80 to 400 times, and the student images have been visited 1 to 40 times. *Index.html* or *index.shtml* pages in all directories are visited more than 100 times, showing that HCIL is popular

Figures 5.2.3 – 5.2.5 show abstract visual representations of HCIL web logs during Week_2 to Week_4 respectively. The overall activities in these visualizations are similar to the one in Figure 5.2.2 with some minor changes. *Kiddesign* and *photolib* are still popular applications. *Iv03contest* directory doesn't exist in week_2.

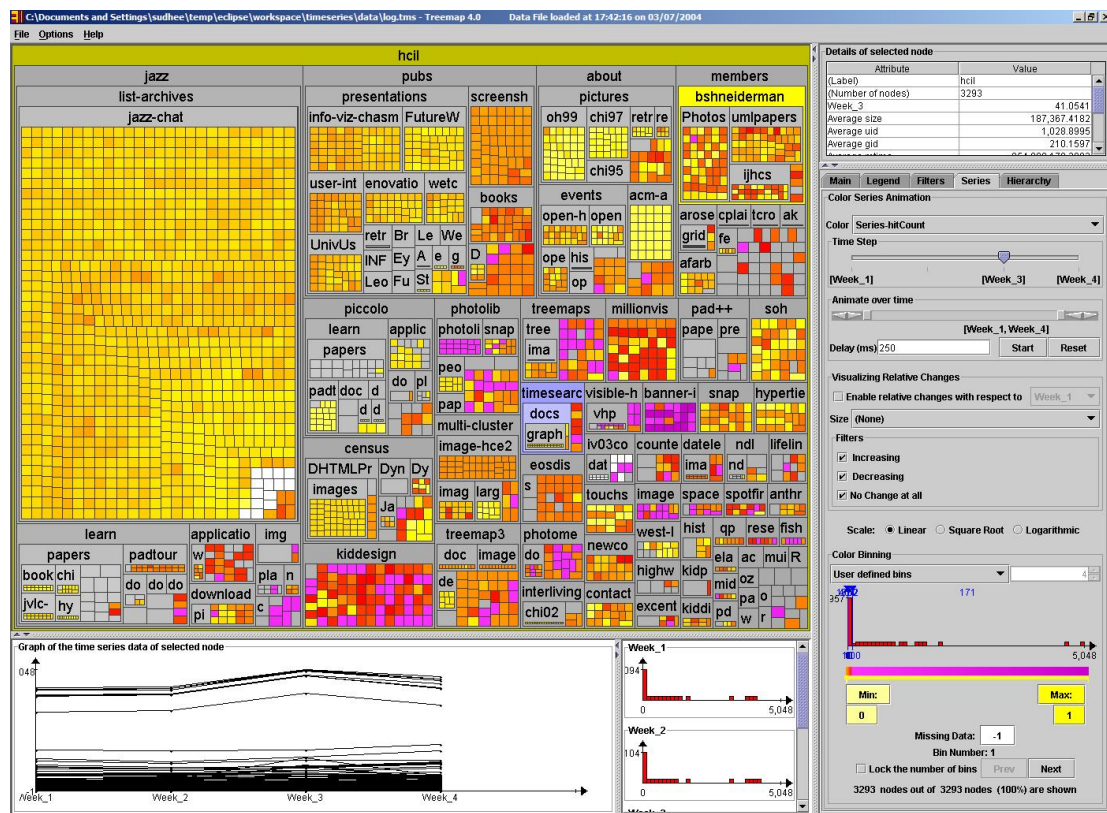


Figure 5.2.4 Visualization of HCIL web logs during Week_3.

Jazzchat was busy in week_3. A directory *hcil/iv03Contest* is created with three files, and these files had 190 to 220 hits. There were 0 hits for *acm-award-pics*.

Since the HCIL web logs data is large (3300 files) and is deep (9 levels), most of the screen space is lost for showing directory structure. While monitoring page hitcounts over time, users are interested in visualizing the changes in different areas rather than the details. Users can reclaim more pixels by removing the borders as shown in Figure 5.2.6.

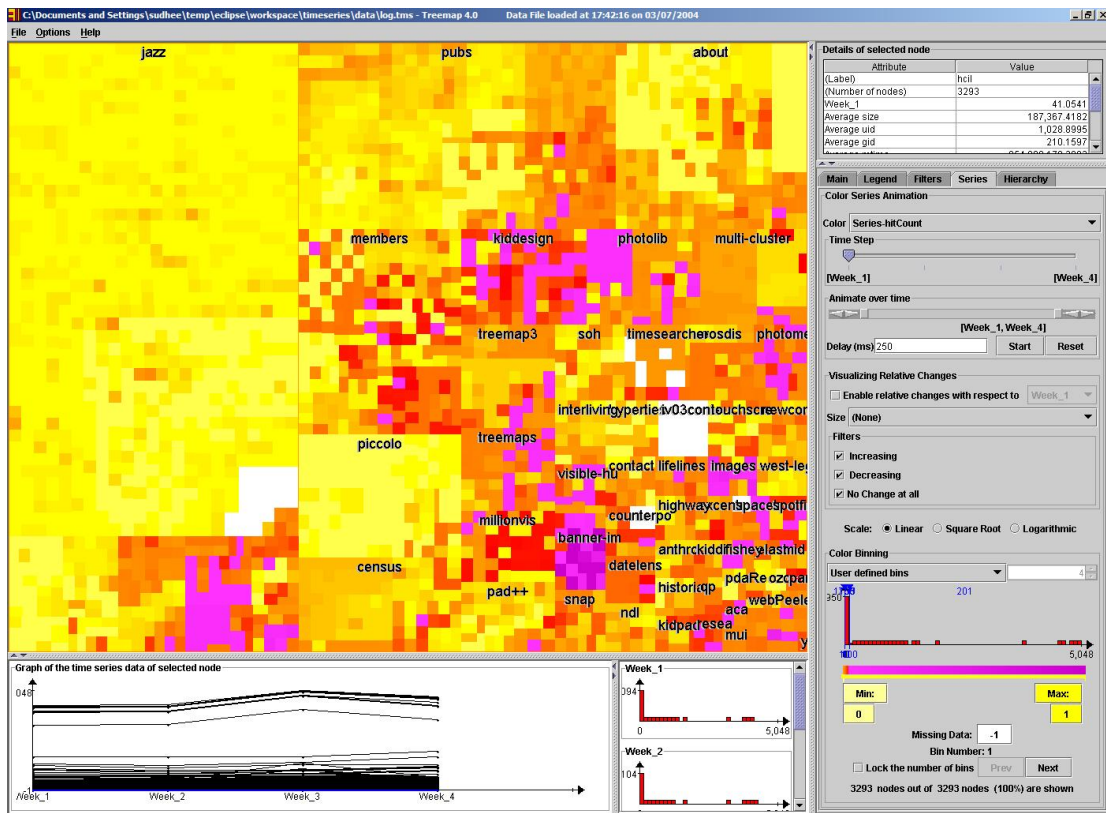


Figure 5.2.6 Visualization of HCIL web logs in Week_2 without borders and with overlaid labels.

Users can remove the borders and navigate through time periods focusing on the color changes over the treemap. Figure 5.2.6 overlaid labels of directories at level 4 in the directory structure. This label overlay helps the users retain context. Users can

switch between removing and adding borders depending on their needs. Users can identify the dark colored areas to be most popular and the white color rectangles represent the files that were deleted.

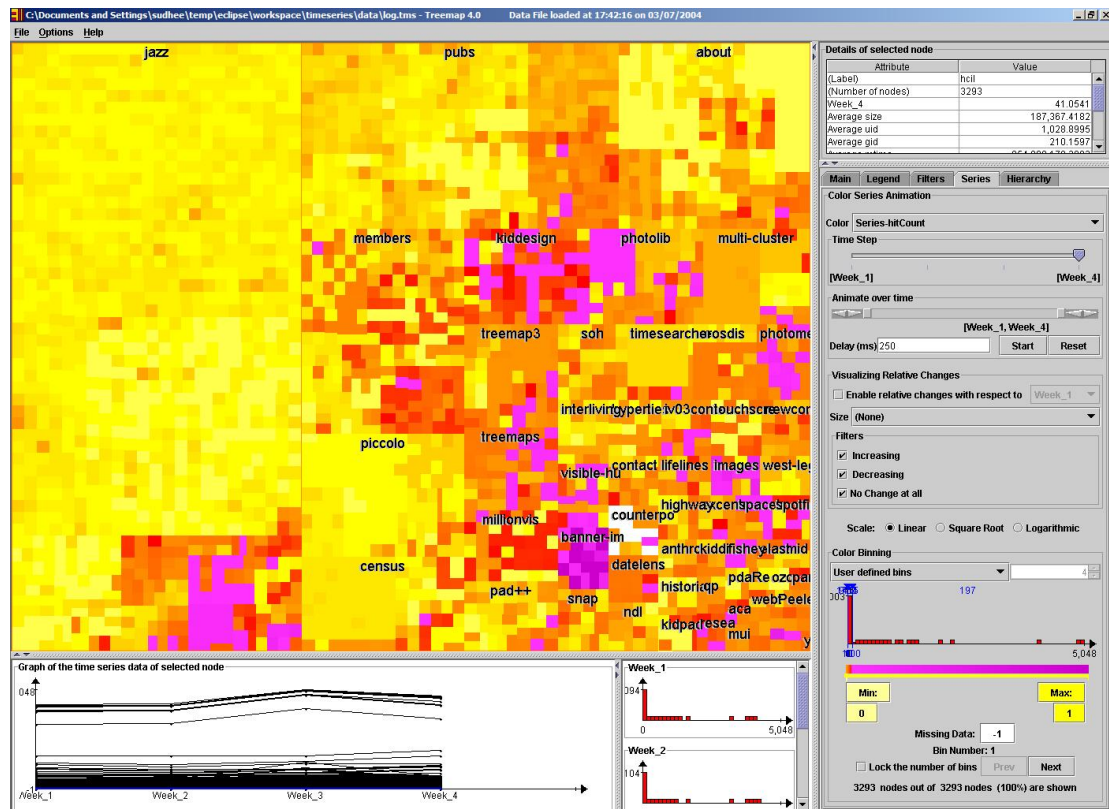


Figure 5.2.7 Visualization of HCIL web logs in week_4 without borders.

