Visual Discovery and Analysis

Stephen G. Eick, *Senior Member*, *IEEE*

Abstract—We have developed a flexible software environment called ADVIZOR for visual information discovery. ADVIZOR complements existing assumptive-based analyses by providing a discovery-based approach. ADVIZOR consists of five parts: a rich set of flexible visual components, strategies for arranging the components for particular analyses, an in-memory data pool, data manipulation components, and container applications. Working together, ADVIZOR's architecture provides a powerful production platform for creating innovative visual query and analysis applications.

Index Terms—Information visualization, data analysis, visual design patterns, perspectives, linked views.

1 Introduction

Immense amounts of information are available to decision-makers today. Tomorrow, there'll be even more. And that's the problem. Data is exploding. Every report produces another. Presentations get more complex. The best opportunities are often the hardest to see. The challenge is to find the treasure, to spot the nuances in customer behavior that will give a quick, sure market edge or help notice a profitable trend, or make the right inventory call.

Previous approaches to making sense of data involved manipulating text displays such as cross tabs, running complex statistical packages, and assembling the results into reports using presentation graphics. Browsers and the web have popularized the idea that modern interfaces combine text and graphics. Extending this trend, we have developed an interactive visualization-based application to help users make sense of data and take actions. Our tool, called ADVIZOR, takes the web one step further by making the text and graphics interactive, using rich metaphors to encode information, and enabling the user to pose and resolve queries dynamically using the mouse. Our goal is to provide users with an immediate and intuitive grasp of what is significant and actionable in their data.

In contrast to previous approaches used in statistical graphics or scientific visualizations, ADVIZOR is an information visualization system focusing on customer analysis. As illustrated in Fig. 1, it uses visual metaphors, data structures, and interactive operations to help users see micro trends and anomalies in business data.

Broadly speaking, the problems addressed by information visualization tools fall into three classes:

1. *Presentation Graphics* such as is included with Microsoft PowerPoint™ or even spreadsheet graphics. These generally consist of bars, pies, and line charts that are easily populated with static data and drop into printed reports or presentations. The next generation of presentation graphics, exemplified by VRML-based browsers, enriches the static displays with a 3D-information landscape. Users can then navigate through the landscape and animate it to display time-oriented information. This class of visualizations is generally useful for answering "what" questions and for conveying results.
2. *Visual Interfaces for Information Access* are focused on enabling users to navigate through complex information spaces, such as the web, to locate and retrieve information. Supported user tasks involve searching, back tracking, and history logging. User interface techniques attempt to preserve user context and support smooth transitions between locations.
3. *Full Visual Discovery and Analysis* systems, such as ADVIZOR, that combine the insights communicated by presentation graphics with an ability to probe, drill-down, filter, and manipulate the display to answer the "why" questions as well as "what" questions.

The difference between answering a "what" and a "why" question involves an interactive operation. For example, in a set of sales data, the answer to "what happened" might be that sales went up, generally discernable in the slope of a line graph or the height of a bar in a bar chart. Answering why requires an interactive operation such as drilling-down, drilling-across, excluding, or rescaling to discover that increased sales were due to one product having an exceptional quarter. Going further requires a drill-down, e.g., showing sales increases by customer purchases to notice that the sales increase is due to a single huge order.

1.1 User Needs

Many organizations, particularly within the business community, have made significant investments in collecting, storing, and converting business information into actionable results. Unfortunately, typical implementations of business intelligence software[1] have proven to be too complex for most users except for their core reporting and charting capabilities. Users' demand for multidimensional analysis, finer data granularity, and multiple data sources simultaneously, all at Internet speed, require too much specialist

• *The author is with Visual Insights, Inc., 215 Shuman Blvd., Suite 200, Naperville, IL 60563-8495. E-mail: eick@visualinsights.com.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 110935.

1. Business Intelligence is a broad industry analyst category that includes all software used to gain insights and business data.

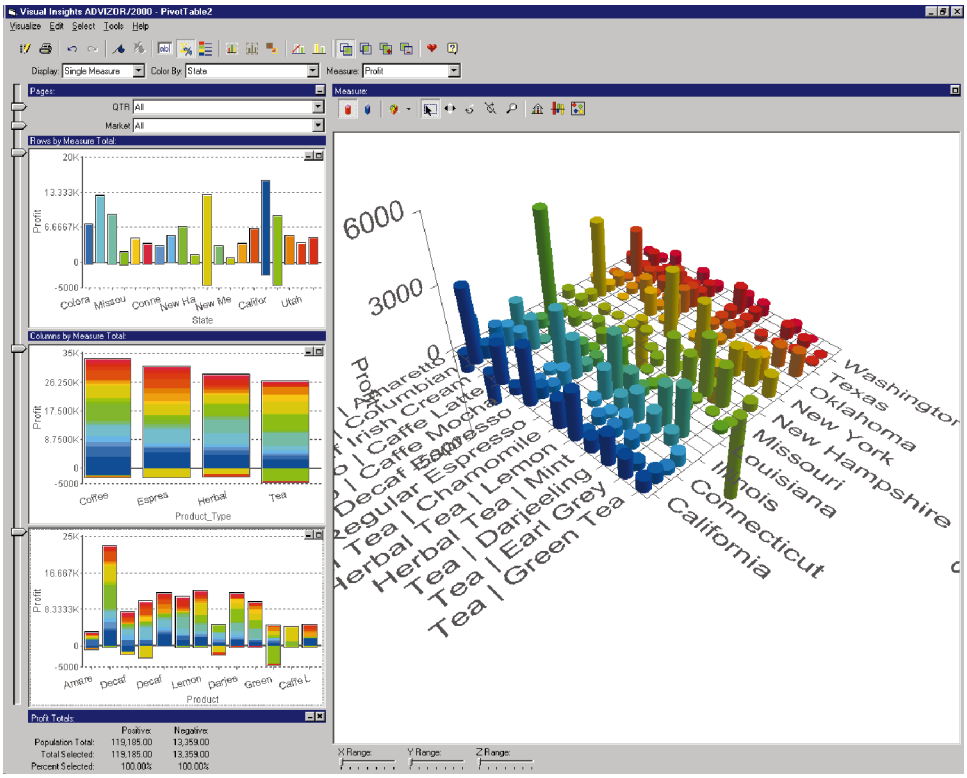

Fig. 1. An ADVIZOR/2000 perspective with bar charts (left) and a Multiscape (right) for PivotTable visualization.

intervention for broad utilization. The result is a report explosion in which literally hundreds of predefined reports are generated and pushed throughout the organization. This is in direct conflict with the needs of front line decision-makers and knowledge workers who are demanding to be included in the analytical process. Finding a strategic outlier in multiple pages of nicely formatted reports or cross tab displays is analogous to finding a needle in a haystack.

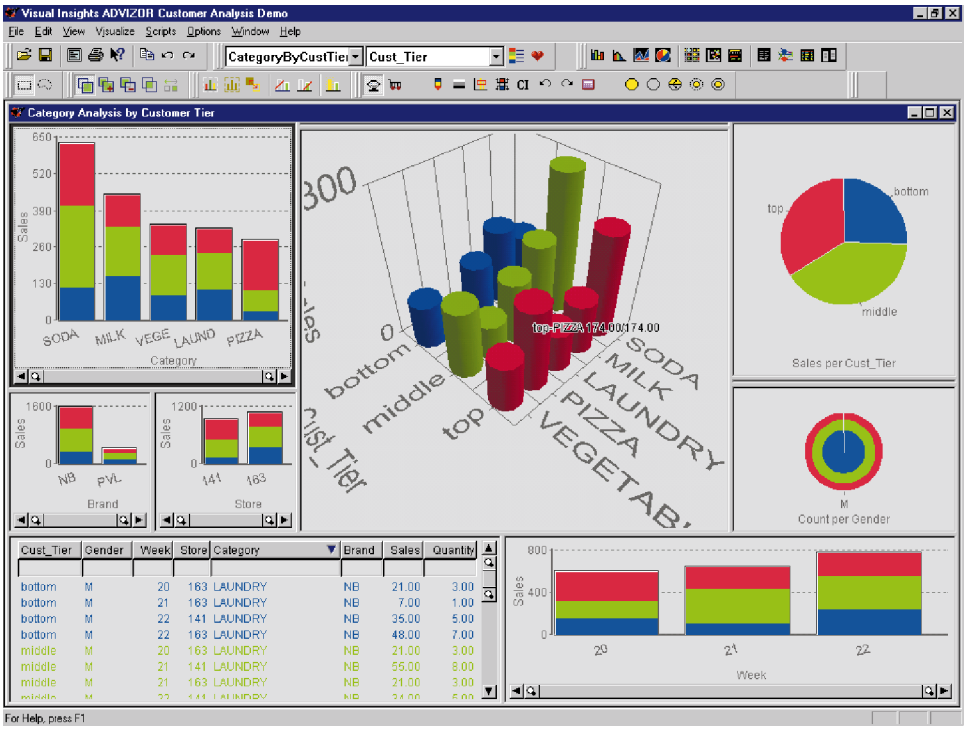

Fig. 2. An ADVIZOR perspective showing male purchases in weeks 20-22 for the top five product categories.

