Chapter 5
Interaction Support

> A graphic is not "drawn" once and for all; it is "constructed" and reconstructed until it reveals all the relationships constituted by the interplay of the data. The best graphic operations are those carried out by the decision-maker himself.
>
> Bertin (1981, p. 16)

The previous chapter discussed various options for designing visual representations that help people in understanding time and time-oriented data. 'Seeing' trends, correlations, and patterns in a visual representation is indeed a powerful way for people to gain knowledge from data. However, we have also seen earlier in this book that manifold aspects are involved in the generation of a visual representation: characteristics of time and data, user tasks, choice and parametrization of visualization techniques, and so forth. Particularly when feeding unknown data into a visualization method, the visual outcome might not turn out as expected. But we usually do not know exactly what it is that is expected or whether the visual representation is effective with regard to the task that the user is trying to accomplish. And because there are things that we do not know, we have to seek assistance from the user. So, visual exploration and analysis is not a one-way street where data are transformed into images, but in fact is a human-in-the-loop process controlled and manipulated by one or more users.

Having said that, it becomes clear that in addition to visual methods, a high degree of interactivity and advanced interaction techniques for working with time-oriented data are important. Interaction helps users in understanding the visual mapping, in realizing the effect of visualization parameters, in carving out hidden patterns, and in becoming confident about the data. But interaction also provokes curiosity – users want to get their hands on their data – which is particularly useful when exploring unknown data. The importance of interaction is nicely reflected in the following statement:

> Visual representations alone cannot satisfy analytical needs. Interaction techniques are required to support the dialogue between the analyst and the data.
>
> Thomas and Cook (2005, p. 30)

W. Aigner et al., *Visualization of Time-Oriented Data*,
Human-Computer Interaction Series, DOI 10.1007/978-0-85729-079-3_5,

5.1 Motivation & User Intents

The constantly increasing size and complexity of today's datasets are major challenges for interactive visualization. Large datasets cannot simply be loaded to limited computer memory and then be mapped to an even smaller display. Users are only able to digest a fraction of the available information at a time. Complex data contain many different aspects and may stem from heterogeneous sources. As complexity increases, so does the number of questions that one might ask about the data and that should be answered with the help of visual representations.

In our particular case, we need to account for the specific aspects of time and time-oriented data in the context of what, why, and how they are visualized (see Chapters 3 and 4). Any attempt to indistinctively encode all facets of a complex time-oriented dataset into a single visual representation is condemned to fail, because it would lead to a confusing and overloaded display that users could hardly interpret.

Instead, the big problem has to be split into smaller pieces by focusing on relevant data aspects and particular tasks per visual representation. Several benefits can be gained: computational costs are reduced in a kind of divide-and-conquer way, the visual representations become more effective because they are tailored to emphasize a particular point, and users find it easier to explore or analyze the data since they can concentrate on important and task-relevant questions.

Dividing the visualization problem and separating different aspects into individual views raises the question of how users can visually access and mentally combine these. The answer is *interaction*. In an iterative process, the user will focus on different parts of the data, look at them from alternative perspectives, and will seek answers to diverse questions. Starting with an overview of the data, which requires appropriate data abstraction and dedicated visual representations, the user will most certainly identify parts to focus on for more detailed examination (see Shneiderman, 1996). From there, it might make sense to move on to data that are related or similar, or it might be better to return to the overview and investigate the data from a different point of view, or with regard to a different question. In other words, the user forms a mental model of the data by interactively navigating from one focus to the next, where the term focus includes data subsets, data aspects, analysis tasks, and so forth. While exploring data this way, users develop understanding and insight.

The general motivation for interaction is clear now, but what specifically motivates the user to interact? Yi et al.'s (2007) study on a deeper understanding of interaction gives an answer to this question. They identified several user intents for interaction and introduced a list of categories that describe on a high level why users would like to interact. In the following, we make use of the categories from Yi et al. (2007) and adapt them to the case of interacting with time and time-oriented data:

Select – Mark something as interesting When users spot something interesting in the visual representation, they want to mark and visually highlight it as such, be it to temporarily tag an intriguing finding or to permanently memorize important analysis results. The subjects to be marked can be manifold: interesting points in

time, an entire time-dependent variable, a particular temporal pattern, or certain identified events.

Explore – Show me something else In order for the visualization of larger or more complex multivariate time-oriented data to be practically usable, it has to focus on only a subrange of time and on only a subset of the data variables. As a consequence, users have to interactively visit different parts of the time domain and consider alternative variables for the inclusion in the visual encoding to arrive at an overall view of the data.

Reconfigure – Show me a different arrangement Different arrangements of time and associated data can communicate completely different aspects, a fact which becomes obvious when recalling the distinction between linear and cyclic representations of time. As users want to look at time from different angles, they need to be provided with facilities that allow them to interactively generate different spatial arrangements of time-oriented data.

Encode – Show me a different representation Similarly to what was said about the spatial arrangement, the visual encoding of data values has a major impact on what can be derived from a visual representation. Because data and tasks are versatile, users need to be able to adapt the visual encoding to suit their needs, be it to carry out localization or comparison tasks, or to confirm a hypothesis generated from one visual encoding by checking it against an alternative one.

Abstract/Elaborate – Show me more or less detail During visual analysis, users need to look at certain things in detail, while for other things schematic representations are sufficient. The hierarchically structured levels of granularity of time, where higher-level abstractions provide aggregated overviews, and lower levels hold the corresponding details, are a natural match to drive such an interactive information drill-down into time-oriented data.

Filter – Show me something conditionally When users search for particular information in the data or evaluate a certain hypothesis about the data, it makes sense to restrict the visualization to show only those data items that adhere to the conditions imposed by the search criteria or the hypothesis' constraints. Interactively filtering out or attenuating irrelevant data items clears the view for users to focus on their current task.

Connect – Show me related items When users make a potentially interesting finding in the data, they usually ask themselves whether similar or related discoveries can be made in other parts of the data. So users intend to interactively find, compare, and evaluate such similarities or relations, for example, to see whether a trend they discovered in one season of a year is present for other data variables or is repeated at the same time in subsequent years.

Undo/Redo – Let me go to where I have already been Users have to navigate in time and look at it at different levels of granularity, they have to try different arrangements and visual encodings, and they have to experiment with filtering con-

ditions and similarity thresholds. To account for the explorative nature of interactive analytic reasoning, a history mechanism with undo and redo operations is needed. Undo/redo enables users to try out new views on the data and to return effortlessly to the previous visual representation if the new one did not work out as expected.

Change configuration – Let me adjust the interface In addition to adapting the visual representation to the data and tasks at hand, users also want to adapt the overall system that provides the visualization. This includes adapting the user interface (e.g., the arrangement of windows or the items in toolbars), but also the general management of system resources (e.g., the amount of memory to be used).