From the graphic overview, users can see that banner images had highest hitcount in all time periods and this hitcount was very high compared to all other files.

Analysis:

This data is published by IV03 Contest (www.cs.umd.edu/hcil/iv03contest/) with intent to evaluate visualization tools based on some data dependent tasks. Analyzing

this data in temporal treemaps has resulted in the implementation of TTSGenerator, missing data, overlay labeling, and search option.

The original data was large consisting of 70,000 leaf nodes. Since treemap (Treemap 4.0) cannot handle (interaction was too slow) such a large data, a subset of the data (everything under HCIL subtree) is exported into a new treemap data set for all the four logs. Then the issue was to convert the regular treemap data set to treemap time series format. To create a super set of all the four logs in time series format, a new routine TTSGenerator was developed. TTSGenerator is implemented by Aleks Aris, a graduate student in HCIL.

Some pages were newly created and some existing pages were deleted resulting in missing data. The design decision to use a supplement value of “-1” for page hitcounts was made. This value was shown in the data file in the first row, second column of the series attribute. After the data file was created, the problem was to identify the missing pages in the data set, which led to the missing data color. This helped to identify the newly created pages and the deleted pages.

One of the tasks was to find the gif images in the directory. In an effort to answer the question, search option was added to filters. Search in the label field was implemented first which was then improved to search in any attribute.

Since the directory structure is deep (10 levels), much space is used for drawing the hierarchy levels compared to the actual data items. While viewing the changes over time it was really hard to visualize the color changes, as the leaf nodes were hardly visible. Viewing the changes without borders helped to identify the popular

areas in the treemap, but the context information is lost. Hence overlay labels are used to show the labels at any selected hierarchy level to retain the context information.

5.3 Daily Oil Production

Temporal treemaps can be used to monitor daily production from oil and gas wells. Data is collected from oil wells and reported each day.

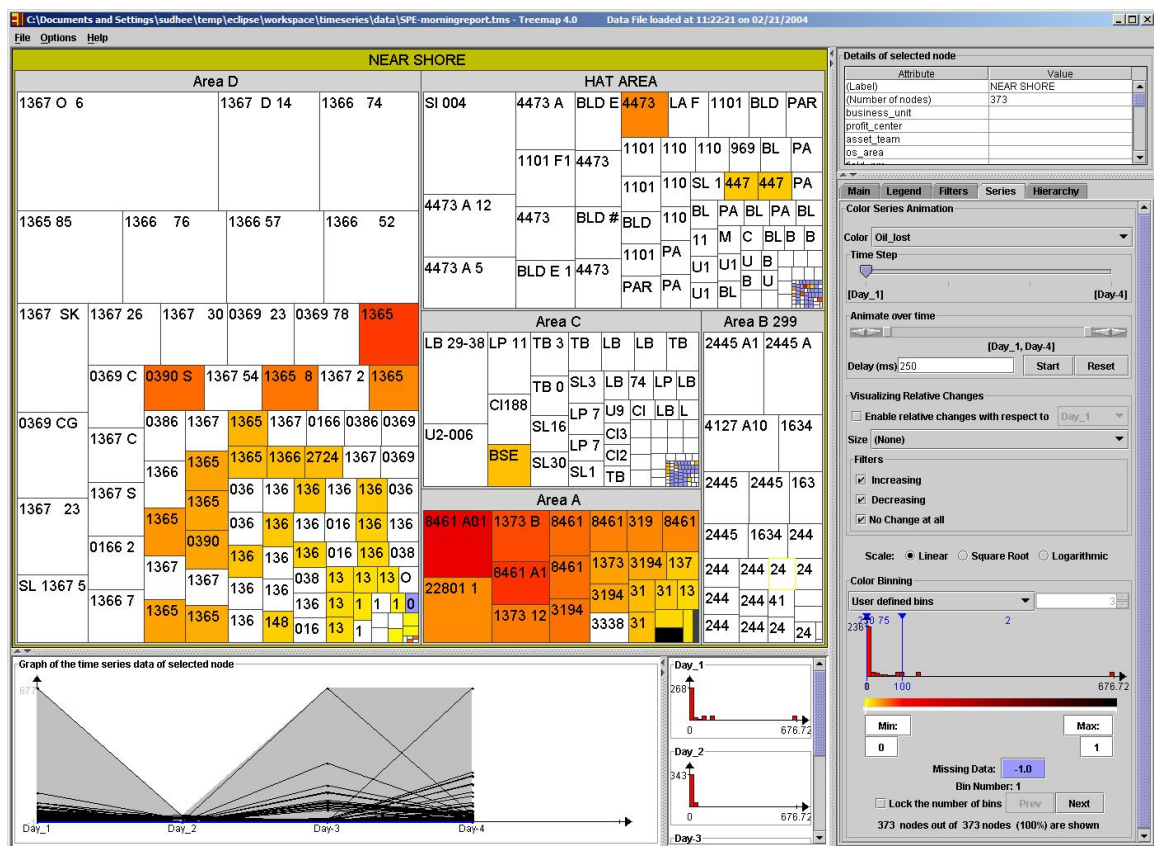


Figure 5.3.1 Monitoring Oil well data on Day_1, grouped by their geographical region/location. Each rectangle represents an oil well, size proportional to amount of oil produced and color proportional to lost production, white represent zero oil lost, shade of yellow to black is proportional to maximum oil lost. Light gray color represents oil wells that did not report data.

This data consists of the amount (barrels) of oil produced in each oil well, amount of oil lost in production (less amount of oil is produced than expected), amount of gas produced, delta oil lost, delta gas lost etc. The main goal of monitoring this production data is to observe the amount of oil lost in production each day for four consecutive days

The Treemap shown in Figure 5.3.1 summarizes the production of 373 oil wells grouped by geographical region/location on Day_1. Each box in the Treemap (Figure 5.3.1) represents a well. The size of the rectangle is proportional to number of barrels of oil produced in a well and the color indicates “lost” production – the difference between the actual and expected barrels produced.

Figure 5.3.1 shows at a glance which of the wells are the largest producers (i.e. the large rectangles), and the color indicates where the oil is lost in production. In this data some of the oil wells did not report the production statistics due to technical problems. This missing data from these oil wells is replaced with a “-1” and is identified with a different color (light gray/blue in Figure 5.3.1).

Area A is the problem area as almost all the wells in that area, have reported oil loss. Area D has few oil wells, which have lost 175 units of oil. Oil wells in Area B 299 did not report any oil loss.

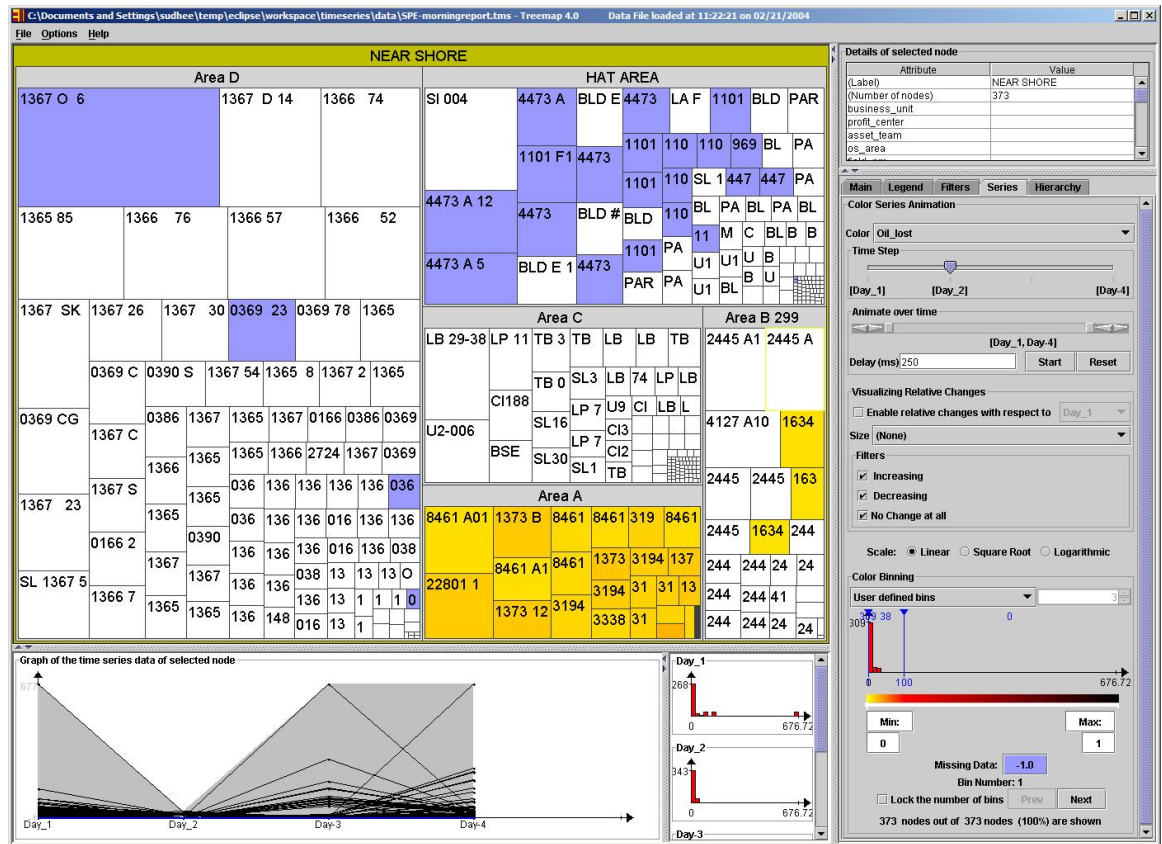


Figure 5.3.2 Visualizing same oil well data on Day_2

Figure 5.3.2 shows the oil production data on day _2. Area A has been reporting oil loss for two continuous days. Area C is doing well, all the oil wells have produced the estimated oil to be produced. Area B 299 has 3 oil wells, which have lost about 10 barrels of oil. Half of the oil wells in HAT AREA did not report. The largest rectangle in Area D did not report the data (inferred from the gray color).