A report-based style of analysis is "assumptive-based" since reports presuppose the relationships that are reported upon. Visual analysis, as presented in this paper, proposes a new style of "discovery-based" analysis. Presenting information visually in an environment that encourages the exploration of linked events leads to deeper insights and more actionable results. This is particularly true when applied to time and detail critical applications associated with customer behavior, such as cross selling, target marketing, total quality of service, customer loyalty, and product and category analysis.

In today's highly sophisticated retail world, for example, product purchasing decisions often are influenced more by product features and attributes such as color and size rather than by brand or item; distribution strategies involving geographic attributes, such as shelf position, may prevail over channel or store location; detailed customer demographics and real time behavior statistics define segments more than buying communities and behavior trends; and time is now measured in Internet time. In a dynamic retail environment, it is impossible to combine the effects of all these attributes in a meaningful way using reports.

Combining these factors to find salient relationships constitutes complex analysis. When confronted with the complexity of such inqueries, users face tough problems: Where do I start? What looks interesting here? Have I missed anything? What are other ways to derive the answer? Is there other data available? People think iteratively and ask ad hoc questions of complex data while looking for insights. Predefined reports and drill downs through OLAP cubes specifically inhibit the power of human thought. Ad hoc "slice and dice" of tabular data is difficult and limited by the inability to display and comprehend more than several dimensions and two-dozen rows at a time.

To help address these and related questions, we have developed software and a methodology supporting visual discovery. Our approach, which we call "Visual Query and Analysis," is embedded in ADVIZOR and involves:

1. presenting complex multidimensional information in a natural an intuitive way;
2. structuring a visual analysis to support discovery;
3. providing fast navigation through large and complex datasets;
4. identifying important areas with a single mouse click;
5. publishing analysis in web documents; and
6. sharing results sets with operational systems for implementing business actions.

Candidate visual query, discovery and analysis (VQA) users include business, government, and research organizations who have made investments in a data infrastructure and need a richer organization-wide analysis capability. In many ways, VQA complements the reporting capabilities of existing solutions with significantly more powerful and easier to use analytical functions. VQA identifies actionable contitions based on detail, attribute, and micro-trend analysis, resulting in better business decisions—faster.

1.2 Human Intelligence

When designing and implementing VQA systems, it is important to consider the perceptual acuity and activities of the end user. Visuals must be arranged to reveal key answers to task-specific questions. Visual encoding of data must be perceptually effective. Interactive operations must be natural and intuitive. Frequently, data conditioning is necessary, e.g., taking square roots of count data or logging salary data.

Our experience is that designing perceptually effective, easy-to-use software is extremely hard and often task dependent. There are a huge number of conflicting goals and the best solution frequently involves engineering tradeoffs. Our approach to overcoming these problems involves implementing and trying many solutions, evaluating their usefulness with real users, and capturing the successful solutions in methodologies and a knowledge base.

1.3 Research Contributions

This paper addresses four significant research questions:

1. *What visual components or views are needed for visual analysis?* In our work, we have developed a set of 11 components that are broadly useful. Furthermore, our components are highly scalable and can display significantly larger data sets than their traditional counterparts. (See Section 2.1.)
2. *How should visual components be organized?* We propose fixed arrangements, called perspectives, that answer specific classes of questions. The advantage of perspectives over general linked view analysis systems is that they significantly reduce complexity for nonexpert users. Perspectives are "authored" by domain experts, provide a starting point and guiding framework for interactive analysis by power users, and can be viewed by casual users. (See Section 2.2.)
3. *What are the recurring visual design patterns in the perspectives?* In our work implementing solutions, we have found certain patterns of components are broadly useful for general classes of problems. (See Section 2.3.)
4. *How is this class of software application built?* We have developed a plug and play software architecture and data model for building visual query and analysis applications. Components supporting our framework automatically integrate into the environment, support all interactive operations, and are easily programmed by our containers.

This paper is organized into five sections. Section 2 presents our visual components and their organization into perspectives. Section 3 describes our software architecture and interesting implementation details. Section 4 briefly lists some related research and Section 5 concludes.

2 Components, Perspectives, and Design Patterns

This section describes our visual components, explains how we organize them into perspectives, and highlights design patterns in the perspectives.

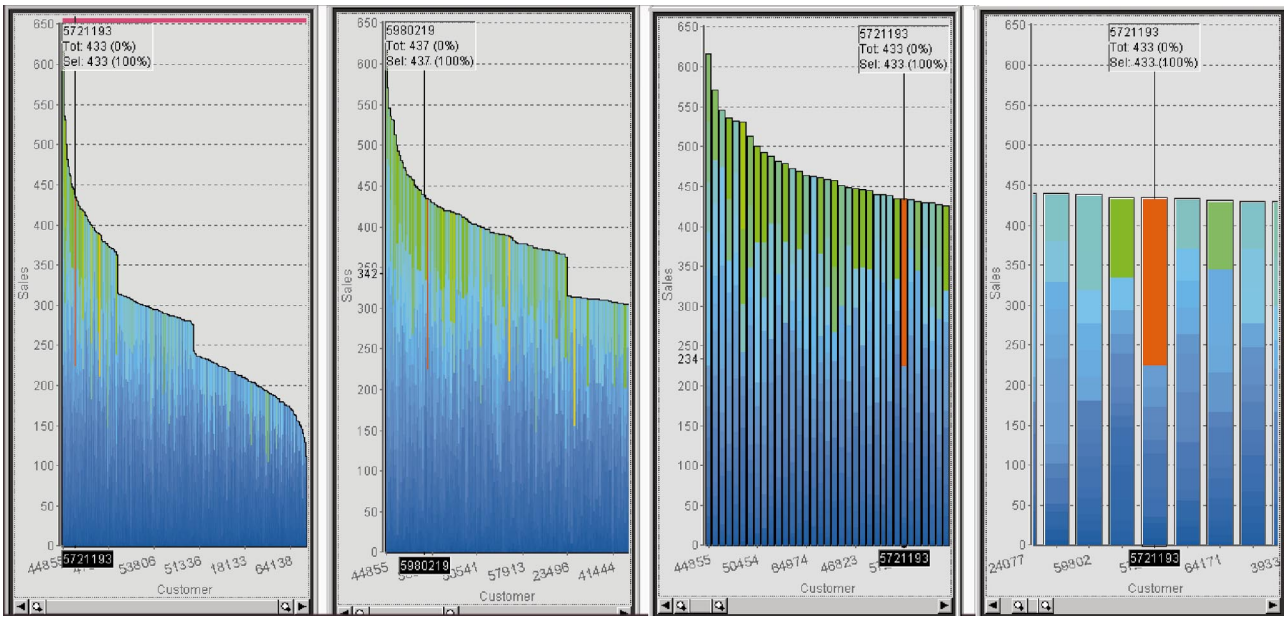

Fig. 3. Bar chart scalability is increased by using levels of rendering detail, a red over-plotting indicator (left), and a 1,000 to 1 zoombar.

2.1 Visual Components

ADVIZOR includes 11 interactive visual components or "views." Nine are graphical, one is textual, and one does both.[2] Each component is richly parameterized and can assume many different visual looks. Even a simple bar chart, for example, can be oriented vertically or horizontally, pointed left or right, and may appear as a spine plot or a two-dimensional Treemap [16]. The bars may be interactively reordered, sorted, and shown in groups.

Components may be partitioned into three groups:

1. *Traditional* visual metaphors including:
   - **Bar Chart** that shows aggregations and frequencies, as in Fig. 3, that uses a 1,000 to 1 scaling "zoombar," an over-plotting indicator, and stacked colors to show tens of thousands of bars.
   - **Histogram** that shows the distribution of a single numeric variable with adjustable interactive smoothing.
   - **Line Chart** for understanding trends in ordered.
   - **Pie Chart** for visualizing fractions of a total.
   - **Scatter plot** for bivariate analysis with interactive scaling, navigation, and point manipulation.
2. *Novel* metaphors:
   - **Time Table**, Fig. 4, is a component for showing thousands of time-stamped events. Described in [9], it encodes each event using a tick mark positioned on a *time-by-type* grid. The tick marks are color, angle, and shape-coded to represent the events, type, and associated characteristics, and may contain tails to encode the event duration. Scalability is increased by including an over-plotting indicator, filtering, and zoombars.
   - **Multiscape**, shown in Fig. 1 (right), is a landscape visualization. It encodes information using 3D glyphs ("skyscrapers") on a 2D landscape. The glyph symbol choice is parameterized, the bars may be negative or positive, interactively labeled, and rendered in 2D or 3D. The 3D renderings are exciting and dynamic, whereas the 2D better supports crisp comparisons between glyph sizes.
   - **ParaBox**, Fig. 5, combines Box, parallel coordinates, and bubble plots, for visualizing n-dimensional data. It handles both continuous and categorical data [6]. The reason for combining Box and parallel coordinate plots involves their relative strengths. Box plots work well for showing distributional summaries. Parallel coordinates' strength is their ability to display high dimensional outliers, individual cases with exceptional values. Combining the two techniques results in a visual component that excels at both tasks.
   - **Data Constellations**, illustrated in Fig. 6, is a component for visualizing large graphs. Many data sets can be represented as graphs including purchasing associations, web click streams, and market basket analysis. Two tables parameterize Data Constellations, one corresponding to nodes and another to links. It contains four built-in graph layout algorithms, described in [22], that dynamically position the nodes so that patterns are emergent. The strength of the algorithms is their ability to layout extremely large graphs with thousands of nodes and links.