Taken together, these intents represent what a visualization system for time-oriented data should support in terms of interaction in order to take full advantage of the synergy of the human's and the machine's capabilities. Many of the approaches we describe in the survey in Chapter 7 offer support for one or the other user intent. While marking (or selecting) interesting data items and navigation in time are quasi-mandatory, facilities for other intents are often rudimentary or not considered at all. This is most likely due to the extra effort one has to expend for implementing effective interaction methods. But in fact, all of these user intents are equally important and corresponding techniques should be provided.

5.2 Fundamental Principles

Now that we know about the general motivation and more specific user intents behind interaction, we can move on and take a look at how interaction is actually performed. To this end, we will describe fundamental interaction concepts next.

Conceptual considerations

When users interact they express their intent to change their point of view and they expect that the visual representation reflects the intended change. Norman (2002) models interaction as a loop of two gulfs: the gulf of execution and the gulf of evaluation. The first part subsumes steps that are related to the execution of interaction, including the intention, the mental construction of an interaction plan, and the actual physical interaction (e.g., pressing a button). The second part is related to the feedback, which in our case is of a visual nature. It involves perceiving and understanding the feedback as well as evaluating the success of the interaction. Figure 5.1 illustrates Norman's conceptual model.

Different modes of interaction can be identified depending on how the interaction loop is performed. Spence (2007) distinguishes two modes of active user interaction:

- stepped interaction and
- continuous interaction.

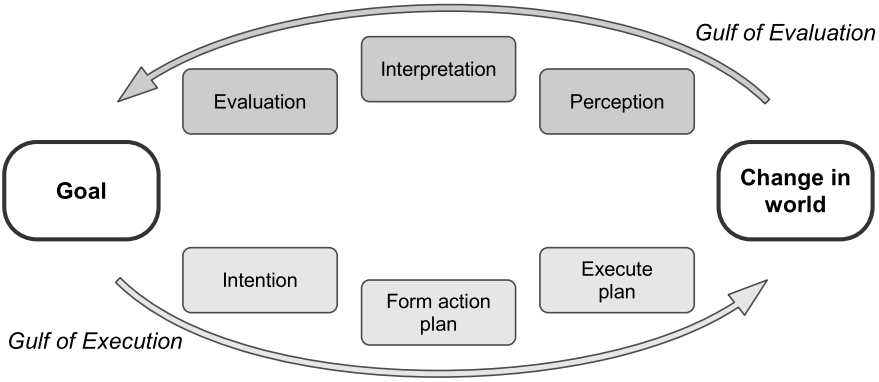


Fig. 5.1: Norman's model of interaction (adapted from Norman, 2002).

For *stepped interaction* the interaction loop is executed in a discrete fashion. That is, the user performs one interaction step and evaluates the visual feedback. Much later the user might perform another step of interaction. As an example one can imagine a user looking at a visualization of the data at the granularity of years. If more details are required, the user might take an interaction step to switch to a finer granularity of months.

The term *continuous interaction* is used to describe interaction for which the loop is iterated at high frequency. The user continuously performs interaction and evaluates feedback for a significant period of time. This is particularly useful, because examining multiple 'what if' scenarios is a key aspect of exploratory analysis of time-oriented data.

An example could refer to setting the cycle length for a spiral visualization in order to find cyclic patterns. For stepped interaction, the user has to explicitly specify the cycle length of interest in a successive manner. The stepped approach requires much time to explore parameter ranges and one does not see patterns emerge as a suitable cycle length is approached. Continuous interaction (e.g., by dragging a slider) allows the user to explore any range at any speed and reduces the risk of losing interesting patterns.

Technical considerations

Technically, Jankun-Kelly et al. (2007) model the loop of user interaction as adjustments of visualization parameters, where concrete parameters can be manifold, e.g., the rotation angle of a 3D helix glyph, the focus point of a fisheye-transformed time axis, thresholds of a filter operation, or parameters that control a clustering algorithm.

For smooth and efficient interaction, the ensemble of visual and interaction methods has to generate feedback in a timely manner (within 50 - 100 ms according to

Shneiderman (1994) and Spence (2007)). However, even data of moderate size can pose computational challenges. On the one hand, mapping and rendering the visual representation might take some time, particularly if complex visual abstractions have to be displayed. On the other hand, analytical methods (see Chapter 6) involved in the visualization process consume processing time before generating results. The adverse implication for interaction is that visual feedback might lag, disrupting the interaction loop.

Another aspect adds to the time costs for presenting visual feedback. As interaction involves change, we want users to understand what is happening. However, abrupt changes in the visual display will hurt the mental model that users are developing while exploring unknown data. Pulo (2007) and Heer and Robertson (2007) provide evidence that smoothly interpolating the parameter change and applying animation to present the visual feedback is a better solution. However, animation consumes time as well, not to mention the possibly costly calculations for interpolating parameter changes.

Thus there are two conflicting requirements. On the one hand, interaction needs synchrony. An interactive system has to be responsive at all times and should provide visual feedback immediately. From the interaction perspective, a system that is blocked and unresponsive while computing is the worst scenario. On the other hand, interaction needs asynchrony – for both generating the feedback (i.e., computation) and presenting the feedback (i.e., animation). The difficulty is to integrate synchrony and asynchrony.

In order to utilize the power of the human-in-the-loop, Norman's model must be outfitted with intuitive interaction methods to allow users to change the visualization, and with intuitive ways of presenting the visual feedback to reflect the change. From a technical perspective, both the gulf of execution and the gulf of evaluation must work together smoothly and seamlessly under the umbrella of an effective user interface.

User interface

The user interface is the channel through which a human and a machine exchange information (i.e., interaction input and visual feedback). This interface is the linchpin of interactive visual exploration and analysis of time-oriented data. Any visual representation is useless if the user interface fails to present it to the user in an appropriate way, and the diversity of available visualization techniques lies idle if the user interface fails to provide interactive access to them. In order to succeed, the user interface has to bridge the gap between the technical aspects of a visualization approach and the users' mental models of the problems to be solved. In this regard, Cooper et al. state:

> [...] user interfaces that are consistent with users' mental models are vastly superior to those that are merely reflections of the implementation model.
>
> Cooper et al. (2007, p. 30)

The user interface is responsible for numerous tasks. It has to provide visual access to time-oriented data and to information about the visualization process itself at different levels of graphical and semantic detail. Appropriate controls need to be integrated to allow users to steer exploration and analysis with regard to the interaction intents mentioned before, including marking interesting points in time, navigating in time at different levels of granularity, rearranging data items and elements of the visual representation, or filtering for relevant conditions. Moreover, the user interface has to support bookkeeping in terms of the annotation of findings, storage of results, and management of the working history (undo/redo).

In general, the user interface has to offer facilities to present information to the user and to accept interaction input from the user. This separation is reflected in the *model-view-controller* (MVC) architecture by Krasner and Pope (1988), where views provide visual representations of some model (in our case time, time-oriented data, and visualization parameters) and controllers serve for interactive (or automatic) manipulation of the model.