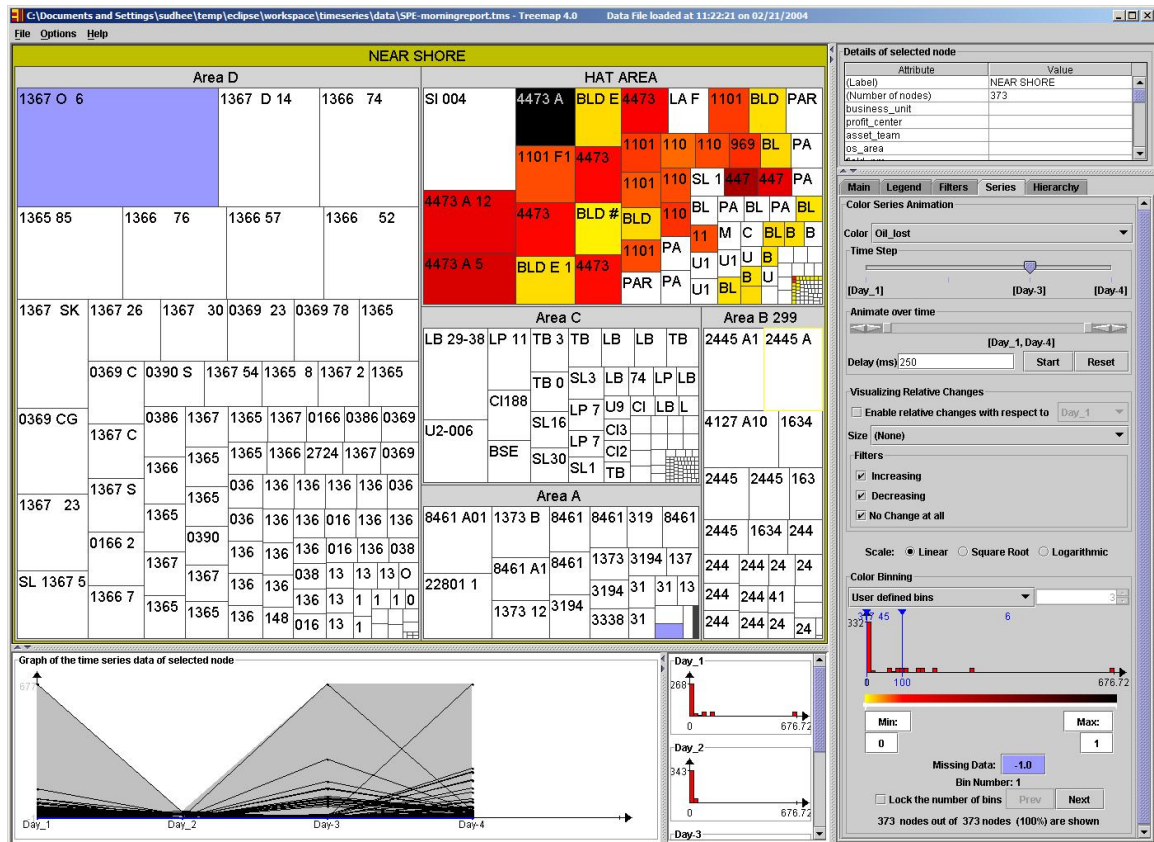


Figure 5.3.3 Visualizing same oil well data on Day_3

Data has not received from the largest oil well on Day_3 as well. On Day_3, oil loss is confined to HAT AREA only. Remaining oil wells did not report any oil loss. HAT AREA has reported the maximum oil loss and almost half the oil wells in this area have lost oil on day 3.

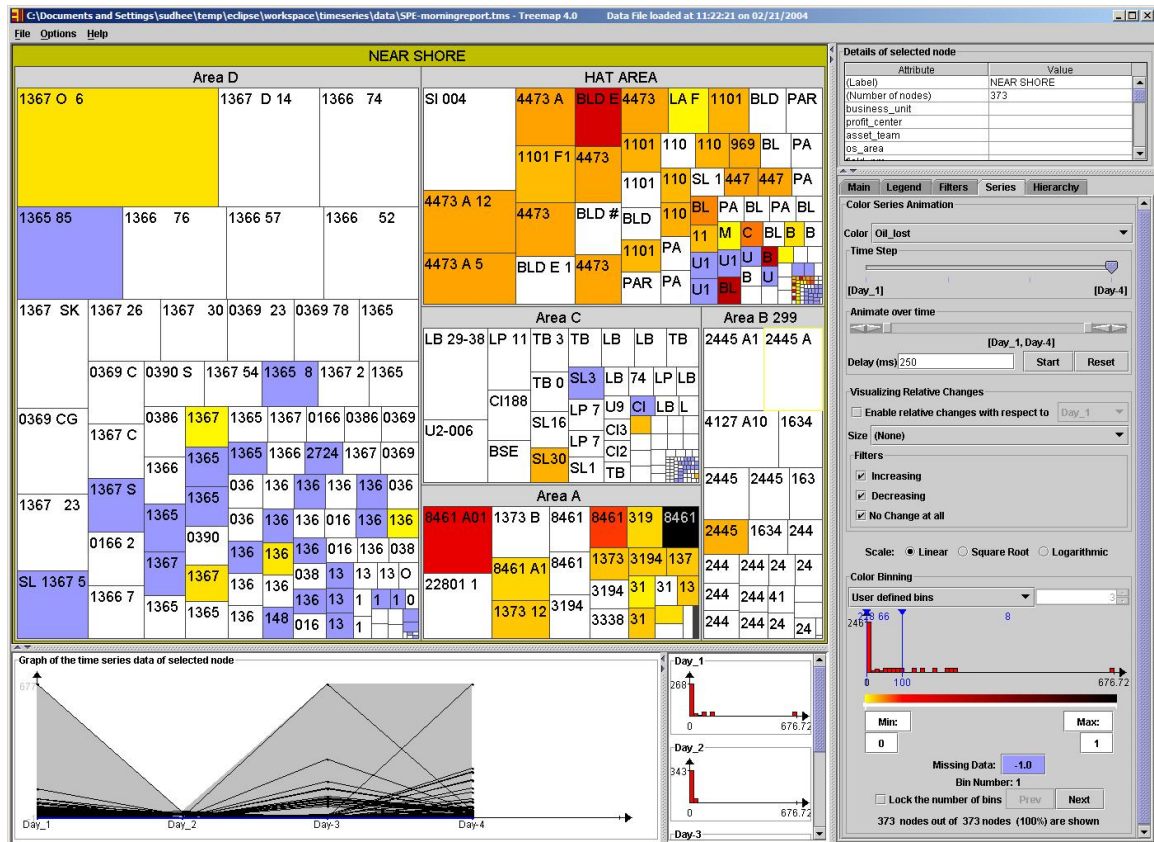


Figure 5.3.4 Visualizing same Oil well data on Day_4

Figure 5.3.4 shows the oil loss in different areas on day_4. The largest oil well has reported oil loss after two continuous days of missing data. Few oil wells in Area D did not report data. HAT AREA continues to be the problem area from day 3 to day 4 with most of wells losing oil. Data was not received from a few oil wells in HAT AREA. Area A is also the problem area since the maximum amount of oil is lost in the oil wells in this area. Area B 299 is doing well except for one oil well which has been losing oil constantly.

From the histogram display and the envelope, it can be observed that the maximum amount of oil loss occurred on days 1, 3 and 4, but not on day 2. On day 3 and day 4, there are few wells, which have lost half the amount of maximum oil loss.

5.4 NCHS Live births data

Live births data is provided by March of Dimes perinatal data center published in NCHS natality file. Live births consists of number of live births, percentage of all live births that were singletons, percentage of all live births that were multiples, percentage of all live births that were born preterm, percentage of singleton live births that were born preterm, percentage of multiple live births that were born preterm, and percentage of live births that were born preterm in whites, blacks, Hispanics, native Americans, and Asians. This data is published by NCHS and is collected for every state from 1990 to 2001.

This case study demonstrates the utility of temporal treemaps for identifying the trends in percentage live births that were preterm as a whole and in each race.

Figure 5.4.1 shows an abstract graphical visualization of percentage of all live births that were born preterm in 1990. Each rectangle represents a state in US and are grouped by geographical regions such as 1-west, 2-Rockies, etc. There is no size coding. The color of the rectangle (shade of green) is proportional to the %preterms in all live births.

Users can infer that District of Columbia has highest %preterm births from its dark color. The time series graph overview in the lower left corner shows that District of Columbia (top line graph) has the highest %preterm births for all the years.

Louisiana and Mississippi have the next highest %preterm births. 5-Central has highest average %preterm births (from the dark shade in that region).