2. Research versions of another dozen or so components have been developed, but are not yet integrated into the tool.

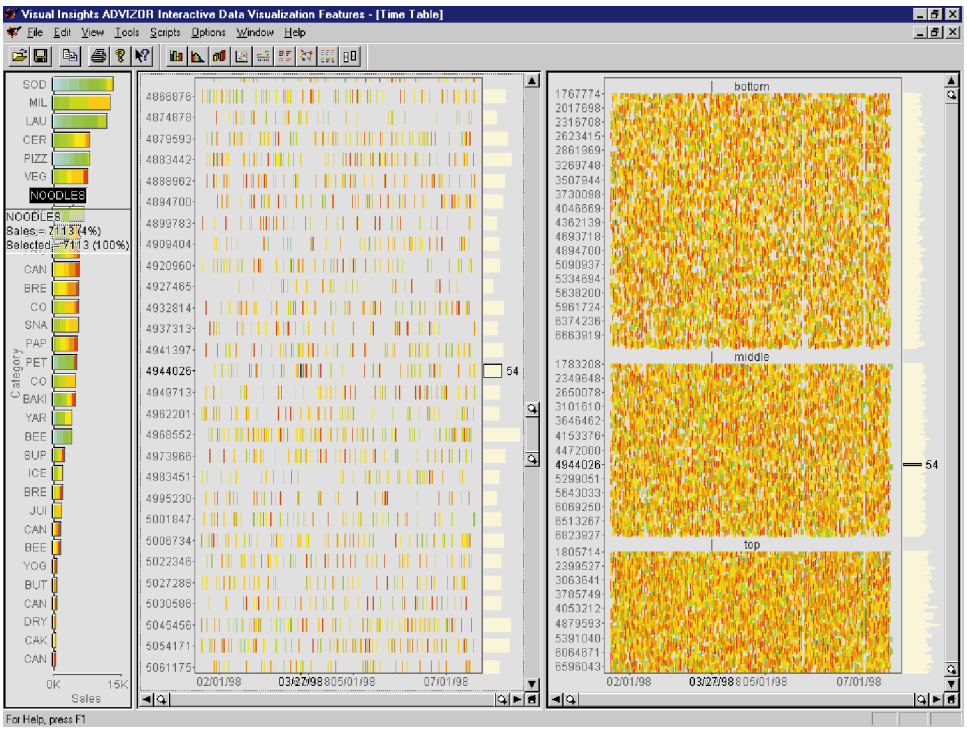

Fig. 4. *TimeTable* shows thousands of time-stamped events using color-coded tick marks.

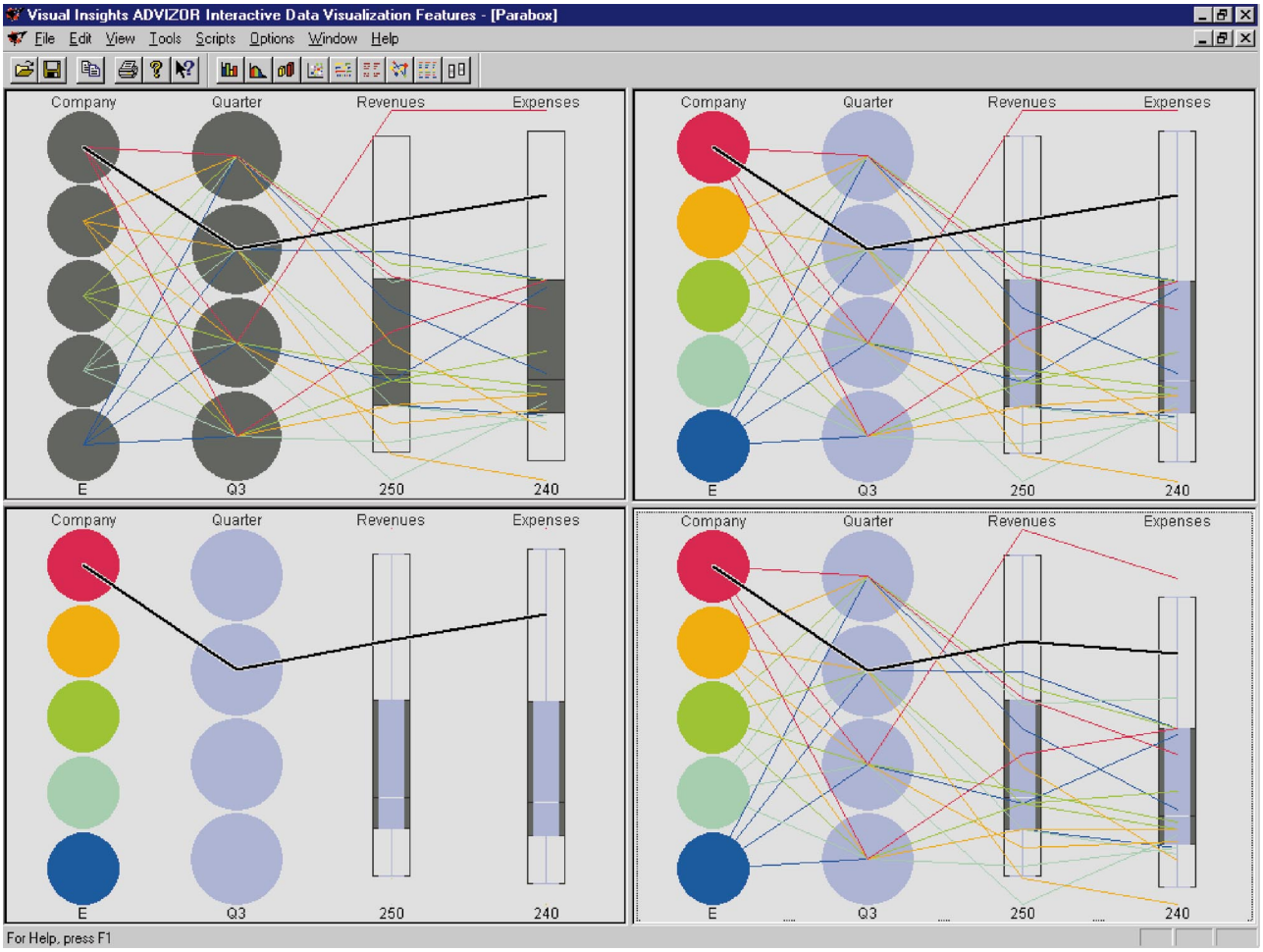

Fig. 5. ParaBox is a combination of Box plots and parallel coordinate plots for showing multidimensional data.