Visualization views Especially the different temporal granularities make it necessary to present the data at different levels of graphical and semantic detail. *Overview+detail* as well as *focus+context* are the key strategies to address this demand. Overview+detail methods present overview and detail separately. The separation can be either spatial or temporal. Spatial separation means that separate views show detail and overview. For example, on the bottom of Figure 5.2, an overview shows the entire time domain at a high level of abstraction. On top of the overview there is a separate detail view, which shows the data in full detail (i.e., detailed planning information), but only for a narrow time interval. Temporal separation means a view is capable of showing any level of detail, but only one at a time. This is

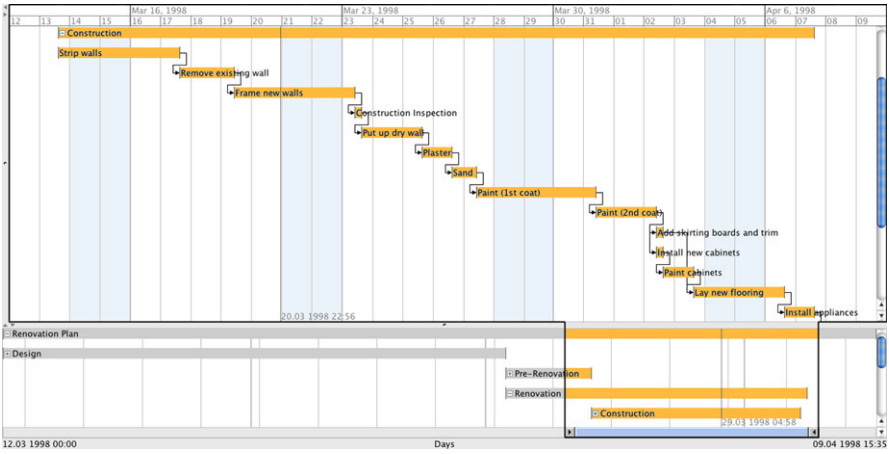


Fig. 5.2: Overview+detail. The detail view at the top shows individual steps of the construction phase of a renovation plan. In the overview at the bottom, the entire project is shown, including the design, pre-renovation, renovation, and construction phases.

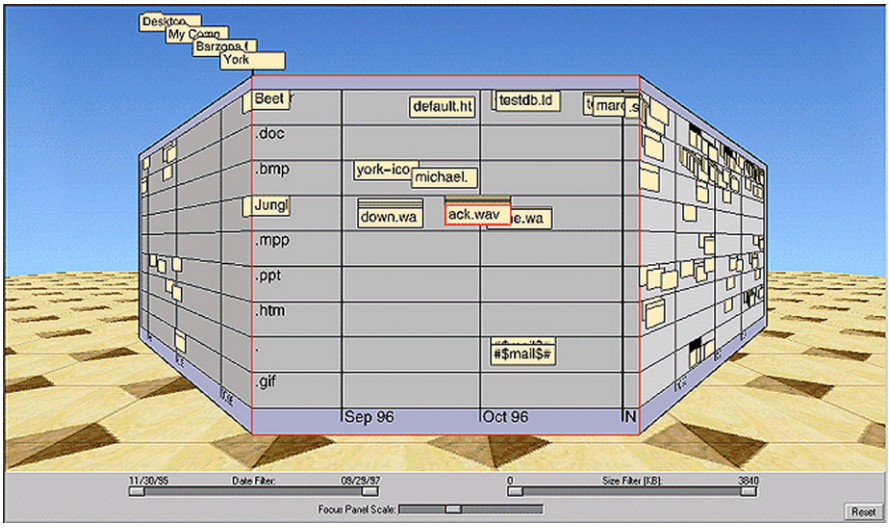


Fig. 5.3: Focus+context. The center of the perspective wall shows the focus in full detail. The focus is flanked on both sides by context regions. Due to perspective distortion, the context regions intentionally decrease in size and show less detail.
Source: © Inxight Federal Systems.

usually referred to as *zooming*, where the user can interactively zoom into details or return to an overview. *Semantic zooming* denotes zooming that is performed in the data space, whereas *graphical zooming* operates in the presentation space. Contrary to overview+detail, focus+context methods smoothly integrate detail and overview. For the user-chosen focus, full detail is presented, and the focus is embedded into a less-detailed display of the context. Figure 5.3 shows the perspective wall technique (↪ p. 168) as a prominent example of the focus+context approach. Cockburn et al. (2009) provide a comprehensive survey of overview+detail, zooming, and focus+context and discuss the advantages and disadvantages of these concepts.

When visualizing time-oriented data, a sensible approach is to provide *multiple coordinated views*[1], each of which is dedicated to particular aspects of time, certain data subsets, or specific visualization tasks. Views are coordinated to help develop and maintain a consistent overall image of the visualized data. This means that an interaction which is initiated in one view is automatically propagated to all coordinated views, which in turn update themselves to reflect the change visually. A practical example is browsing in time. When the user navigates to a particular range of the time axis in one view, all other views (that are coordinated) follow the navigation automatically, which otherwise would be a cumbersome task to be manually accomplished by the user on a per-view basis. Figure 5.4 shows an example where multiple coordinated views are applied to visualize spatio-temporal data in the VIS-STAMP system (↪ p. 244).

[1] Baldonado et al. (2000) provide general guidelines for when to use multiple coordinated views.

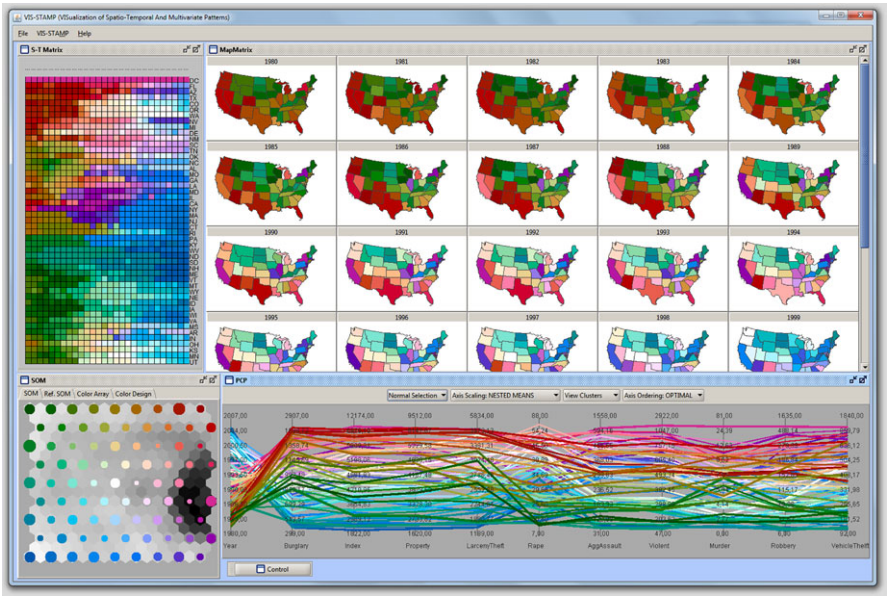


Fig. 5.4: Multiple coordinated views. Analysts can look at the data from different perspectives. The views are coordinated, which means selecting objects in one view will automatically highlight them in all other views as well.
Source: Generated with the VIS-STAMP system.