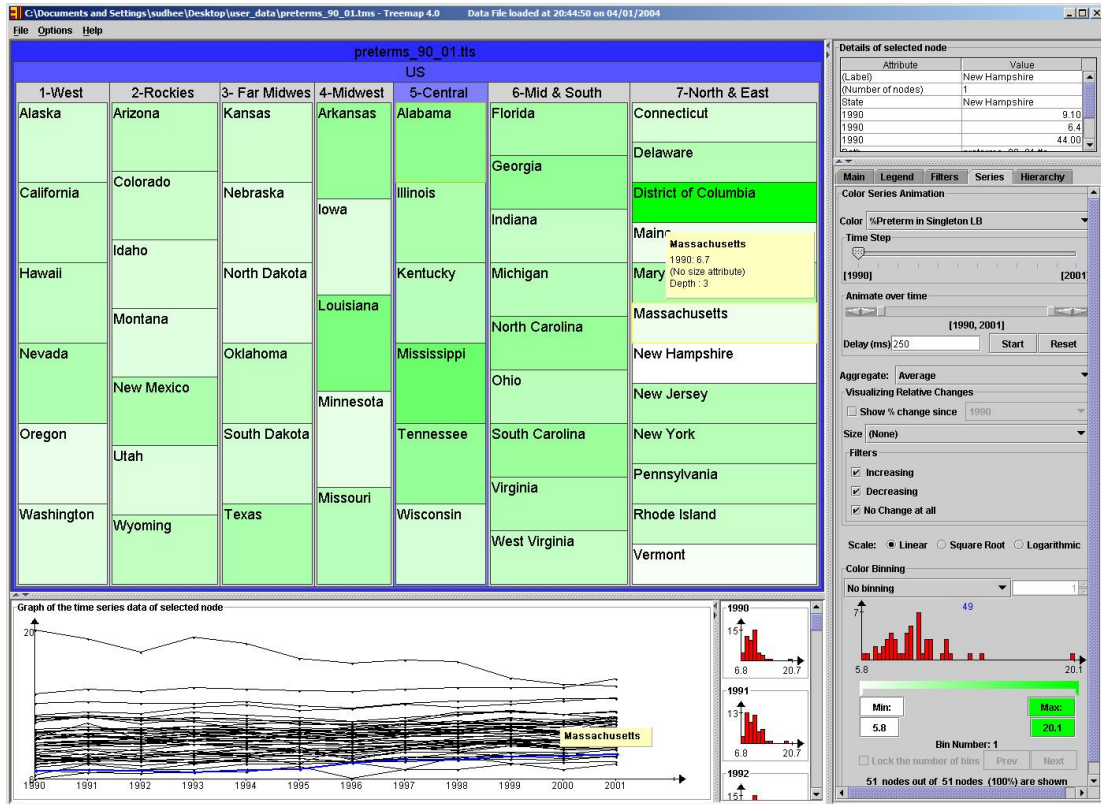


Figure 5.4.1 Visualizing percentage of live births that were born preterm in 1990.

Each rectangle is a state; grouped by geographical region; layout is slice and dice;
colored by %preterm in live births.

West, Rockies, and Far MidWest (light shade in these regions) have lower %preterm compared to MidWest, Central, and MidSouth regions (dark shade in these regions).

Figure 5.4.2 shows the same preterm birth data with color showing %preterm births in 2001.

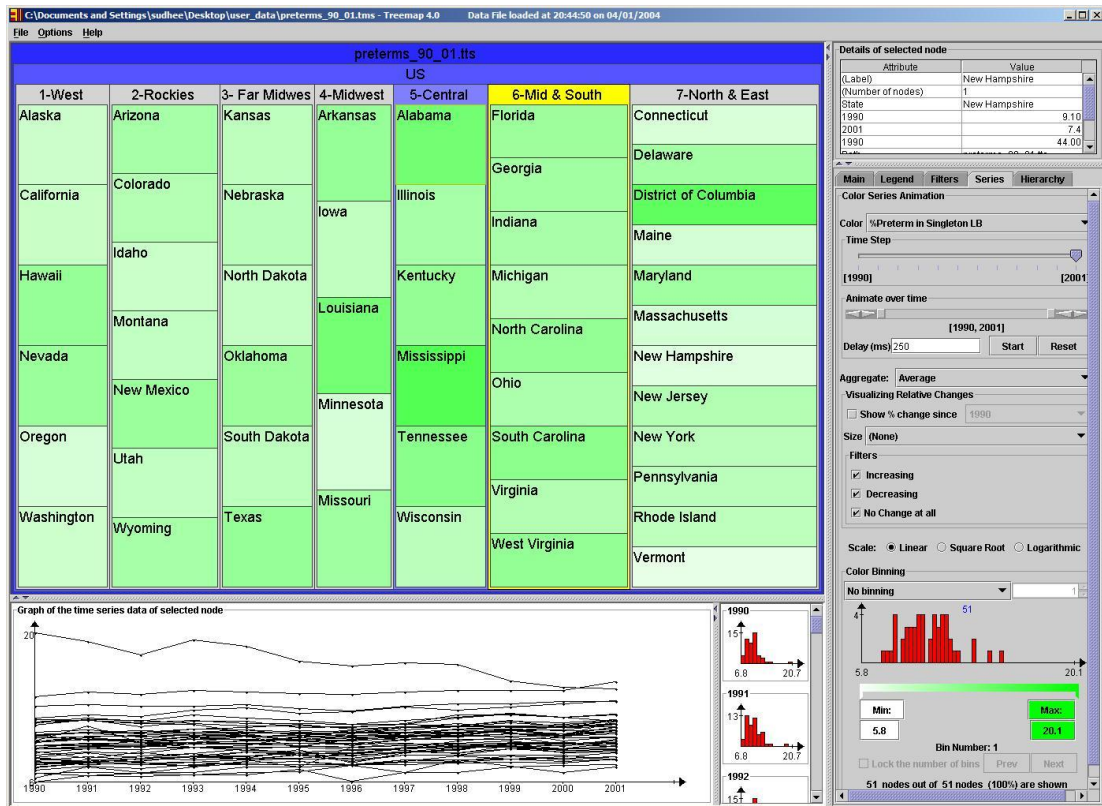


Figure 5.4.2 Visualizing %preterm births for each state in 2001.

The percentage of live births that were born preterm is increasing from 1990 (light shade in Figure 5.4.1) to 2001 (dark shade in Figure 5.4.2). District of Columbia and Mississippi has the highest %preterm births in 2001 as well.

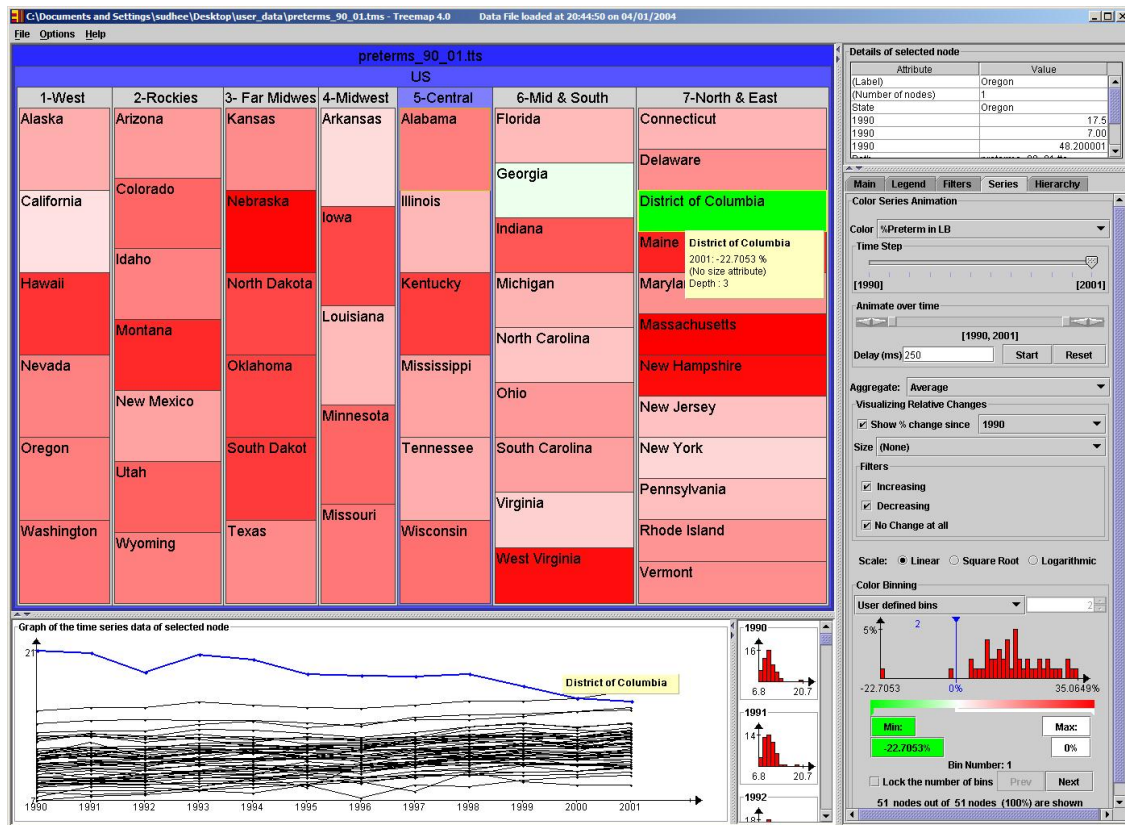


Figure 5.4.3 Visualizing %changes in live births that were born preterm from 1990 to 2001.

Figure 5.4.3 shows the %changes in %preterm births from 1990 to 2001. A shade of red indicates %increase in preterm birth rate and shade of green indicates %increase. District of Columbia is the only state that has showed a decrease in preterm birth rate and for every other state preterm birth rate is increasing. Massachusetts, New Hampshire, and West Virginia showed an increase in preterm birth rate by 35%.