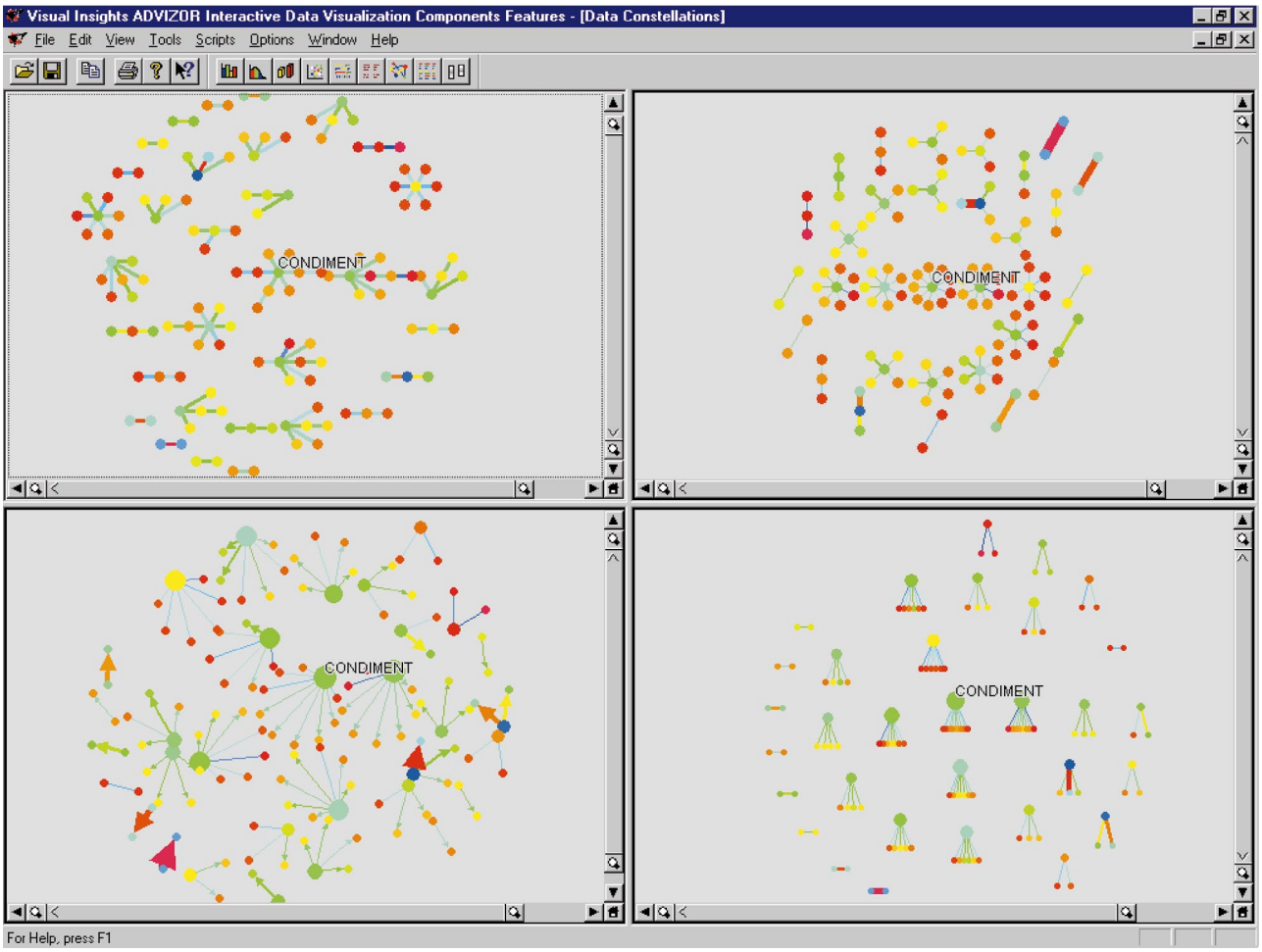

Fig. 6. Data Constellations is a component for visualizing large graphs. Each of the four components is a different graph layout algorithm.

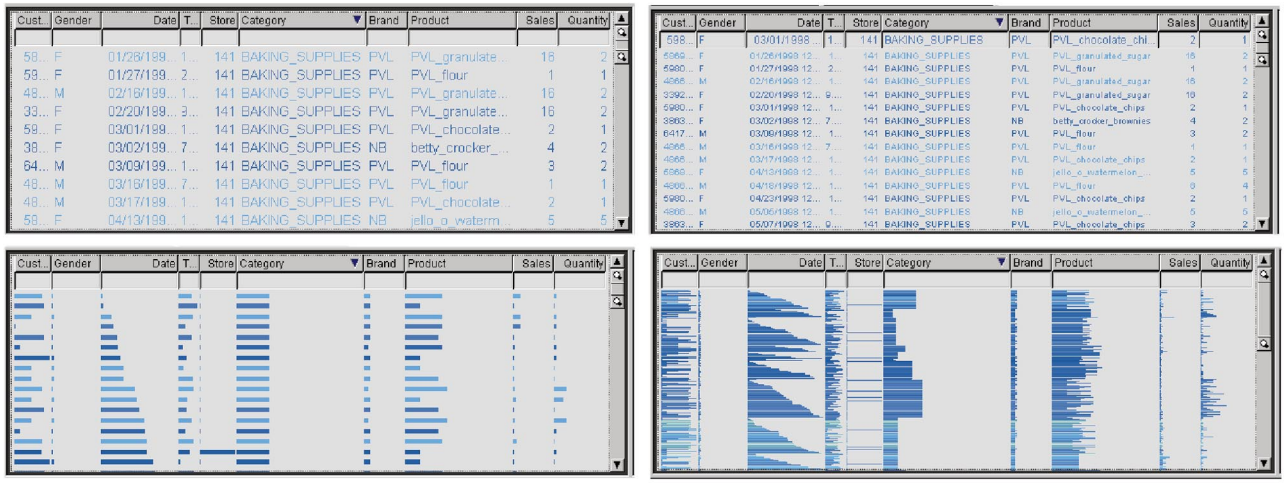

Fig. 7. Data Sheet is a scalable text view that smoothly transitions between text and graphics to increase view scalability. Upper left: full scale, lower right: completely smashed where each thin bar represents a text field.

- *Text-oriented* tools:
  - **Data Sheet**, shown in Fig. 7, is a dynamic scrollable text visualization for tables [7] that bridges the gap between text and graphics. It extends traditional displays by color coding cell values, and supporting sorting and filtering. Furthermore, as the user adjusts the zoom factor with the scrollbar, the component progressively displays more information, using smaller and smaller fonts, eventually switching to one-pixel high bars. This process is called smashing.

- **Counts** is a text view providing a statistical summary of a data table and information about the table's selection vector (see Section 2.2).

Perhaps the most unique aspect of the visual components is their scalability. The Bar Chart, for example, can easily display tens of thousands of bars. At full scale, the bars are collapsed down to a smoothed distribution. As the user zooms in, the scale progressively expands until the bars become individually visible and, eventually, become standard size. Time Table can easily handle 100,000 tick marks and Data Constellations scales to tens of thousands of nodes.

Each component is packaged as an ActiveX control with a set of properties that may be manipulated programmatically or through the ADVIZOR GUI. Each control has five classes of properties:

1. *Data* that binds the view to one or more columns of a data table in the data pool.
2. *Color* that automatically ties color scales to data ranges.
3. *Selection* for modifying the user interaction model.
4. *Component-specific* properties such as smoothing, node placement, and statistical parameters.
5. *Viewing and Navigation*, including view-port operations such as panning and zooming, and camera position for the 3D components.

The *Color* and *Selection* properties affect the data table attached to the component and, thus, carry over to all other components attached to the same data table.

2.1.1 Context Menus

The visual components each contain context menus for users to manipulate the properties. The context menus may be customized for particular applications, or overridden, but are particularly useful for hosting the components in containers, such as browsers, with limited UI support.

2.1.2 Undo/Redo Command stack

Every component maintains an *undo/redo* command stack, containing a history of component changes. By manipulating the command stack, container applications can back out or repeat user manipulations.

2.1.3 Color Scales

A key strength of ADVIZOR is its use of color. Color, a retinal variable [3], is frequently tied to a data range, and used to overlay an additional dimension on a conventional visual.

ADVIZOR supplies six color scales. *Rainbow*, the most popular scale, walks the color wheel from blue to red. The other scales are *Green/Red*, *Pastel*, *Equalized*, *Thermal* (red to white), and, of course, *Gray* level (light to dark). A general problem with color scales is that increments in color are not perceived as equal increments in the data value. The *Equalized* scale is a perceptually uniform scale based on user experimentation that is perceived correctly.

2.1.4 Component-Specific Properties

Properties such as view port manipulations, e.g., panning and zooming, are view-specific. Besides scrolling, other nonlinked properties include scaling, sorting, label options, and layout.

2.2 Perspectives

Each component has inherent strengths and weaknesses. Some (particularly Multiscape, Data Constellations, Time Table, and ParaBox) support high-level structure discovery. Others, for example Data Sheet, provide immediate access to details, but are less effective at conveying structure. Bar and pie charts are particularly useful as filters for selection and to summarize data. A key contribution in ADVIZOR is to organize sets of components into perspectives, which are multiple linked components that work together to answer a class of questions. Many questions are just too complicated to answer with a single component. A well-engineered perspective contains a set of components that function together in a complementary manner. (See Fig. 1.) Each component shows the data from a different perspective. Combined components are more powerful than the sum of each individual contribution.

2.2.1 Linking in Perspectives

Components attached to the same data table in a perspective are linked in four ways: by *color*, *focus*, *selection*, and *exclusion*. Components linked by color use the same color scale. *Focus*, *selection*, and *exclusion*, described below, are interactive properties involving ADVIZOR's *case-based* linking model [10].

In case-based linking, selection, focus, labeling, and other interactive operations affect the state of cases (or rows) in the data table. State information is kept in vectors parallel to the fields in the data table. When state changes occur, other components displaying this table are automatically notified and can update themselves to reflect the new state information. Other types of linking include constraint-linking, where constraints specify the relationships between components, and many-to-one linking.

Fig. 8 illustrates the automatic linking among the components using a bar chart and Data Sheet. Both are visualizing the product sales table. Focusing on `quaker oats crunchberries` in the DataSheet automatically highlights details off the `CEREAL` bar in the bar chart.

2.2.2 Selecting, Excluding, and Restoring

Associated with every data pool table is a parallel selection vector that marks each row in one of three mutually exclusive states: *selected*, *unselected*, or *excluded*. By convention, data items corresponding to the *excluded* rows are not displayed, the *unselected* rows are drawn in light gray, and the *selected* rows are drawn in color.[3] Excluded rows may be restored using menu operations. Initially, by default, all rows in a table are selected.