In addition to coordination, the arrangement of multiple views on the display plays a significant role. The two extremal positions one could take are to use a fixed arrangement that has been designed by an expert and has proved to be efficient, or to provide users with the full flexibility of windowing systems, allowing them to move and resize views arbitrarily. Both extremes have their advantages and disadvantages and both are actually applied. An interesting alternative is to make use of view docking. The main reason for applying docking is that while it maintains flexibility, it also imposes a certain order in terms of what arrangements are possible. For instance, it is preferable that views do not overlap partially; a view should either be visible or not. To this end, docking works with the available screen space being partitioned into regions, each of which contains one or more views. The regions can be resized and moved with the constraint that the arrangement remains a partition, that is, remains overlap-free. A region that contains more than one view provides an interface to switch between them (usually tab-based).

Interaction controls The user interface also consists of various interaction controls to enable users to tune the visualization process to the data and task at hand. Figure 5.5 shows a simple example with only a single spiral view (see Tominski and Schumann, 2008 and ↪ p. 184), which however already depends on a number of parameters, which in turn demands a corresponding number of controls in the control panel. The figure shows sliders for continuous adjustments of parameters such

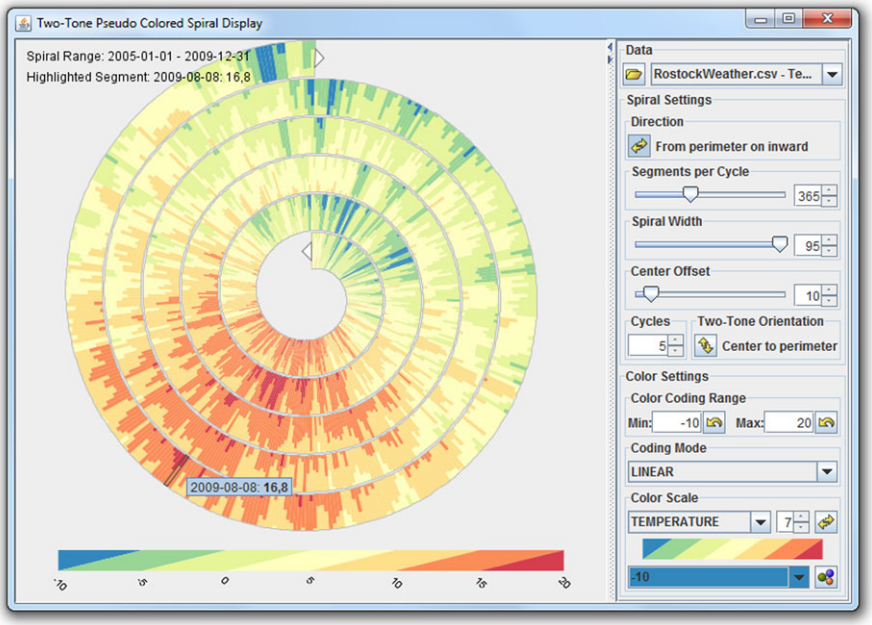


Fig. 5.5: User interface for a spiral visualization. The interface consists of one spiral view and one control panel, which in turn consists of various controls to adjust visualization parameters.

as *segments per cycle*, *spiral width*, and *center offset*. Buttons, drop boxes, and custom controls are provided for selecting different modes of encoding (e.g., adjusting individual colors or choosing different color scales).

In this example, user input (e.g., pressing a button or dragging a slider) is immediately committed to the system in order to realize continuous interaction. However, this puts high demands on the system in terms of maintaining responsiveness and generating visual feedback at interactive rates (see Piringer et al., 2009). Therefore, a commonly applied alternative is to allow users to perform a number of adjustments and to commit the adjustments as a single transaction in a stepped manner only when the user presses the "Apply" button.

Certainly, there are visualization parameters that are adjusted more often than others during interactive visual exploration. Resources should preferably be spent on facilitating continuous interaction for important parameters. Moreover, Gajos et al. (2006) provide evidence that duplicating important functionality from an all-encompassing control panel to an exposed position is a useful way to drive adaptable user interfaces. For example, toolbars allow for interaction that is most frequently used, whereas rarely applied tools have to be selected from an otherwise collapsed menu structure.

5.3 Basic Methods

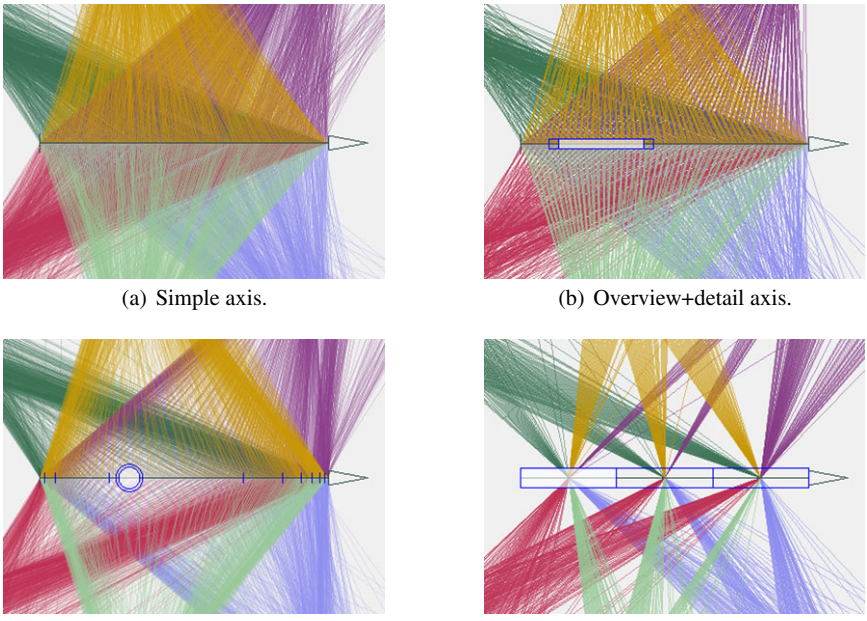
It is clear now that we need visualization views on the one hand, and interaction controls on the other hand. Views are usually equipped with visual representations, and we described many examples for time and time-oriented data in the previous chapters. Let us now take a closer look at interactive means of controlling the visualization beyond standard graphical user interface controls. To this end, we briefly describe *direct manipulation*, *brushing & linking*, and *dynamic queries* as key methods for interactive visualization and show how these methods can be applied for exploration and analysis of time-oriented data.