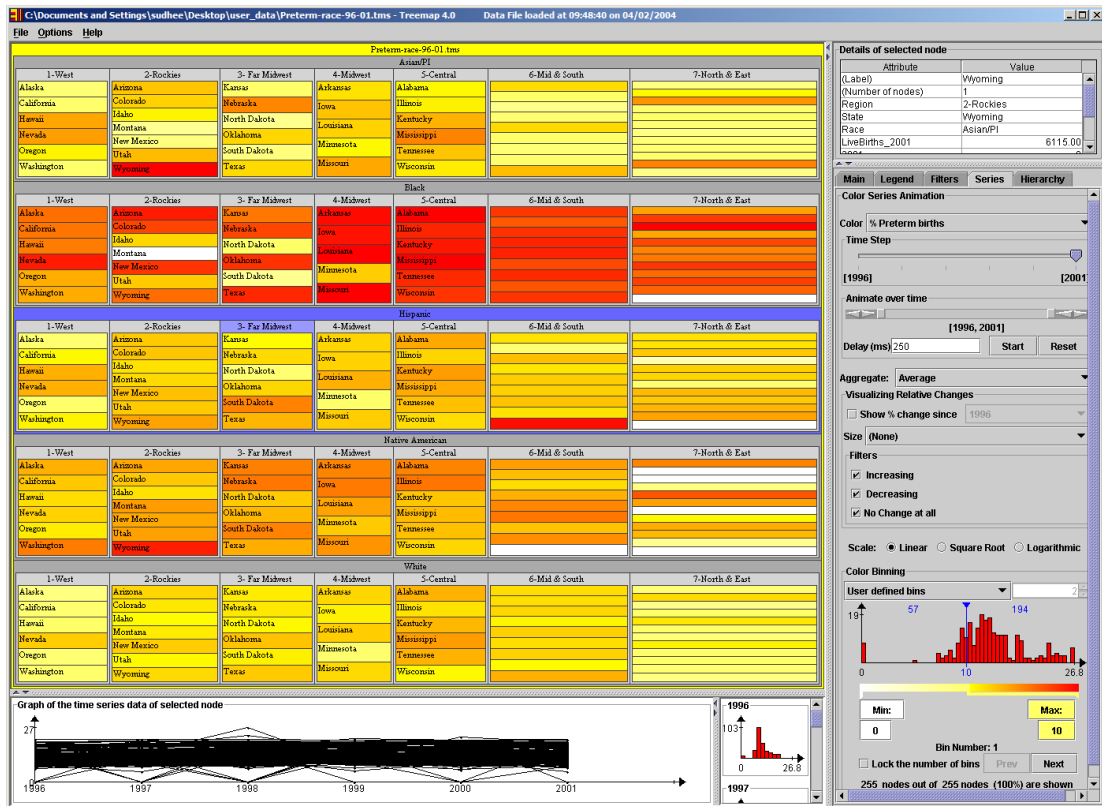


Figure 5.4.4 Visualizing %preterm in live births in 2001. Each rectangle is a state, grouped by geographical region and race. Color shows the %preterm in live births in 2001. A shade of yellow shows %preterm births less than 10% and shade of red shows 10% to 36%.

Figure 5.4.4 shows the %preterm birth for each state and race. Users can see that blacks have highest %preterm births and the highest %preterm birth rate is observed in Central region.

5.5 Monitoring monthly help desk tickets

This case study explains how temporal treemaps were modified to handle categorical attribute changes over time. The IT department in Chevron Texaco at

Area is also used for color coding the treemap, so that each group can be uniquely identified by its color. These settings are set by a manager and are saved in treemap settings file. This view is deployed on the web so that other people can monitor the help desk tickets data. Their idea is to use the same settings but with different data every month.

In the following month, the data file was replaced with the new data file as new tickets are issued. Prior implementation of treemap used to save the attribute values in the settings file and is dependent on the data file. When the new data is loaded with the previous settings file, it crashed. Each time when the data is changed, a new similar settings file is created and deployed on the web.

This problem is solved by modifying the treemap settings file. Temporal treemaps write the attribute name into the file and while reading it is made sure that the settings are applied to the attribute with that name only. Hence, the settings file can be used with changing data.

Another option was to present all the data in a single view. This is achieved by creating a list file with the first column showing the titles (time periods in this case), second column showing the data file names, and the third column showing the treemap settings file names. The actual list file used in this case study is shown in Appendix B. When a settings file is loaded with this list file, a new list of settings tab is displayed beside the detail on demand table on the top right corner (Figure 5.5.2).

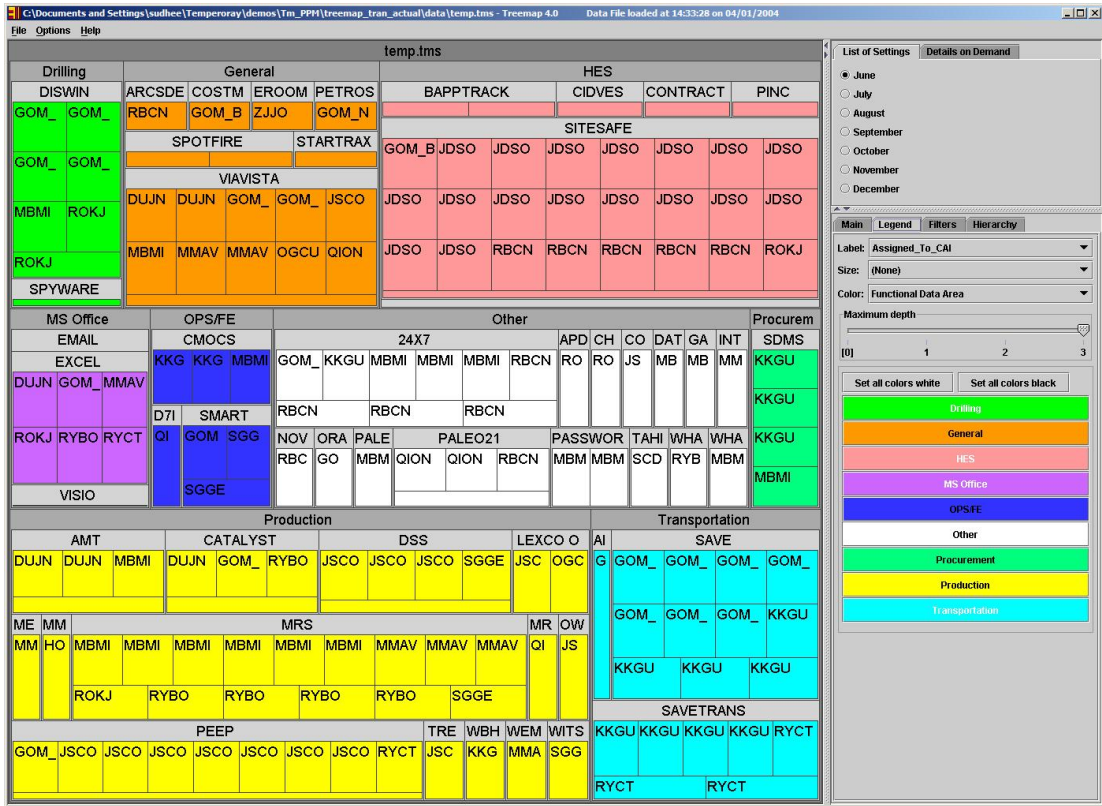


Figure 5.5.2 Visualizing help desk tickets in June. Each rectangle is a ticket, grouped by Functional Data Area and First App attributes. Color shows the Functional Data Area.

Users can access each month data in a single view, without having to load the data separately. Users can select the time period and view the data corresponding to that time period. Figure 5.5.2 shows the help desk tickets in June. Production has highest number of tickets in June.

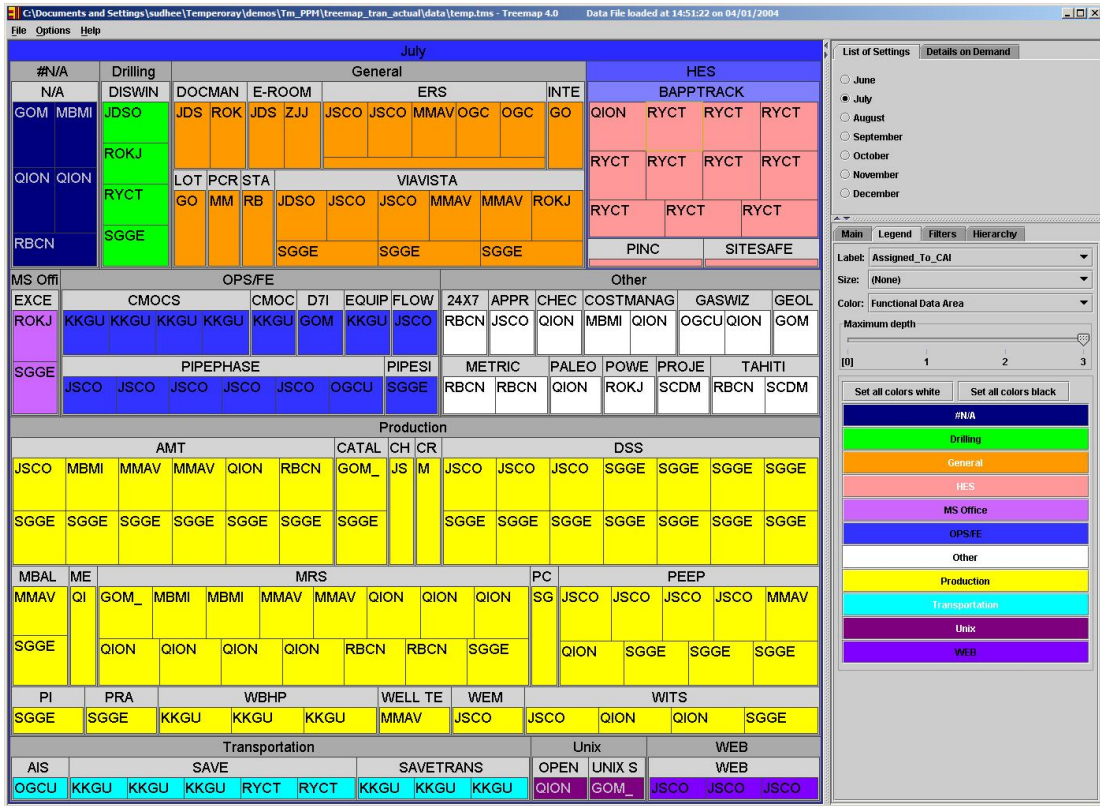


Figure 5.5.3 Visualizing help desk tickets in July.