Used together, *selection* and *exclusion* are extremely powerful. Fig. 9 shows, for example, a perspective using bar charts to display sales by quarter, year, sales channel, and product and a Multiscape showing sales by product

3. Default colors may be customized via a color property page.

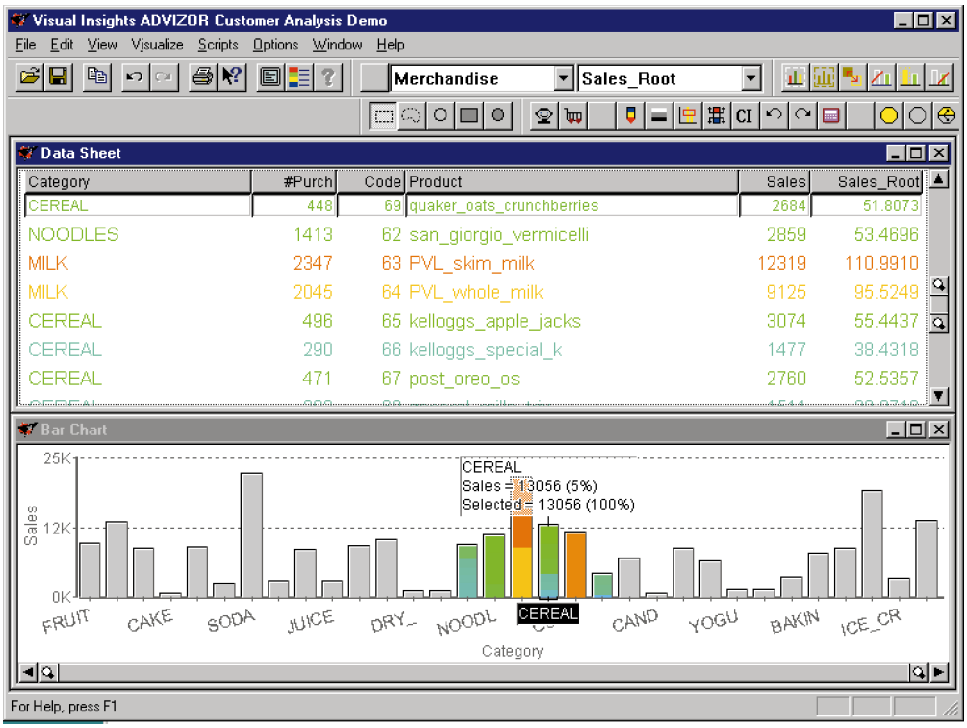

Fig. 8. Bar chart and data sheet showing sales by product are automatically linked so that selection and focus events propagate between the components.

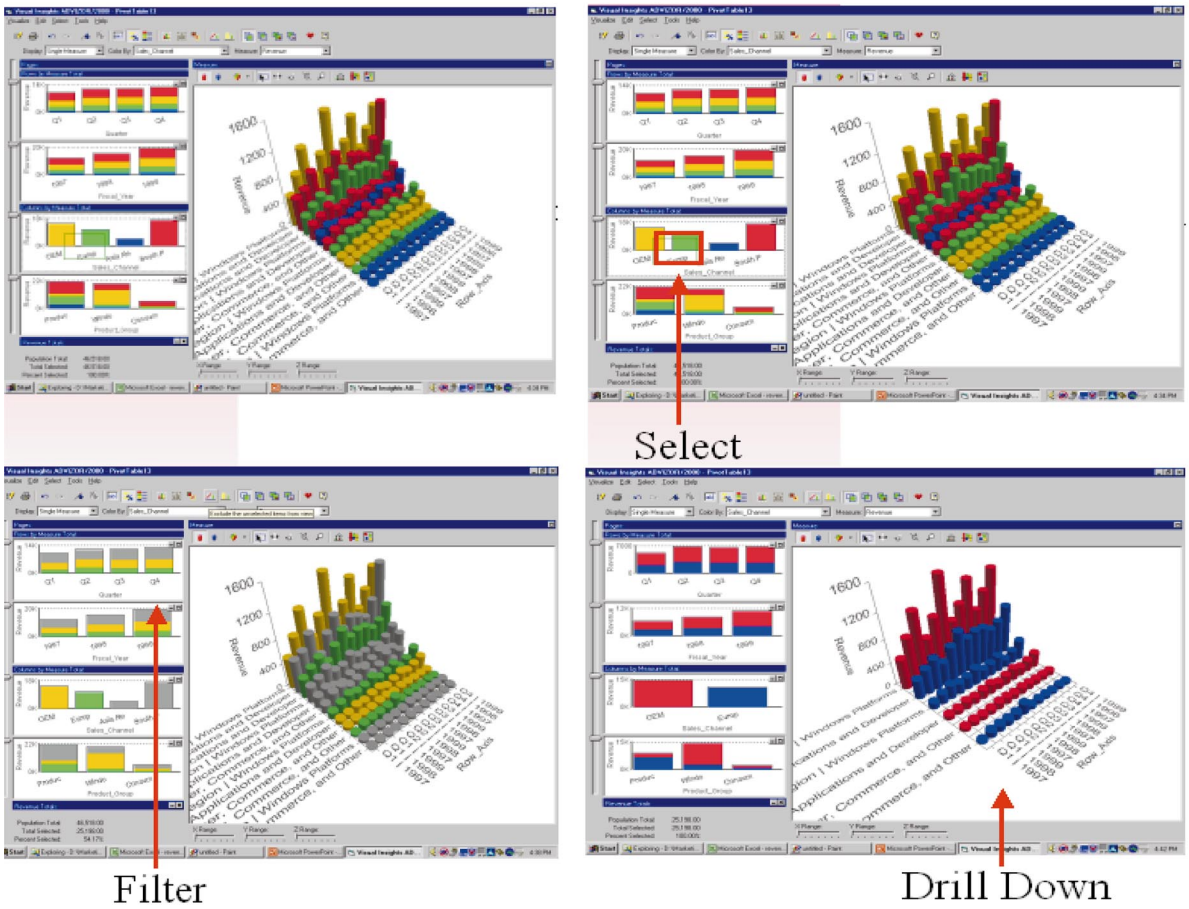

Fig. 9. ADVIZOR/2000 perspectives showing sales by quarter, year, sales channel, and product. Upper left: initial perspective. Upper right: user selects `OEM` and `Europ` sales channels. Lower left: filtering results shown on perspective. Lower right: excluding restricts the display to drill down on only the two selected channels.

and quarter. The initial perspective is in the upper left. Selecting the *OEM* and *Europ* channels (upper right) shows sales for these channels (lower left). Excluding focuses the perspective on only these two selected channels (lower right).

Excluding and *restoring* are particularly useful for eliminating bad data from a perspective, e.g., all cases corresponding to the *unknown* bar in the male-female bar chart.

2.2.3 Focus

Focus provides a means for users to see the value of data items. Mousing over a graphical entity corresponding to a row, either *selected* or *unselected*, sets the *focus*. By convention, the entity corresponding to the *focus* row is dynamically labeled. *Focus* provides a convenient interface for labeling and interactively identifying interesting entities in the current and all other linked components.

2.2.4 Selection Tool and Modes

The selection tool and mode, described in detail in [21], are extremely powerful. The most common selection tool is a swept out *rectangle*, but other options include a *circle* with radial sizing and a *lasso* that can assume arbitrary shapes.

The most frequently used *selection mode* is *replace*, which causes the current selection to replace any previous selections. Other possible modes are *toggle* that inverts the selection state of the chosen items, *add* that increases the selection set, *subtract* that decreases the selection set, and *intersect* that subsets the selection set by intersecting the new and existing selections.

2.2.5 Automatic Selection Aggregation

By default, the components automatically recalculate aggregations, roll-ups, and statistical routines every time the selection set changes. Although this is computationally intensive, automatic recalculation facilitates "drill-across" comparisons.

For example, consider a perspective containing four bar charts, respectively, showing *sales* (bar height) by *sex*, by *store location*, by *age*, and by *marital status*. Each component is attached to a data table containing transactions with additional columns containing the purchaser's *sex*, *age*, *marital status*, and *store location*. The bar charts will automatically aggregate over the data table and display sales totaled by category. Sequentially selecting first the males and then the females in the *sex* bar chart shows how sales vary by sex across all of the other variables. Using the intersect selection mode to select from all *males* those whose marital status is *single* shows three-way interactions between variables. A particularly handy technique is to aggregate over a "color-by" variable. The colors are then stacked within each bar, showing the subpopulations involved within the aggregation.

2.2.6 Programming ADVIZOR

There are two ways to program ADVIZOR content, either visually by interactively creating and saving perspectives or by writing Visual Basic scripts that set component properties. A convenient technique is to use `AutoScript.vbs`, which automatically runs at startup by default, to bind scripts to ADVIZOR buttons and to populate the initial data tables. Application-oriented scripts are then available with a single mouse click.

Although VB provides a rich programming environment for ADVIZOR, it is also possible to program ADVIZOR using J++, C++ or any language that supports *COM* bindings. All important attributes are exposed through the component properties.

Adding a new ActiveX component to ADVIZOR, e.g., a map control or a data mining tool, is straightforward. The integration step involved with a tight linkage involves writing message handling code that translates events between ADVIZOR and the new component. The translation should be logical, e.g., if a user focuses on the bar corresponding to `Illinois`, the state *Illinois* should be highlighted in a linked choropleth (color-coded) map.