Direct manipulation

Working with a graphical user interface has the advantage that standardized components can be used to control the visualization process. However, a disadvantage is that visual feedback usually does not appear where the interaction was performed, but in a different part of the display, i.e., the view where the visualization is shown (see Figure 5.5). Direct manipulation as introduced by Shneiderman (1983) is the classic means to address this disadvantage. The goal is to enable users to directly manipulate the visual representation without a detour. To this end, a visualization view or graphical elements are implemented so as to be responsive to user input. A visualization may for instance allow zooming into details under the mouse cursor simply by rotating the mouse wheel, or visiting different parts of the visual representation simply by dragging the view. Such functionality is often present in zoomable user interfaces (see Cockburn et al., 2009). Virtual trackballs (see Henriksen et al., 2004) are more object-centric in that they allow users to grab and rotate virtual objects to view them from different angles.

In terms of interacting with visual representations of time-oriented data, navigating time is of particular importance. Many tools provide standard slider or calendar controls in the user interface to support navigation. However, these controls do not realize direct manipulation. In order to do so, interaction has to be tightly coupled with the display of the time axis. We explain what this means by two examples. Take a look back at Figure 5.5. You may notice that a slider for navigating in time is missing. Instead, the spiral display allows direct manipulation. For this purpose, small arrow-shaped handles are displayed at the beginning and at the end of the spiral. Clicking these handles navigates back and forth in time. Clicking and dragging the mouse off a handle adjusts the navigation speed, which is determined by the distance of the mouse cursor to the handle. A textual label provides precise feedback about the time span currently mapped to the spiral.

For a second example of direct manipulation, we refer to the interactive axes of the TimeWheel (↪ p. 200). Besides a simple non-interactive axis representation, the TimeWheel provides three different types of interactive axes: (1) overview+detail axis, (2) focus+context axis, and (3) hierarchical axis (see Figure 5.6). Each of these interactive axes displays time and additionally offers different options for direct



(a) Simple axis. (b) Overview+detail axis.

(c) Focus+context axis. (d) Hierarchical axis.

Fig. 5.6: The TimeWheel's mapping of time along the time axis can be manipulated directly in different ways. The simple axis uses a fixed linear mapping of time. The overview+detail axis allows users to select any particular range of the time domain to be mapped linearly to the time axis. The focus+context axis can be used to untangle dense parts of the time domain by applying a non-linear mapping. The hierarchical axis represents time at different levels of granularity, where individual axis segments can be expanded and collapsed.

manipulation. The overview+detail axis basically extends the simple axis with three interactive handles to control the position and extent of the time interval to be displayed in the TimeWheel, effectively allowing users to zoom and scroll into any particular part of the data. The focus+context axis applies a non-linear distortion to the time axis in order to provide more drawing space for the user's focus and less space for the context. This allows users to untangle dense parts of the data. Finally, for the hierarchical axis, the display is hierarchically subdivided into segments according to the different granularities of time (e.g., years, quarters, months, days). Users can expand or collapse these segments interactively to view the data at different levels of abstraction. Figure 5.6 illustrates the three types of interactive axes and the corresponding visual mapping compared to the simple non-interactive axis.

The advantage of directly manipulating the visual representation is, as indicated, that interaction and visual feedback take place at the very same location. However, direct manipulation always involves some learning and training of the interaction facilities provided. This is necessary because most of the time the interaction is not standardized but custom-made to fit the visual mapping.

Brushing & linking

Brushing & linking is a classic interaction concept, which takes up the idea of direct manipulation. Becker and Cleveland (1987) describe brushing as a technique that enables users to select interesting data items directly from a data display. There are various options one can follow when selecting data items. We will often find brushing being implemented as point and click interaction to select individual data items. Rubber-band or lasso interaction serve the purpose of brushing subranges in the data or multiple data items at once. Hauser et al. (2002) introduce brushing based on angles between data items, and Doleisch and Hauser (2002) go beyond binary selection to allow for smooth brushing (i.e., data items can be partly selected).

After brushing, selected data items are highlighted in order to make them stand out against the rest of the data. The key benefit of brushing & linking is that data selected in one view are automatically highlighted in all other views, effectively linking the views. In this sense, brushing & linking is a form of coordination among multiple views. This is especially useful when visualizing the variables of a multivariate time-oriented dataset individually in separate views. Then, brushing a temporal interval of interest in one view will highlight the same interval and corresponding data values in all views, and we can easily compare how the individual variables develop during the brushed time period.

For complex data, using a single brush is often unsatisfactory. Instead, users need to perform multiple brushes on different time-dependent variables or in different views. Compound brushing as explained by Chen (2004) allows the combination of individual brushes into composite brushes by using various operators, including logical, analytical, data-centric, and visual operations. With such facilities, brushing is much like a visual query mechanism.

Dynamic queries

Shneiderman (1994) describes dynamic queries as a concept for visual information seeking. It is strongly related to brushing & linking in that the goal is to focus on data of interest, which in the case of dynamic queries is often realized by filtering out irrelevant data. Because time-oriented data are often large, dynamically omitting data with respect to task-specific conditions can be very helpful.

Depending on the view characteristics and visualization tasks, two alternatives can be applied to display filtering results: filtered objects can be displayed in less detail or they can be made invisible. Reducing detail is useful in views that maintain an overview, where all information needs to be displayed at all times, but filtered objects need only to be indicated. Making objects invisible is useful in views that notoriously suffer from cluttering.

Filter conditions are usually specified using dedicated mechanisms. Threshold or range sliders are effective for filtering time or any particular numerical variable; textual filters are useful for extracting objects with specific labels (e.g., data tagged by season). Similar to what has been said for brushing & linking, the next logical

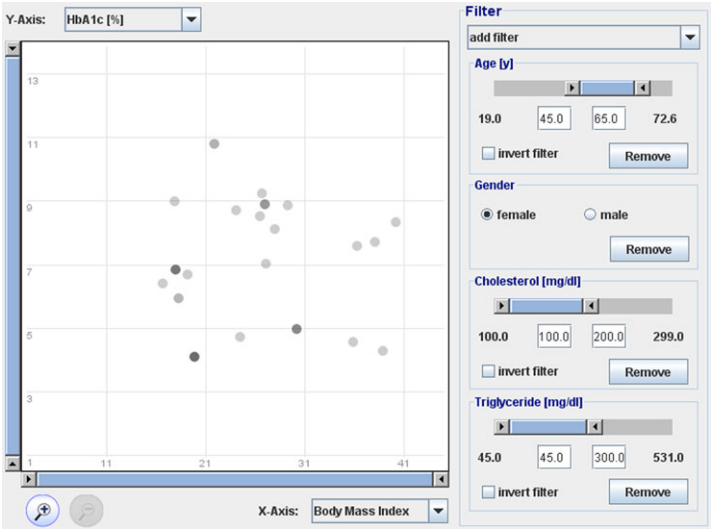


Fig. 5.7: Several filters can be adjusted in order to dynamically restrict the scatter plot visualization to data items that conform with the formulated conditions.

step is to combine filters to provide some form of multidimensional data reduction. For instance, a logical AND combination generates a filter that can be passed only if an object obeys all filter conditions; an object can pass a logical OR filter if it satisfies any of the involved filter conditions. Figure 5.7 shows an example of a dynamic query interface.