From Figure 5.5.2 and Figure 5.5.3 it can be seen that there are three new areas (#N/A, Unix, WEB) in which help desk tickets are issued in July but not in June. The number of tickets in HES and Transportation areas has decreased in July compared to June. There are more tickets in Production in July compared to those in June.

Another possible method to visualize the number of tickets in each category is to aggregate the number of tickets in each category and view it using the time slider in temporal treemaps. Users can select to export a time series file from the current visualization. This action would create a new time series file with the categories as items and introduce a new time series attribute whose value is equal to the number of

tickets in each category. Figure 5.5.4 shows the time series visualization of number of tickets in each category for 8 months (June – December).

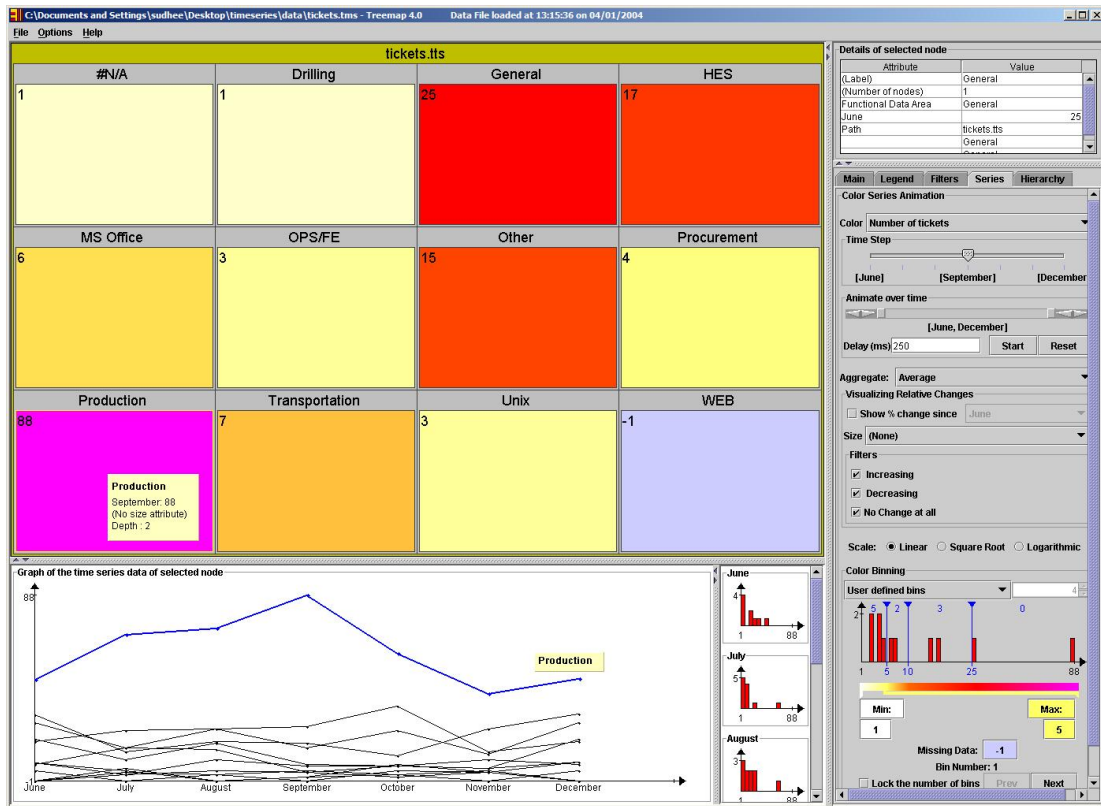


Figure 5.5.4 Visualization of help desk tickets, aggregated by number of tickets in each Functional Data Area, and number of tickets as the time series attribute.

The visualization in Figure 5.5.4 shows the aggregated number of tickets in each category. Each rectangle is a Functional Data Area, grouped by Functional Data Area, labeled by number of tickets in September, colored by number of tickets in September.

From Figure 5.5.4, Production has the highest number of tickets in September. There are no tickets issued for WEB category. Drilling has the minimum number of

tickets issued in September. General, HES, and Other categories have around 30 tickets issued. From the top most time series graph (shown in popup) Production has the highest number of tickets issued in all the months and in September it has 88 tickets issued.

Figure 5.5.5 shows the aggregated time series visualization of number of tickets in each Functional Data Area in December.

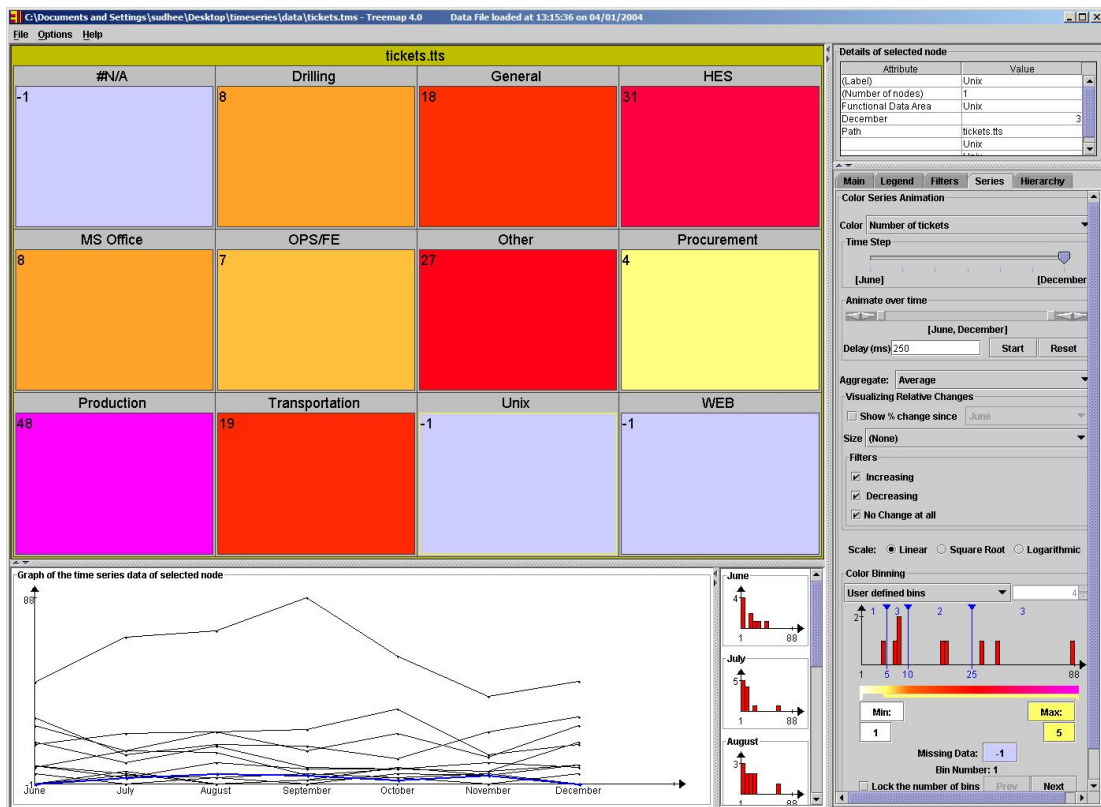


Figure 5.5.5 Visualizing aggregated number of tickets in each area in December.

More number of tickets are issued in each area in December compared to September. No tickets are issued in Unix, WEB, and #N/A areas.

Temporal treemaps present an aggregated time series view of help desk tickets.

Analysis: The original implementation was different and is refined based on the feedback collected from the Chevron Texaco IT department. This work has saved time to create a settings file each time when the data is changed. It also enables the users to view the data in a single view.

Initially buttons are used to show the time periods. There was no cue in the user interface to show which data set the user is working on. The buttons are changed to radio buttons, which clearly show the data set that the user is working on. Also the title of the treemap shows the time period. When the user switches time period, main tab is highlighted. The idea was to leave the tab highlighted as it was before. This is fixed in the modified version.

Another suggestion was to carry over the modified settings from one time period to another time period. This is helpful when users are concentrating on only one category. They can filter all other categories, and then switch the time periods. This can be implemented as an option “Remember settings”. When users select this option, the current settings can be saved in a temporary file and applied after switching to a new time period.

5.6 Feedback

This section discusses the feedback collected from two HCIL members.

Subject 1: This subject has not used treemaps before and was more interested in the time series graph overview and hence commented on presenting more information in that overview. More visual cues like color of the line, color of the time point, missing

value points could be added to the time series graph. One option would be to draw a small circle at each time point on the time series graph so that the color of the circle indicates the data item's color in the treemap. In the same way, missing values can be shown with the circles colored with the missing value color. The time series graph should be disconnected at the time points with missing values for that data item.

The visual clutter observed in most of the examples can be avoided by using a log scale on the y-axis of the graph. This can be provided as an option. Another possible solution is to filter out the extreme time series graphs and zoom the graphs to occupy the entire available space. Apply some interpolation techniques to show a smooth transition from one point to another.

Filters were distracting as there is no cue in the interface that it filters the items whose values were increasing from the “previous” time period.