Some of the more common ADVIZOR programming operations involve manipulating the *data pool*, modifying the color vector, and redefining the context menus. These operations are only available through the programming APIs. Typical *data pool* operations involve adding new rows or columns to an existing table, performing row transforms, and creating new tables.

2.2.7 Embedding Visual Components in Other Applications

ADVIZOR's visual components are easily hosted by other containers, such as MS Internet Explorer™, and that may be integrated into other applications. The components are packaged in four dynamic link libraries (*dll*s), *VzUni*, *VzBi*, *VzMulti*, and *VzLib* containing the univariate, bivariate, multivariate views, and common routines, respectively.

There are three successive levels of component integration:

1. *Graphical widgets*, where the components function as richer and better graphical displays.
2. *Stand alone visual data analysis environment*, where the container hosts the components and exploits their built-in menus, linking, statistical functions, and analysis capacity.
3. *Visual controls* integrated into an application and functioning both as a visual display and interactive environment for manipulating the application.

In embedded applications, it is the responsibility of the container to populate the data pool, either directly through the API or by accessing the *SDR/W*, described in Section 3.2. For an integration involving visual controls, the hosting application must write code that maps ADVIZOR events to the appropriate control actions.

2.2.8 ADVIZOR/2000—End User Visual Workspace

We have developed two container applications, ADVIZOR and ADVIZOR/2000, that provide a rich environment for building interactive visual query and analysis applications. ADVIZOR/2000, shown in Fig. 1, focuses on visualizing multidimensional data. It provides an out-of-the-box end user experience for navigating, selecting, manipulating, and analyzing multidimensional data. The ADVIZOR/2000 container hosts two perspectives, a *single measure* perspective for visualizing one measure and a *multiple measures* perspective for visualizing two or three measures simulta-

Fig. 10. ADVIZOR/2000 toolbar supports rich navigation.

neously. It goal is to provide an easy-to-use tool specifically suited for one class of data.

Important features include:

1. *Undo and Redo* that is a standard in Windows applications;
2. *Bookmarking* to save and return to the current state;
3. *Reporting and Printing* for sharing analysis results;
4. *Toolbars* for ease of use and navigation (Fig. 10);
5. *Write-back* to export a result set to an operational system for further analysis or business action.

2.2.9 ADVIZOR—Scriptable Visual Workspace

ADVIZOR is a powerful, general-purpose platform for constructing visual applications. Using ADVIZOR menus and dialogs, users import data tables into the data pool and manipulate the data using the data and date calculator to create new fields and perform other data conditioning operations. Users then instantiate visual components, e.g., bar charts, pies, ..., attach them to the data tables, and organize them into perspectives. This process, called authoring, enables domain knowledge to be captured in reusable perspectives. This process supports arbitrary ad-hoc analysis since visual components can be created and populated as needed for the analysis.

ADVIZOR provides:

1. *Window management* including MDI and splitter windows;
2. *Visual authoring* to interactively build and save perspectives;
3. *Application templates* that capture important problem domain knowledge (see below);
4. *Viewing* for "light weight" users;
5. *Session management* including save and restore;
6. *Undo and Redo* that is becoming standard in Windows applications;
7. *Reporting and Printing* for sharing analysis results;
8. *Toolbars* for ease of use and navigation;
9. *Preferences* for end user customization.

When a set of perspectives is bundled together to solve a business problem, it is called an *Application Template*. Application templates may be customized and modified as needed. A Category Analysis template, for example, might include perspectives answering: significant changes from previous time periods; branded and private label performance; share and growth percentages; emerging new products; and shelf stocking by store sizes and category.

2.3 Visual Design Patterns

In the process of constructing many perspectives, we have observed certain recurring design patterns. These patterns associate perspectives with specific types of problems or analyses. The patterns are broadly useful across many applications and apply to many problem domains.

2.3.1 Overview, Zoom, Filter, Details on Demand

As noticed by Card et al. [4, p. 625], a common design pattern involves an overview that shows the entire dataset, e.g., all folders on your hard disk, and supports the ability to zoom in on interesting folders visible in the overview using direct manipulation techniques. It incorporates interactive filters, frequently bar and pie charts, that enable you to filter out uninteresting folders so that you display only the data that is interesting. Filtering might be by category, numeric range, or even selected value. An effective user interface, for example, might allow you to click on the interesting items and thereby retrieve folder details. This design pattern is called: *Overview, Zoom, Filter, Details on Demand*.

ADVIZOR perspectives using this design pattern frequently use a Data Constellations, Scatterplot, ParaBox, or Time Table with zoombars as the overview, coupled with linked bar charts for filtering, and labeling and perhaps a Data Sheet for details. Examples of other systems using this design pattern include Spotfire (a scatterplot overview) and VisDB [13] (space-filling pixel overview). The overview is the primary workspace, with little if any substantive interpretation coming from the filter graphics.

2.3.2 Linked Bar Charts For Categorical Analysis

Another design pattern, called *Linked Bar Charts* is particularly strong for data tables containing categorical data. The design pattern uses one bar plot for each categorical column, with the height of the bar tied to the number of rows having that particular value. In statistical terms, each of the bar charts shows a marginal distribution. As the user selects an individual bar, the display recalculates to show one-way interactions. Using exclusion and selection shows two-way interactions.

A typical example of a categorical dataset might involve a pizza customer survey. A data table contains one line (row) for each surveyed customer that with many attributes (columns) including:

- *topping satisfaction:* numeric value from 1 and 7;
- *complaint:* no problem, ...;

Fig. 11. Categorical data design pattern. Left: overall satisfaction by attribute. Right: dissatisfaction by attribute.

- *type of crust:* deep dish, ...;
- *topping:*
- *pizza size:* small, medium, large;
- *overall satisfaction:* numeric value from 1 to 7;
- ...

Fig. 11 (left) illustrates this design pattern with bar charts showing satisfaction by attribute. Most customers are highly satisfied since the bar corresponding to a 7 is the longest in the overall satisfaction bar chart. If a user selects any individual bar, e.g., *Deep Dish*, the perspective recalculates to show marginal satisfaction for that attribute (not shown). By sequentially selecting individual or sets of bars, users can explore relationships between the attributes.

The perspective on the right side of Fig. 11 has excluded the satisfied customers and focused in on just those who were somewhat unhappy. Using the perspective further, we can find that the primary reason customers are unhappy is because they are not getting enough toppings. Going further, we discover the reason is that a small number of stores are not putting on enough pepperoni.

In this design pattern, users interact with multiple workspaces. No one bar chart is primary or secondary. User questions dictate the starting point for an analysis and their journey through the rest of the interrelated bar charts.

2.3.3 3D Multiscape with Bar Charts

Another common design pattern uses a 3D Multiscape or landscape visualization to show tabular information with the height of each bar encoding the value of the respective cell. The bars may be organized on a 2D grid or positioned spatially on a geographical substrate. Bar charts and other controls that function as filters are positioned around the landscape visualization or, in some cases, directly on the walls. Users rotate and manipulate the landscape visualization to see patterns and can label interesting bars by touching them with the mouse and can filter using the linked bar charts. See Fig. 1.

Exemplar systems using this design pattern include SDM [5], Wright's information animations [23], and Visual Insights ADVIZOR/2000 [8].

This design pattern is midway between the single and multiple-workspace apprach. The overview graphics—the 3D Multiscape—is a primary workspace to keep the users oriented to solutions in regard to the table population. Yet, users take "side journeys" into the bar charts for substantive intrepretation, not just filtering. With the bar charts as secondary workspaces, user can explain detailed issues while still maintaining a big picture view in the primary 3D workspace.

3 Implementation Details

This section describes some of the interesting aspects of ADVIZOR's construction.

3.1 Data Model

ADVIZOR uses a table-based data model. *VzTable*, shown in Fig. 12, consists of named columns, each of which contains data elements of a single type. Internally, the data within columns is stored in auto-sizing vectors. The indigenous data types are `String`, `32 bit integer`, `doubles`, and `dates`. The meta data for each table includes householding information such as vector names, lengths, access state, etc. The left side of Fig. 12 shows that two additional vectors are automatically associated with every table: a *selection* and *color* vector. Sets of in-memory tables are maintained in the *Data Pool*.

3.1.1 Data Access

ADVIZOR achieves ubiquitous access via the *Structured Data Reader/Writer*, an OLE-component that interfaces with standard sources and can directly populate the data pool. The engineering goal for the SDR/W is to provide ubiquitous data access. The currently supported standards include ODBC, ADO, Excel spreadsheets, and text files.