While some systems offer only fixed filter combinations or require users to enter syntactic constructs of some filter language, others implement a visual interface where the user can interactively specify filter conditions. Examples for querying time-oriented data that are visualized as line plot (↪ p. 153) are the time boxes by Hochheiser and Shneiderman (2004) and the relaxed selection techniques by Holz and Feiner (2009).

Time boxes are used to filter out variables of a multivariate line plot. To this end, the user marks regions in the visual display by creating one or more elastic rectangles that specify particular value ranges and time intervals. The system then filters out all variables whose plots do not overlap with the rectangles, effectively performing multiple AND-combined range queries on the data. Figure 5.8 depicts a query that combines three time boxes to restrict the display to stocks that performed well in the first and the last weeks of the year, but had a bad performance in the middle of year.

The relaxed selection techniques are useful for finding specific patterns in the data. For that purpose, the user specifies a query pattern by sketching it directly on the display. When the user is performing the sketching, either the distance of the sketch to the line plot or the user's sketching speed are taken into consideration in order to locally relax the query pattern. This relaxation is necessary to allow for a

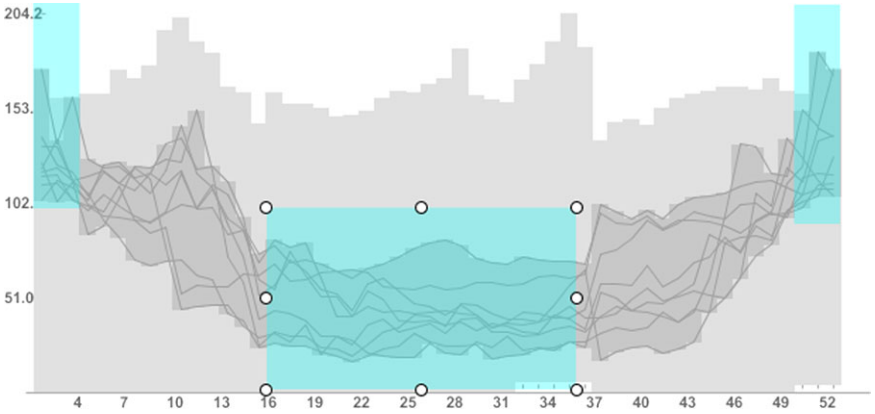


Fig. 5.8: Three time boxes are used to dynamically query stock data. Only those stocks are displayed that are high at the beginning, but low in the middle, and again high at the end of the year.
Source: Generated with the TimeSearcher software.

certain tolerance when matching the pattern in the data. An interactive display of the query sketch can be used to fine-tune the query pattern. Once the query pattern is specified, the system computes corresponding pattern matches and displays them in the line plot as depicted in Figure 5.9.

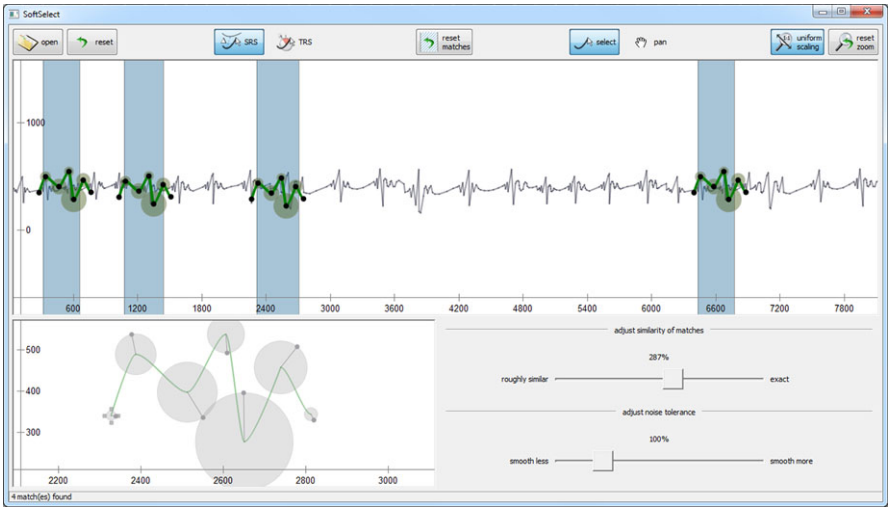


Fig. 5.9: The user can sketch a query pattern directly in the line plot and optionally refine it locally in a dedicated query view. The line plot then shows where in time the query matches with a certain tolerance.
Source: Image courtesy of Christian Holz.

Specifying dynamic queries visually as illustrated by the previous two examples is definitely very useful. However, the user can sketch or mark only those things which are displayed. Formulating queries with regard to potential but not yet existing patterns in the data beyond some tolerance requires additional formal query languages, whose expressiveness, in turn, depends on the formalism used.

Direct manipulation, brushing & linking, and dynamic queries are vital for effective exploration and analysis of time-oriented data. Although these concepts have existed for quite some time now, many visualization tools offer only a fraction of what is possible. Again one can find a reason for that in the higher costs accruing from designing and implementing efficient interaction methods. Moreover, because visual and interactive means must be coupled tightly, it is difficult to develop interaction techniques that can be interchanged among the different visualization techniques for time-oriented data. Finding a solution to this problem is an open research question.

5.4 Integrating Interactive and Automatic Methods

So far in this chapter, we have shown that interaction is mandatory to allow users to parameterize the visualization according to their needs and tasks. However, with increasing complexity of data and visualization methods alike, it is not always easy for users to find parameter values that suit the analysis task at hand. Particularly if parameters are not self-explanatory, they are not easy-to-set manually. So, some form of support is needed to assist users in the parametrization process.

A possible solution is to employ the concept of *event-based visualization*, which combines visualization with event methodology (see Reinders et al., 2001; Tominski, 2011). In diverse application fields, including active databases, software engineering, and modeling and simulation, events are considered happenings of interest that trigger some automatic actions. In the context of visualization, such an event-action-scheme is useful for complementing manual interaction with automatic parametrization of visual representations.

The basic idea of event-based visualization is (1) to let users specify their interests, (2) to detect if and where these interests match in the data, and (3) to consider detected matches when generating the visual representation. This general procedure requires the three main components: (1) *event specification*, (2) *event detection*, and (3) *event representation*. Figure 5.10 illustrates how they are attached to the visualization pipeline. Next we will look at each of these components in more detail.