Subject 2: This subject has extensive experience in using treemaps and was able to use GUI features like color binning without any problem. He was distracted with the slider and was not able to identify the color changes over time. He was able to coordinate the time series graphs with the treemap overview. In the time series graph overview, there is no visual cue, which would indicate the current time period. A vertical bar in the time series graph overview could show the current time period. As the user moves the slider, the position of the vertical bar is updated. Clicking on any time series graph would select the time period nearest to the position of mouse click. This would update the position of the vertical bar, the time slider, and the treemap to show the new time period.

Chapter 6

Conclusions

This thesis extends the concept of treemaps to visualize time series data through mapping temporal changes to the color attribute of treemaps.

6.1 Research Contributions

Specific contributions of this thesis include:

- Visualizing time series data: This thesis extends the concept of treemaps to visualize time series data while retaining the hierarchical structure of the data. Users can identify trends at the item level as well as at the sub tree level. The user interface supports interactive exploration of data with features such as animating over time, visualizing %changes, and filtering data items whose values have increased, decreased or remained constant etc.
- Time series graph overview: The implementation coordinates the time series graph overview with the treemap overview to provide additional capabilities to explore the data. These two views are tightly coupled in response to mouse movements and encourage visual exploration of the data.
- Improved labeling: Temporal treemaps support overlay labeling to gain screen space by removing borders but still retain the context. This is achieved by painting labels at the center or top of the group of items. The labels are painted in such a way that they are visible in both light and black backgrounds.

- **Missing data:** Missing data is one of the common problems in visualizing time series data, which would mislead the users in interpreting the visualization. Temporal treemaps handle missing data in a simple method by using a supplemental value. Users can assign any color to the supplemental value and identify the data items with that color in the treemap. Distraction can be avoided by using a value and color different from the time series values and colors respectively. For example, in visualizing HCIL web logs, supplemental value for page hitcount is -1 and is colored white, which enables users to distinguish existing and non-existing pages in the treemap.
- **Implementation:** Temporal treemaps are an extension to treemap implementation and are implemented in Java using the Swing toolkit for user interface widgets. It uses object oriented design techniques that support the use of sub-classing (packaging) to easily add new classes of features.
- **Systems engineering approach:** The design and implementation of temporal treemaps has followed the systems engineering developmental cycle. Analysis of user requirements, identification of the software classes to be modified, identification of design alternatives, and tradeoff analysis between various design alternatives is carried out prior to the implementation. This approach has resulted in reusable and extensible software modules, in terms of input data format and user interface widgets.
- **Validation through case studies:** The utility of temporal treemaps was demonstrated through five case studies from task domains like health

statistics, web logs (Chapter 5). Animation, ability to view relative changes over absolute values, and tightly coupled graphic overviews helped users to identify trends in their data sets. This has also resulted in numerous suggestions for improvements.

- Section 6.2 describes a subset of future possible extensions to temporal treemaps to increase their utility. Further work needs to be done in this area to identify the potential extensions that are interesting and relevant to the users.
- Temporal treemaps will be included in regular treemaps code base as Treemap 4.2. The user interface features in temporal treemaps must comply with those of the original Treemap 4.1. For example the time series controls in the series tab have to be removed and placed in the legend tab.

6.2 Future Work

This work has mainly focused on exploring the ideas and demonstrating the capabilities thereby leaving a wide scope of future possibilities to extend the current work.

6.2.1 Scaling

Time series data sets are large both in terms of number of data items, number of hierarchy levels or number of time periods to be monitored. For data sets with large numbers of data items (order of 10^5) traditional treemap display may not be helpful since the area occupied by each item is not large enough to visualize the changes in color. For data sets with deeper hierarchies, data items can be aggregated. Scaling

treemaps to handle large data sets can be achieved through animated display [14]. Another possibility of scaling temporal treemaps is to accommodate a large number of time points (100 or more time points). This would include a new input file format, efficient data structures for storing large data and efficient display techniques. Separating time series attributes from static attributes and replacing tab delimiters with other delimiters like “,” or “;” would result in easy management of input files. Another improvement would be to replace the tab delimited file with an xml file and use an xml parser for reading the input data. Horizontal scrolling and zooming should be implemented to interactively view the graphical overview of time series data. The histogram display can be discarded, as it is hard to accommodate large time periods.

6.2.2 Generating textual summaries

The main goal of visualizing time series data is to identify the patterns and/or outliers. Graphical visualization and statistical analysis are two common methods of examining time series data. A method to generate textual summaries by selecting important patterns, mapping the patterns to words and phrases, and generating actual text based on these words and phrases for weather forecasts, gas-turbine sensor readings, and hospital intensive care data is discussed in [39]. MIMSY [16] provides a user interface for textual queries to study temporal patterns in stock market data.

A subroutine can be implemented in temporal treemaps which generates a short summary of time series data upon user request. This summary may contain a description of the time-dependent pattern, outlier items, leaders and laggards along

with the data item details. For example, when the user clicks on a treemap node, a text message “The (attribute name) for (node name) has increased by (%attribute value) from (time period)”.

6.2.3 Coupling TimeSearcher with Treemap

TimeSearcher [17] provides a prototype environment for interactive querying and exploration of time-series data. Temporal treemap utility can be enhanced by tight coupling of treemaps with TimeSearcher. The graphic overview and the menu in TimeSearcher can be added below the main treemap window. The details on demand in TimeSearcher can be shown adjacent to the details on demand table in treemap.

Interaction is achieved by coordinating mouse motions and highlighting. Mouse over in the TimeSearcher graphic overview would highlight the nearest time series graph and the corresponding rectangle/data item in the treemap and vice versa. While posing a query in the time searcher, the corresponding nodes of the qualifying graphs are shown in their original color, while the rest of the items can be grayed out or hidden. This enables users to quickly spot the clusters and patterns in the query result set. Drag and drop query by example can be implemented by dragging an item from the treemap and dropping it in the TimeSearcher’s query space which would result in all graphs with that pattern.

6.2.4 Additional Features

The following are few possibilities that would provide more features or options.

- A modified time slider can be added to the x-axis of the graphic display. This time slider consists of three pointers, one for the current time period and the other two for selecting a range of time periods for animation.
- The color gradient can be added parallel to the y-axis of graphic display. This helps users to quickly match the time series graphs with the data items in the treemap.
- Binning by node option which lets the users create three bins based on the selected node's value, being one less than the value of the selected node, other equal to the node and the third greater than the value of the selected node. This can be done by selecting the node and dragging and dropping the node in the binning widget, which would insert a bin separator at the selected node's value in that particular time period. Binning by average value could be another possible extension to group by the average value, similar to binning by node.
- Export functionality can be added to export selected part of the data in selected time periods to create a new treemap time series data file.

Appendix A

A Sample Treemap Timeseries (.tts) data file

A sample timeseries data file is shown in Figure A.1 based on help desk tickets data. The original data is modified and only a few data items are shown to illustrate the file format. The static data attributes are Functional Data Area and First App and the time series attribute is Number of tickets over four time periods (June, July, August, and September).

The first four rows show the data attribute information. Each column corresponds to a data attribute and each row shows the data item. The missing value is shown in the first row, fourth column.

The image shows a spreadsheet titled 'L.tts' with columns A through H. The first four rows (1-4) contain static data attributes: Functional Data Area, First App, and Number of tickets for June, July, August, and September. Rows 5-14 contain data items with their attributes and a hierarchy column. Annotations include: 'Attribute Names' pointing to columns A-D, 'Data items' pointing to rows 5-14, 'Missing Value' pointing to cell D1, 'Time Periods' pointing to columns E-H, and 'Hierarchy' pointing to column H.

	A	B	C	D	E	F	G	H
1	1	1	4	-1	4	4		
2	Single	Single	Number of t	Number of t	Number of t	Number of t		
3	Functional Data Area	First App	June	July	August	September		
4	STRING	STRING	INTEGER	INTEGER	INTEGER	INTEGER		
5	Other	INTELLEX	1	-1	3	2	Other	
6	Other	COST CONTROL	-1	-1	-1	-1	Other	
7	OPS/FE	INTOUCH	-1	-1	-1	-1	OPS/FE	
8	Other	HYSYS	-1	-1	-1	-1	Other	
9	Production	TECPLOT	-1	-1	-1	-1	Production	
10	Production	CRYSTAL BALL	-1	1	-1	-1	Production	
11	OPS/FE	NAVIS WORKS	-1	-1	-1	-1	OPS/FE	
12	General	E-ROOM	-1	2	-1	-1	General	
13	Other	PASSWORD UTILITY	2	-1	-1	1	Other	
14	OPS/FE	PIPESIM	-1	1	-1	-1		

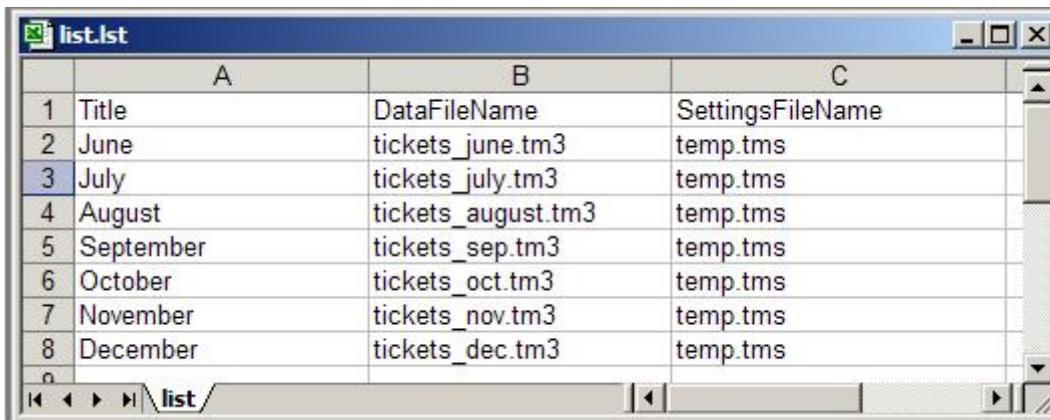
Figure A.1 A sample timeseries (.tts) data file.