3.1.2 Linking and Joining Tables

Joining tables within the *data pool* combines multiple tables to make them appear as one. Linking tables propagates events between tables, an especially useful feature for selection, color, and handling real-time data feeds. Both of

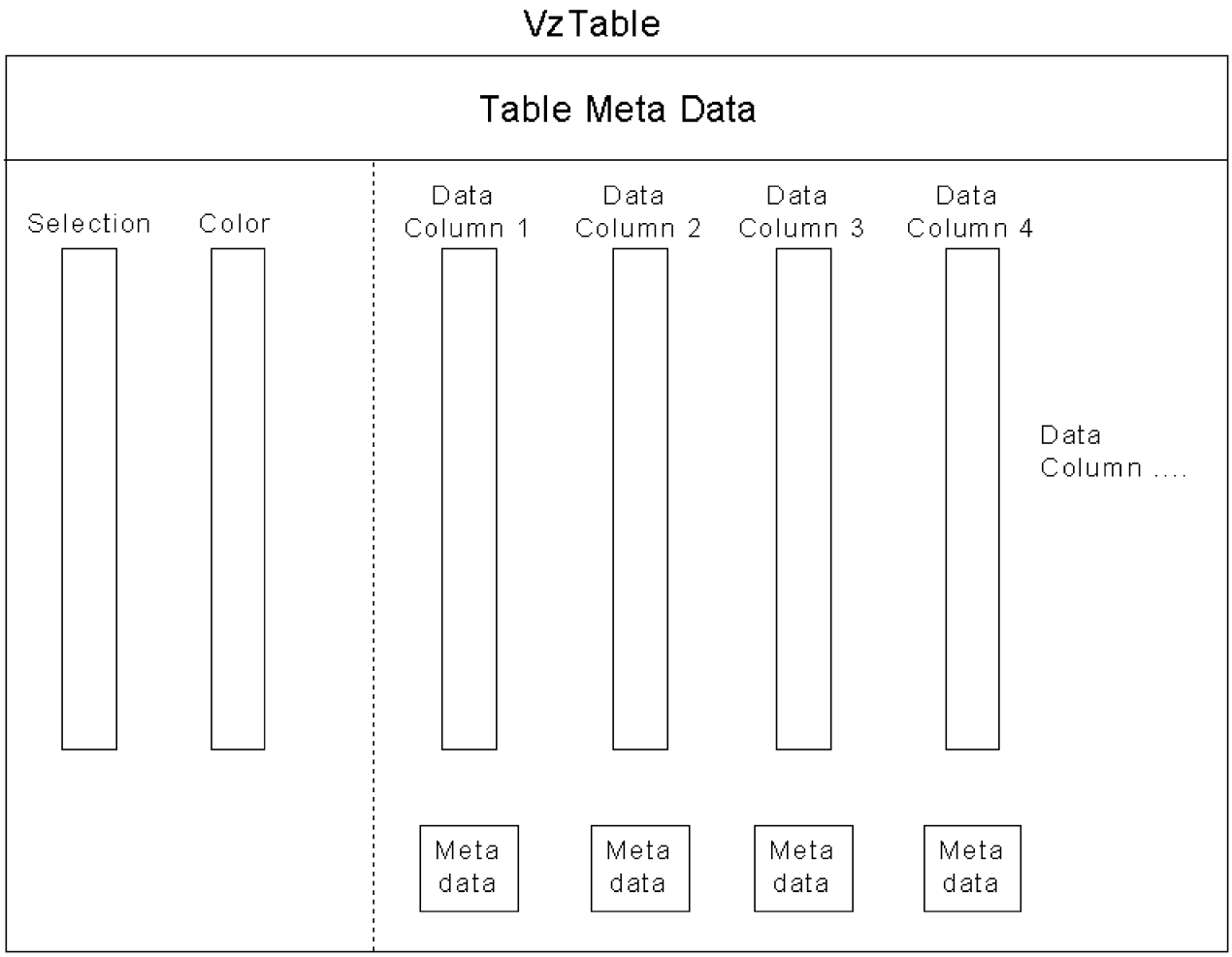

Fig. 12. *VzTable*, ADVIZOR's internal data store.

these features are currently available only through the programming API. Tables may be linked in four ways: selection linking along a key, color linking, aggregation linking, and clone linking. The mechanism ADVIZOR uses for implementing the linking is a general purpose capability called *Connectors*.

There are several significant advantages to maintaining multiple tables in the data pool. First, it simplifies application development. Second, having multiple tables, each representing a different level of aggregation, enables ADVIZOR to navigate through drill-down, drill-up, or drill-across at various levels in a data hierarchy. For example, a retail product management perspective might include transactions, a roll-up of transactions by customer loyalty card number, and a further roll-up of sales by product. Third, by linking the tables together and bringing in data one slice at a time, it is possible to handle arbitrarily large data stores.

3.1.3 Operations on Tables

Tables within the data pool may be manipulated, cloned, edited, transformed, and aggregated. Cloning and aggregating creates a new table. Manipulations are performed either through the programming APIs or by using ADVIZOR's *Calculator*, a general purpose table manipulation tool.

3.1.4 Answering "Why" Questions

Answering "why" questions frequently involves multiple tables. In single table visualization systems, the initial visual display is often sufficient and rich enough to answer the "what happened" question, e.g., sales went up in Nevada, profit increased, beer and diapers are purchased together on Fridays, and so on. Probing more deeply and answering the "why" questions often involves another table. Understanding why sales in Nevada increased might involve linking the initial perspective, showing the sales by state table, with another perspective, showing the sales by product table restricted to Nevada. Answering the why question for beer and diapers might involve another perspective, showing the customer profiles table for beer and diaper purchasers, to discover that young husbands are frequently sent on errands to purchase diapers and buy beer for themselves.

The system architecture needed to answer why questions includes:

1. a multiple table data pool;
2. linking and event passing among the tables so that drill-up, down, and across operations propagate;
3. Perspectives with different visual components showing multiple tables simultaneously.

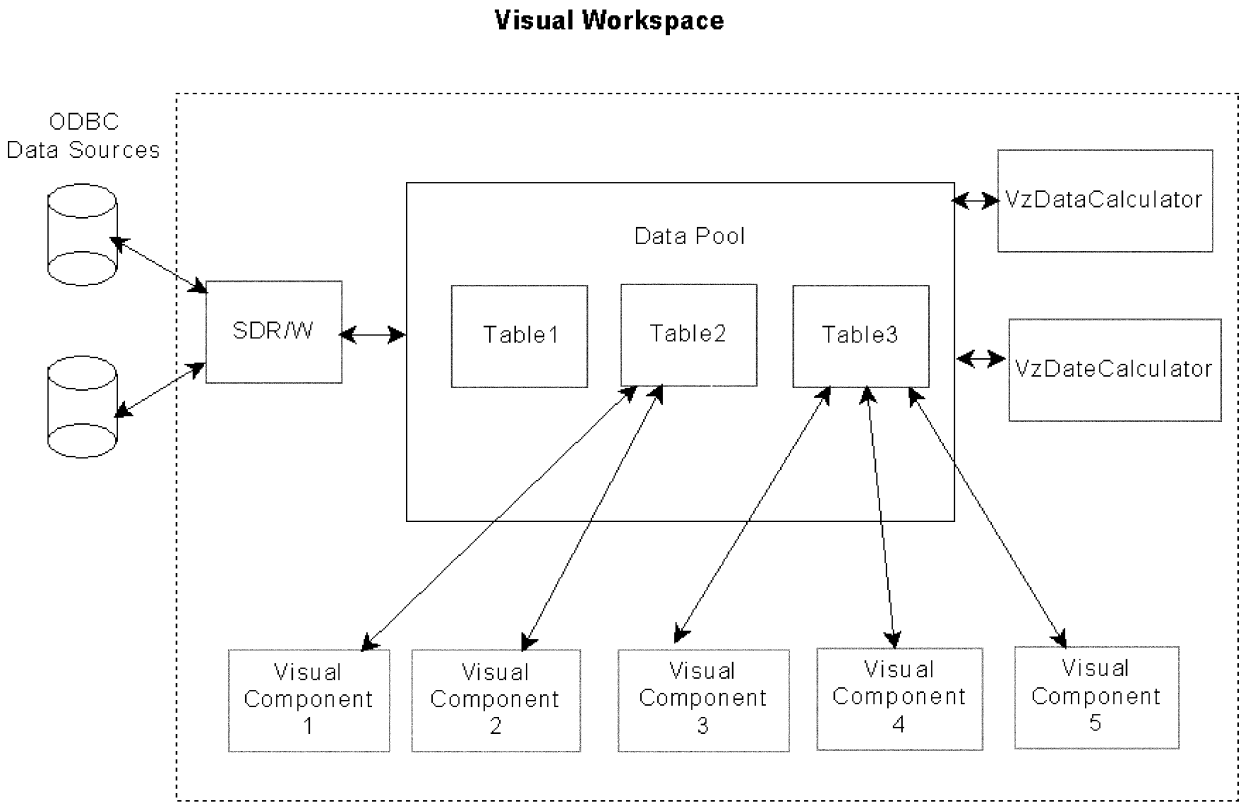

Fig. 13. ADVIZOR software architecture.

3.1.5 Event Model

ADVIZOR uses events to propagate changes, both user and data initiated, among the visual components and data tables. The events are accessible through programming APIs and can be intercepted, generated, and manipulated using Visual Basic code. Every event is associated with a data table and is propagated using a *publish and subscribe* programming pattern. Visual components direct events generated by user manipulations, e.g., selections, excluding, focus, etc., to their associated data table, which then propagates the events to any linked components.