Describing user interests

The event specification is an interactive step where users describe their interests as *event types*. To be able to find actual matches of user interests in the data, the event specification must be based on formal descriptions. Tominski (2011) uses elements

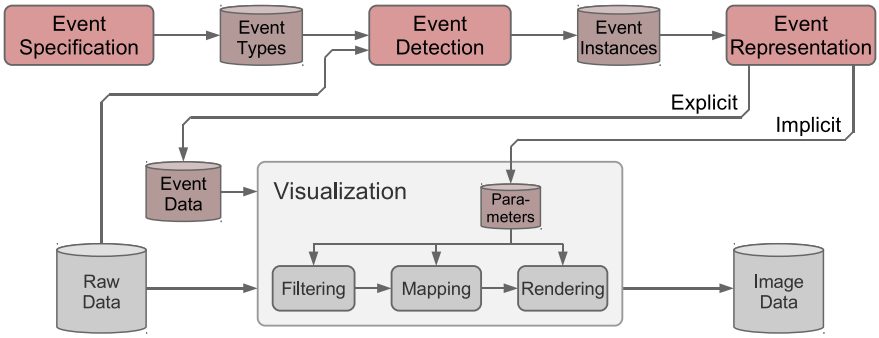


Fig. 5.10: The main ingredients of event-based visualization – event specification, event detection, and event representation – attached to the visualization pipeline.

of predicate logic to create well-defined event formulas that express interests with respect to relational datasets (e.g., data records whose values exceed a threshold or attribute with the highest average value). For an analysis of time-oriented data, sequence-related notations (for instance as introduced by Sadri et al., 2004) have to be added to enable users to specify conditions of interest regarding temporally ordered sequences (e.g., sequence of days with rising stock prices). A combination of event types to composite event types is possible via set operators.

To give a simple example of a sequence event type, we formulate the interest: *"Find three successive days where the number of people suffering from influenza increases by more than 15% each day."* This interest is expressed as:

$$\{(x,y,z)_{date} \mid z.flu \geq y.flu \cdot 1.15 \wedge y.flu \geq x.flu \cdot 1.15\}$$

The first part of the formula defines three variables $(x,y,z)_{date}$ that are sequenced by date. To express the condition of interest, these three variables are set into relation using predicates, functions, and logical connectors.

Certainly, casual users will have difficulties in describing their interests by using event formulas directly. Sufficient specification support starts with providing individual means for experts, regular users, and visualization novices. In this regard, one can think of three different levels of specification: *direct specification*, *specification by parametrization*, and *specification by selection*. Although all levels are based on the same formalism, the complete functionality of the formalism is available only to expert users at the level of direct specification. The idea for the second level is to hide the complexity of event formulas from the user. To this end, parameterizable templates are provided. The user can adjust the templates via easy-to-set parameters, but otherwise has no access to the possibly complicated internal event formula. The amount of increase in the previous event type example is a good candidate for a template parameter. The increase can then be changed from 15% to any particular value without rephrasing the entire event type formula. The third level of event specification is based on simple selection. The idea is to provide a predefined collec-

tion of event types that are particularly tailored to the application context, and that are equipped with expressive labels and descriptions, so that users can easily select what they are interested in. It is also helpful to enhance the event collection with a semantic structure (e.g., by grouping the collection with respect to different user tasks). Devising such a semantic structure and describing event types expressively is a task for domain experts.

Finding relevant data portions

The event detection is an automatic step that determines whether the interests defined interactively are present in the data. The outcome of the event detection is a set of *event instances*. They describe where in the data interesting information is located. That is, entities that comply with user interest are marked as event instances. For event detection, the variables used in event formulas are substituted with concrete data entities. In a second step, predicates, functions, and logical connections are evaluated, so that the event formula as a whole can be evaluated as either true or false. Because this procedure can be very costly in terms of computation time, efficient methods must be utilized for the event detection. A combination of the capabilities of relational database management systems and efficient algorithms (e.g., the OPS algorithm by Sadri et al., 2004) is useful for static data. When dynamic data (i.e., data that change over time, see Section 3.3) have to be considered, detection efficiency becomes crucial. Here, incremental detection methods can help. Such methods operate on a differential dataset, rather than on the whole data. However, incremental methods also impose restrictions on possible event types, because they do not have access to the entire dataset.

Considering user interests in visual representations

The last important step of event-based visualization is the event representation. The goal of this step is to incorporate detected event instances (which reflect the interests of the user) into visual representations. The three requirements that have to be considered are:

1. Communicate the fact that something interesting has been found.
2. Emphasize interesting data among the rest of the data.
3. Convey what makes the data interesting.

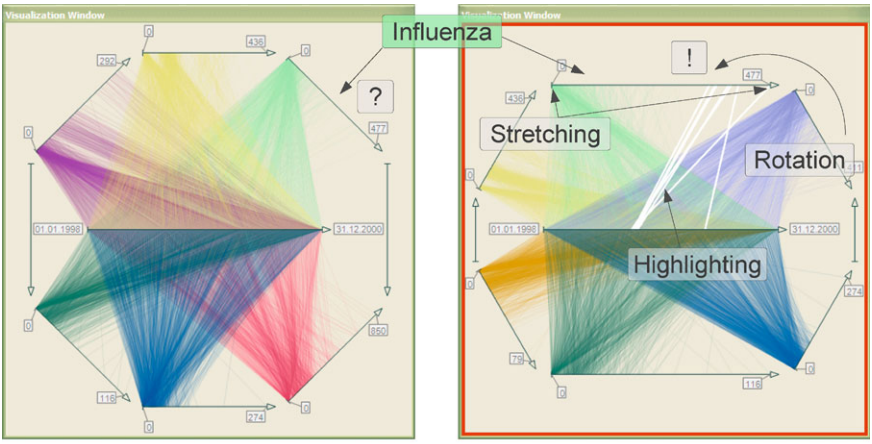
Most importantly, the visual representation must clearly express that something interesting is contained in the data. To meet this requirement, easy-to-perceive visual cues (e.g., a red frame around the visual representation, exclamation marks, or annotations) must be incorporated. Alpha blending can be applied to fade out past events. The second requirement aims at emphasizing those parts of the visual representation that are of interest. Additionally, the visualization should communicate what makes the highlighted parts interesting (i.e., what the particular event type is).

However, when facing arbitrarily definable event formulas, this last requirement is difficult to fulfill.

We can distinguish two basic options for representing events: *explicit* and *implicit* event representation. For the explicit case, the focus is set exclusively on event instances, neglecting the raw data. Since the number of events is usually smaller than the number of data items, even large datasets can be analyzed. In the implicit case, the data context is retained and visualization parameters are set automatically so as to highlight points of interest detected in the data. If we assume that user interests are related to user tasks and vice versa, implicit event representation can help to achieve better targeted visual representations. The big challenge is to meet the above stated requirements solely by adapting visualization parameters. Apparently, availability of adequate visualization parameters is a prerequisite for implicit event representation.

We will illustrate the potential of event-based visualization with an example. Let us assume a user who has to analyze multivariate time-dependent human health data for uncommonly high numbers of cases of influenza. The task at hand is to find out if and where in time these situations have occurred. A possible way to accomplish this task is to use the TimeWheel technique (↪ p. 200).