Each line from the fifth line is a data item with its attributes shown in respective columns. Pre-defined hierarchy can be specified for each data item at the end of row, with a blank line between the attributes and the hierarchy.

Appendix B

A sample list of settings file

A sample list of settings file is shown in Figure B.1. This is the actual list file based on help desk tickets data for eight months. The first header line shows the names of the columns. First column shows the time period name (shown in treemap as list of radio buttons), second column shows the treemap data file name (.tm3), third column shows the treemap settings file name (.tms).



	A	B	C
1	Title	DataFileName	SettingsFileName
2	June	tickets_june.tm3	temp.tms
3	July	tickets_july.tm3	temp.tms
4	August	tickets_august.tm3	temp.tms
5	September	tickets_sep.tm3	temp.tms
6	October	tickets_oct.tm3	temp.tms
7	November	tickets_nov.tm3	temp.tms
8	December	tickets_dec.tm3	temp.tms

Figure B.1 A sample list of settings (.lst) file.

Appendix C

TTSGenerator

TTSGenerator is a command line oriented Java subroutine which generates a treemap time series (tts) data file from two or more treemap (tm3) data sets. It reads the data from the input files and merges the attributes for similar data items in the time series format based on input key. The static attribute values are same for any data item in all the input files and the attribute values are different for the time series attributes. A missing value is substituted for the time series attribute if the data item does not exist in any time period. A key is a combination of one or more attributes and/or hierarchy. If the data has a predefined hierarchy, then it is sufficient to specify the hierarchy as key, otherwise the unique attribute or combination of attributes should be specified as the key.

For example, the following command generates a time series file temp.tts from three treemap files t1.tm3, t2.tm3, and t3.tm3.

```
Java -jar ttsngen.jar -o temp.tts 3 t1.tms t2.tm3 t3.tm3 1 B -1 1 H
```

- *-jat ttsngen.jar*: jar file to execute TTSGenerator.
- *-o temp.tts*: specifies that the output of TTSGenerator is written into temp.tts file.
- *3 t1.tm3 t2.tm3 t3.tm3*: “3” indicates the number of input treemap data sets (tm3) and “t1.tm3, t2.tm3, t3.tm3” are the input treemap data set file names.

- *I B -I*: “1” indicates the number of time series attributes, “B” shows the time series attribute column in the input treemap data files and “-1” shows the missing value.
- *I H*: 1 specifies the number of attributes that constitute the key. This key is used to combine the data attributes in different files to one file. “H” is the first column of the hierarchy.

Note: The column names are the headers that appear in Microsoft Excel on top of each column.

References

- [1] C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 313-317. 1994.
- [2] K. Andrews and H. Heidegger. Information slices: Visualizing and exploring large hierarchies using cascading, semicircular disks. In *Proceedings of IEEE Infovis 1998 Late Breaking Hot Topics*. IEEE, 9-11. 1998. <ftp://ftp.iicm.edu/pub/papers/ivis98.pdf>
- [3] L. Beaudoin, M-A. Parent, L. C. Vroomen. Cheops: a compact explorer for complex hierarchies. In *Proceedings of the 7th Conference on Visualization, 1996*. IEEE, 87-ff. October 1996.
- [4] B. B. Bederson and B. Shneiderman. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann. April 2003.
- [5] B. B. Bederson, B. Shneiderman, M. Wattenberg. Ordered and quantum treemaps: making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics (TOG)*, 21(4):833-854, October 2002.
- [6] S. Bjork. Hierarchical flip zooming: enabling parallel exploration of hierarchical visualizations. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. IEEE, 232-237. May 2000.
- [7] R. Boardman. Bubble trees: the visualization of hierarchical information structures. *CHI '00 Extended Abstracts on Human Factors in Computer Systems, POSTER SESSION: Student Posters, 2000*. ACM, 315-316. April 2000.
- [8] S. Card, J. Mackinlay, B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann. San Francisco. 1999.
- [9] J. V. Carlis and J. A. Konstan. Interactive Visualization of Serial Periodic Data. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*. ACM, 29-38. November 1998.
- [10] A. Cedilnik and P. Rheingans. Procedural annotation of uncertain information. In *Proceedings of the Conference on Visualization, 2000*. IEEE, 77-83. 2000.
- [11] E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, S. K. Card. Visualizing the evolution of web ecologies. In *Proceedings of the SIGCHI*

- Conference on Human Factors in Computing Systems, 1998.* ACM, 400-407. 1998.
- [12] E. H. Chi and S. K. Card. Sensemaking of evolving web sites using visualization spreadsheets. In *Proceedings of IEEE Symposium on Information Visualization, 1999.* IEEE, 18-25. San Francisco, October 1999.
 - [13] S.B. Cousins and M.G. Kahn. The visual display of temporal information. *Artificial Intelligence in Medicine, 1991.* 3(6):341-357, 1991.
 - [14] J-D. Fekete and C. Plaisant. Excentric labeling: dynamic neighborhood labeling for data visualization. In *Proceedings of the CHI Conference on Human Factors in Computing Systems.* ACM, 512-519. 1999.
 - [15] M. Friendly. A Brief History of the Mosaic Display. *Journal of Computational & Graphical Statistics, 2002.* 11(1):89-107, March 2002.
 - [16] W. Gibson. MIMSY: A system for analyzing time series data in the stock market domain. Master's thesis, University of Wisconsin, Department of Computer Science, 1993.
 - [17] H. Hochheiser. Interactive graphical querying of time series and linear sequence data sets. Ph. D. Dissertation, University of Maryland, Department of Computer Science, 2003.
 - [18] H. Hochheiser and B. Shneiderman. Coordinating overviews and detail views of www log data. In *Proceedings of Workshop on New Paradigms in Information Visualization and Manipulation, 2000.* ACM, November 2000.
 - [19] H. Hochheiser and B. Shneiderman. Using interactive visualizations of www log data to characterize access patterns and inform site design. *Journal of the American Society for Information Systems, 52(4):* 331-343, February 2001.
 - [20] C. Jeong and A. Pang. Reconfigurable disc trees for visualizing large hierarchical information space. In *Proceedings of IEEE Symposium on Information Visualization, 1998.* IEEE, 19-25. October 1998.
 - [21] B. Johnson and B. Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd International Visualization Conference, 1991.* IEEE, 284-291. October 1991.

- [22] D. A. Kiem. Pixel-oriented visualization techniques for exploring very large databases. *Journal of Computational and Statistical Graphics*. 58-77, March 1996.
- [23] E. Kolatch and B. Weinstein. CatTrees: dynamic visualization of categorical data using Treemaps. *url:*
http://www.cs.umd.edu/class/spring2001/cmsc838b/project/Kolatch_Weinstein, 2001.
- [24] M. Kreuseler and H. Schumann. A flexible approach for visual data mining. *IEEE transactions on Visualization and Computer Graphics, Special Selections on Information Visualization and Visual Data Mining 2002*. 8(1):39-51, January – March 2002.
- [25] J. Lamping, R. Rao, P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems 1995*. ACM, 401-408. May 1995.
- [26] J.D. Mackinlay, G.G. Robertson, R. DeLine. Developing calendar visualizers for the information visualizer. In *Proceedings of the 7th annual ACM symposium on User Interface Software and Technology, 1994*. ACM, 109-118. 1994.
- [27] L. Nowell, E. Hetzler, T. Tanasse. Change blindness in information visualization: a case study. In *Proceedings of the IEEE symposium on Information Visualization, 2001*. IEEE, 15. 2001.
- [28] C. Plaisant, G. Chintalapani, C. Lukehart, D. Schiro, J. Ryan. Using visualization tools to gain insight into your data. *SPE Annual Technical Conference and Exhibition*. Denver. October 2003.
- [29] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, B. Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of American Medical Informatics Association Annual Fall Symposium*. AMIA, 76-80. 1998.
- [30] R. Rao and S. Card. The table lens: merging graphical and symbolic representation in an interactive focus+context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems 1994*. ACM, 318-322. April 1994.

- [31] G. Robertson, J. Mackinlay, S. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems 1991*. ACM, 189-194. March 1991.
- [32] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 1992. 11(1):92-99, January 1992.
- [33] S.F. Silva and T. Catarci. Visualization of linear time oriented data: a survey. In *Proceedings of the 1st International Conference on Web Information Systems Engineering, 2000*. IEEE, 310. June 2000.
- [34] E. Sirin and F. Yaman. Visualizing dynamic hierarchies in Treemaps. url: <http://www.cs.umd.edu/class/spring2002/cmsc838f/project/>, 2002.
- [35] SmartMoney, url: <http://www.smartmoney.com>.
- [36] R. Spence. *Information Visualization*. Essex, England, Addison-Wesley. 2001.
- [37] M. Spenke and C. Beilken. Visualization of trees as highly compressed tables with InfoZoom. In *Proceedings of IEEE Symposium on Information Visualization 2003*. IEEE. October 2003.
- [38] Spotfire, url: <http://www.spotfire.com>
- [39] S.G. Sripada, E. Reiter, J. Hunter, J. Yu. Generating English summaries of time series data using the gricean maxims. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003*. ACM, 187-196. 2003.
- [40] E. Tufte. *The Visual Display of Quantitative Information*. Cheshire, CT, Graphics Press. 1983.
- [41] R. Twiddy, J. Cavallo, S. M. Shiri. Restorer: a visualization technique for handling missing data. In *Proceedings of the Conference on Visualization 1994*. IEEE, 212-216. 1994.
- [42] J. J. Van Wijk and E. Van Selow. Cluster and calendar-based visualization of time series data. In *Proceedings of 1999 IEEE Symposium on Information Visualization*. IEEE Computer Society, 4-9. 1999.