There are seven important event types:

1. *DataChanged* indicates that the underlying data table has been modified and usually causes the component to completely redraw itself;
2. *VisibilityChanged* is triggered for an *excluding* (or *restoring*) event;
3. *HighlightChanged* is called for selection events;
4. *NamesChanged* indicates that a variable name in a table has changed;
5. *ColorChanged* happens when the current color vector changes, either by rescaling the colors or by selecting a new color scale;
6. *FocusItemChanged* occurs when the focus changes;
7. *UndoChanged* occurs when a user has selected either *undo* or *redo* from the command stack.

In contrast to many components that return raw mouse coordinates and button clicks, ADVIZOR events are much higher level and oriented toward visualization tasks. To create an animation, for example, ADVIZOR sequentially creates *HighlightChanged* events and walks the focus over the rows in the data table.

3.2 Software Architecture

As shown in Fig. 13, ADVIZOR consists of five architectural components:

1. *Data Pool*, a set of in-memory data tables available to be visualized and populated directly or via the *Structured Data Reader/Writer* (SDR/W), a component providing broad data access.
2. *Data Manipulation components*, *VzCalculator* and *VzDateCalc*, that manipulate data pool tables performing data, date and calendar calculations, and some statistical metrics.
3. *Interactive Visual Components*, each embodying a particular visual representation, including conventional graphs such as bars, pies, and lines, as well as novel metaphors such as *Data Constellations*. Components both show information and provide built-in analysis capability. Components attached to the same table are automatically linked (see below).
4. *Container Applications*, ADVIZOR and ADVIZOR/2000, that function as visual workspaces and provide an interactive sense-making environment. The containers, targeted at different domains, provide windows services, a GUI, and help support.
5. *A Powerful Event-based Linking Mechanism* that coordinates operations among the components.

At the core of ADVIZOR is a patented C++ class library called Vz. Developed over six years in Bell Laboratories, Vz is an object-oriented, platform-neutral, class library focused on visualization. The Vz library:

- handles display rendering in an efficient manner,
- contains placement and graph layout algorithms,
- factors out common visualization operations such as color management, scales, mouse operation,
- includes many utility classes for efficient data management, manipulation, conditioning, and transforms,
- provides a common "look-and-feel,"
- enables efficient 3D navigation,
- supports view linking and event management,
- includes statistical algorithms, such as smoothing and trending, and
- uses an API-neutral rendering engine, *VzDrawer*, that draws with OpenGL for 3D graphics and Win32 for 2D graphics.

4 Related Work

The related work falls into three broad classes: general papers in information visualization, linked-view statistical graphics applications, and visual component-related research. In all cases, there has been a rich history of interesting and significant contributions. The best general summary of related work is clearly Card et al.'s recent book on information visualization [4]. It contains reprints of many fundamental and sustaining papers in information visualization. Another related collection of papers is [14].

More specifically, some work on linked-view systems which clearly influenced the creation ADVIZOR includes:

Dynamic Statistical Graphics are a collection of early techniques for interacting with statistical plots [2].

MANET is a linked-view system focusing on missing values [18].

Data Desk is a powerful interactive teaching tool that has evolved into a sophisticated commercial statistical package [19].

XGobi an early linked-view visualization system for X-Windows [17].

XmdtTool is another pioneering linked-view system embodying rich multivariate data analysis techniques [20].

Spotfire is a commercial package based on work at the University of Maryland that is targeting the pharmaceutical industry [1].

Research related to our visual components includes:

Data Sheet is similar to Xerox Parc's Table Lens [15].

Time Table is built from the SeeLog tool that focused on displaying time-stamped events from log files [9].

Multiscape is our version of a landscape visualization introduced in [12] and extended in [11].

Data Constellations extends Wills' work on NicheWorks [22].

5 Summary

ADVIZOR is an interactive environment for building tightly linked visual query and analysis applications. There are three unique and compelling aspects to ADVIZOR's technology:

- Rich, scalable, interactive *Visual Components* that are tightly linked by selection, focus, data, and color.
- *A Data Pool* containing multiple, linkable tables for visualization. A multiple table data pool is necessary to answer "why" questions and provides better support for drill-up, drill-down, and drill-across.
- *ADVIZOR* and *ADVIZOR/2000* containers that host the components and function as visual workspaces.

Together, the different aspects of ADVIZOR function as a powerful environment for visual query and analysis.

Acknowledgments

Many talented and creative thinkers on the Visual Insights staff have contributed to the creation of ADVIZOR. Also, I particularly appreciate insightful comments and my collaborations with Alan Karr and Barbara Mirel. Graham Wills and Ken Cox contributed greatly to an early research version of this technology.

References

[1] C. Ahlberg and B. Shneiderman, "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays" *SIGCHI '94 Conf. Proc.,* pp. 313-317, Apr. 1994.

[2] R.A. Becker, W.S. Cleveland, and A.R. Wilks, "Dynamic Graphics for Data Analysis," *Statistical Science,* vol. 2, pp. 355-395, 1987.

[3] J. Bertin, *Semiology of Graphics.* London: Univ. of Wisconsin Press, Ltd. 1983.

[4] S.K. Card, J.D. Mackinlay, and B. Shneiderman, *Readings in Information Visualiation: Using Vision to Think.* San Francisco: Morgan Kaufman, 1999.

[5] M.C. Chuah, S.F. Roth, J. Mattis, and J.A. Kolojejchick, "SDM: Selective Dynamic Manipulation of Visulizations," *UIST '95 Conf. Proc.,* pp. 6170, 1995.

[6] W.S. Cleveland, *The Elements of Graphing Data.* Pacific Grove, Calif.: Wadsworth, 1985.

[7] S.G. Eick, "Graphically Displaying Text," *J. Computational and Graphical Statistics,* vol. 3, no. 2, pp. 127-142, June 1994.

[8] S.G. Eick, "Visualizing Multi-Dimensional Data with Advizor/2000™," Aug. 1999, available at: http://www.visualinsights.com.

[9] S.G. Eick and P.J. Lucas, "Displaying Trace Files," *Software Practice and Experience,* vol. 26, no. 4, pp. 399-409, Apr. 1996.

[10] S.G. Eick and G.J. Wills, "High Interaction Graphics," *European J. Operational Research,* vol. 81, pp. 445-459, 1995.

[11] J. Goldstein, S.F. Roth, J. Kolojejchick, and J. Mattis, "A Framework for Knowledge-Based Interactive Data Exploration," *J. Visual Languages and Computing,* vol. 5, pp. 339-363, Dec. 1994.

[12] W.C. Hill and J.D. Hollan, "Deixis and the Future of Visualization Excellence," *IEEE Visualization '91 Conf. Proc.,* pp. 314-320, Oct. 1991.

[13] D.A. Keim and H.-P. Kriegel, "VisDB: Database Exploration Using Multidimensional Visualization," *IEEE Computer Graphics and Applications,* vol. 14, no. 5, pp. 40-49, Sept. 1994.

[14] *Database Issues for Data Visualization,* J.P. Lee and G.G. Grinstein, eds. Springer-Verlag, Oct. 1994.

[15] R. Rao and S.K. Card, "Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus Plus Context Visualization for Tabular Information" *Proc. ACM Conf. Human Factors in Computing Systems (CHI '94),* pp. 318-322, Apr. 1994.

[16] B. Shneiderman, "Tree Visualization with Tree-Maps: A 2-Dimensional Space Filling Approach," *ACM Trans. Graphics,* vol. 11, no. 1, pp. 92-99, Jan. 1992.

[17] D.F. Swayne, D. Cook, and A. Buja, "XGOBI: Interactive Dynamic Data Visualization in the X Window System," *J. Computational and Graphical Statistics,* vol. 7, no. 1, June 1998.
[18] A. Unwin, G. Hawkins, H. Hofmann, and B. Siegl, "Interactive Graphics for Data Sets with Missing Values—Manet," *J. Computation and Graphical Statistics,* vol. 5, no. 2, pp. 113-122, 1996.
[19] P.F. Velleman, *The DataDesk Handbook.* Odesta Corp., 1988.
[20] M.O. Ward, "Xmdvtool: Integrating Multiple Methods for Visualizing Multivariate Data," *Visualization '94 Conf. Proc.,* pp. 326-333, Oct. 1994.
[21] G.J. Wills, "Selection: 524,288 Ways to Say 'This is interesting,'" *Information Visualization '96 Proc.,* pp. 54-60, Oct. 1996.
[22] G.J. Wills, "Nicheworks—Interactive Visualization of Very Large Graphs," *J. Computational and Graphical Statistics,* vol. 8, no. 2, pp. 190-212, June 1999.
[23] W. Wright, "Information Animation Applications in Capital Markets," *IEEE InfoVis '95 Symp. Proc.,* pp. 19-25, 1995.

Stephen G. Eick received a BA from Kalamazoo College (1980), an MA from the University of Wisconsin at Madison (1981), and a PhD in statistics from the University of Minnesota (1985). He is the vice president of research and development for Visual Insights, an emerging growth software company. He and his colleagues have developed a suite of visualizations, including tools for displaying geographic and abstract networks, software source code, text corpora, log files, program slices, and relational databases. Dr. Eick is an active researcher, is widely published, and holds several software patents. He is particularly interested in visualizing e-commerce databases, web analysis, and building high-interaction user interfaces.