Figure 5.11(a) shows a TimeWheel that uses the standard parametrization, where time is encoded along the central axis and multiple diagnoses are mapped to the



(a) Default parametrization. (b) Targeted parametrization.

Fig. 5.11: Default vs. targeted parameterization of a TimeWheel. (a) TimeWheel representing a time-dependent health dataset using the default configuration, which aims at showing main trends, but does not consider the interests of the user. (b) TimeWheel representing the same data, but matches with the user's interests have been detected and corresponding data are emphasized via highlighted lines and automatic rotation and stretching; the presentation is better targeted to the user's task at hand.

axes surrounding the time axis. In particular, influenza happens to be the diagnosis that is mapped to the upper right axis (light green). Alpha-blending is applied by default to reduce visual clutter. Looking at this TimeWheel, the user can only guess from the labels of the axis showing influenza that there are higher numbers of cases, because the alpha-blending made the particular lines almost invisible (see question mark). Several interaction steps are necessary to re-parameterize the TimeWheel to accomplish the task at hand.

In contrast to this, in an event-based visualization environment, the user can specify the interest *"Find days with a high number of cases of influenza."* as the event type ($\{x \mid x.flu \geq 300\}$). If the event detection step confirms the existence of such events in the data, visualization parameters are altered automatically so as to provide an individually adjusted TimeWheel that reflects the special situation. In our particular example, we switch color and transparency of line segments representing event instances: Days with high numbers of influenza cases are excluded from alpha-blending and are drawn in white. Additionally, rotation and stretching is applied such that the axis representing influenza is moved gradually to an exposed position and is provided with more display space. The application of a gradual process is important in this case to support users in maintaining their mental map of the visual representation. The result of applying parameter changes automatically in response to event instances is depicted in Figure 5.11(b). In this TimeWheel, the identification of days with higher numbers of influenza infections is easy.

5.5 Summary

The focus of this chapter was on interaction. We started with a brief overview of intents that motivate users to interact with the visualization. The most notable intent in the context of time-oriented data is the intent to navigate in time in order to visit different parts of the data. Users also need to view time-oriented data at different levels of detail, because the data are often given at multiple granularities. Further intents are related to interactively adjusting the visual mapping according to data and tasks at hand, and to managing the exploration process.

We explained that interactive visualization is an iterative loop where the computer generates feedback in oder to visually reflect the change that resulted from user interaction. This human-in-the-loop process brings together the computational power of the machine and the intellectual power of human beings. In order to take full advantage of this synergy, an efficient user interface is needed that bridges the gap between the algorithmic structures used for visualizing time and time-oriented data, and the mental models and analytic workflows of users. This also includes tackling technical challenges to guarantee smooth execution of the interaction loop.

This chapter also presented brief descriptions of basic interaction concepts, including direct manipulation, brushing & linking, and dynamic queries. These concepts are vital for data exploration tasks where the user performs an undirected search for potentially interesting data features. But still, the potential of these con-

cepts has not been fully exploited by current visualization techniques for time and time-oriented data. There is room for future work to better adapt existing interaction methods or to develop new ones according to the specific needs of time-oriented data.

The example of event-based visualization indicates that a combination of interactive, automatic, and visual methods is quite useful for generating visual representations that are better targeted for the user's task at hand than standard representations. This holds true as long as users know what they are looking for (i.e., users perform a directed search). Event-based visualization as described here is not suited to automatically mine potential events in time-oriented data. In order to do so, one needs to integrate analytical methods as described in the next chapter.

References

Baldonado, M. Q. W., Woodruff, A., and Kuchinsky, A. (2000). Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 110–119, New York, NY, USA. ACM Press.

Becker, R. A. and Cleveland, W. S. (1987). Brushing Scatterplots. *Technometrics*, 29:127–142.

Bertin, J. (1981). *Graphics and Graphic Information-Processing*. de Gruyter, Berlin, Germany. translated by William J. Berg and Paul Scott.

Chen, H. (2004). Compound Brushing Explained. *Information Visualization*, 3(2):96–108.

Cockburn, A., Karlson, A., and Bederson, B. B. (2009). A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31.

Cooper, A., Reimann, R., and Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley Publishing, Inc., Indianapolis, Indiana, USA.

Doleisch, H. and Hauser, H. (2002). Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 147–154, Plzen, Czech Republic. University of West Bohemia.

Gajos, K. Z., Czerwinski, M., Tan, D. S., and Weld, D. S. (2006). Exploring the Design Space for Adaptive Graphical User Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 201–208, New York, NY, USA. ACM Press.

Hauser, H., Ledermann, F., and Doleisch, H. (2002). Angular Brushing of Extended Parallel Coordinates. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 127–130, Los Alamitos, CA, USA. IEEE Computer Society.

Heer, J. and Robertson, G. (2007). Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247.

Henriksen, K., Sporring, J., and Hornbaek, K. (2004). Virtual Trackballs Revisited. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):206–216.

Hochheiser, H. and Shneiderman, B. (2004). Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration. *Information Visualization*, 3(1):1–18.

Holz, C. and Feiner, S. (2009). Relaxed Selection Techniques for Querying Time-Series Graphs. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 213–222, New York, NY, USA. ACM Press.

Jankun-Kelly, T., Ma, K.-L., and Gertz, M. (2007). A Model and Framework for Visualization Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–369.

Krasner, G. E. and Pope, S. T. (1988). A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49.

Norman, D. A. (2002). *The Design of Everyday Things*. Basic Books (Perseus), New York, NY, USA.

Piringer, H., Tominski, C., Muigg, P., and Berger, W. (2009). A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120.

Pulo, K. (2007). Navani: Navigating Large-Scale Visualisations with Animated Transitions. In *Proceedings of the International Conference Information Visualisation (IV)*, Los Alamitos, CA, USA. IEEE Computer Society.

Reinders, F., Post, F. H., and Spoelder, H. J. (2001). Visualization of Time-Dependent Data with Feature Tracking and Event Detection. *The Visual Computer*, 17(1):55–71.

Sadri, R., Zaniolo, C., Zarkesh, A., and Adibi, J. (2004). Expressing and Optimizing Sequence Queries in Database Systems. *ACM Transactions on Database Systems*, 29(2):282–318.

Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69.

Shneiderman, B. (1994). Dynamic Queries for Visual Information Seeking. *IEEE Software*, 11(6):70–77.

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, Los Alamitos, CA, USA. IEEE Computer Society.

Spence, R. (2007). *Information Visualization: Design for Interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition.

Thomas, J. J. and Cook, K. A. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, Los Alamitos, CA, USA.

Tominski, C. (2011). Event-Based Concepts for User-Driven Visualization. *Information Visualization*, 10(1):65–81.

Tominski, C. and Schumann, H. (2008). Enhanced Interactive Spiral Display. In *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactivity*, pages 53–56, Linköping, Sweden. Linköping University Electronic Press.

Yi, J. S., ah Kang, Y., Stasko, J., and Jacko, J. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231.