Scalable Visualization of Event Data

David J. Taylor[1], Nagui Halim[2], Joseph L. Hellerstein[2], and Sheng Ma[2]

[1] Department of Computer Science, University of Waterloo
Waterloo, Ontario, Canada N2L 6H1
dtaylor@uwaterloo.ca
[2] IBM Research Division, P.O. Box 704
Yorktown Heights, New York 10598
{halim,hellers,shengma}@us.ibm.com

Abstract. Monitoring large distributed systems often results in massive quantities of data that must be analyzed in order to yield useful information about the system. This paper describes a task-oriented approach to exploratory analysis that scales to very large event sets and an architecture that supports this process. The process and architecture are motivated through an example of exploratory analysis for problem determination using data from a corporate intranet.

1 Introduction

Effectively understanding the behavior of large, complex distributed systems requires appropriate visualization techniques to discover patterns that indicate underlying problems and the causes of those problems. While visualization is effective for modest sized data sets [4], displaying detailed data scales poorly. This paper describes a process for scalable visualizations for problem-determination purposes and proposes architectural principles to support that process.

We begin by illustrating the problem at hand. Consider event data extracted from a modest-sized intranet, say on the order of 50,000 events [2,3]. Discovering patterns such as periodicities (e.g., due to intermittent monitoring) and cold-start patterns can be done using simple displays, such as a scatter plot of all data by host name versus time.

This scales poorly for at least two of reasons. First, as we increase the number of events, there is a higher probability that events will overlay one another in the scatter plot, thereby diminishing the value of the detailed view of data. Second, while it is feasible to maintain an in-memory database of 50,000 events, larger quantities of events—5,000,000 or more per week is quite possible—cannot be efficiently handled in memory, at least not in a naive way. A third issue is the dimensionality of the data. With richer data, we have difficulties with the visualizations themselves since using two-dimensional renderings of high-dimension data makes it difficult to see complex relationships.

There are two possible approaches for dealing with these problems of scale. One is to design a visualization tool that deals directly with very large volumes of data, presumably keeping data on disk and using sophisticated algorithms both

A. Ambler, S.B. Calo, and G. Kar (Eds.): DSOM 2000, LNCS 1960, pp. 47–58, 2000.

to extract needed information from that data and to build optimal displays given the available resolution. The alternative is to observe that it is very unlikely a user will require detailed information from the complete set of data at one time, particularly given the fundamental difficulties of comprehending a complete picture of a very large set of data. Thus, if the user fundamentally needs to extract subsets of data in order to obtain insight, a two-level visualization system provides a good solution to both difficulties. In such a system, small subsets of the data can be extracted as required and processed by the "core" visualization tool, rather than trying to deal with all the data at once.

The concept is similar to "drill down" [1], but does not match it exactly. The usual concept in drilling down is that multiple, hierarchically structured categorical attributes are used in searching for areas in which a numerical attribute has extreme values. The exploration contemplated here includes such situations, but also includes others. In particular, categorical attributes such as server-up/server-down could be the target of exploration and the total number of occurrences of server-down events under a certain set of restrictions might not be the only relevant property of those events.

In this paper, we first present a scenario for analysis of large-scale data that suggests it is feasible to proceed in the manner just described. Then, we generalize from the scenario to determine principles for performing such analyses and for the architecture of a software tool to support them. Finally, we describe our plans to implement this architecture as an extension to the Event Browser.

2 A Scenario

The following scenario is based on actual exploration of data collected from a corporate intranet. Numerous probes have been installed in the network to check the health of important server machines. Each probe periodically checks on a set of server machines, both by "pinging" the server and by attempting a user-level transaction. In most cases, a server is checked by several probes. Probes are frequently close, geographically, to the servers they are checking, but probes are also located at considerable distances when the server is used extensively from distant parts of the network.

Each probe attempt generates an event record indicating whether the server was successfully contacted and, if it was, what the user-level and ping response times were. In total, roughly 5,000,000 such probe records are generated per week, which are stored in a relational database structured into a data warehouse using a star schema. Extracting useful information from such a large collection, beyond simple summaries such as average availability per server, presents significant challenges.

Here, we consider using this data for problem determination: locating servers that are down or inaccessible or that have unacceptably long response times, and beginning to identify the underlying problems.

The scenario presented here is intended to indicate possible approaches to extracting useful information from this event-data warehouse. The techniques

described contain novel aspects related to the use of a two-level storage structure, but are not included primarily as interesting in their own right. Rather, they demonstrate the feasibility of the approach and provide a necessary foundation for the process and architecture discussions in the two following sections.

The first task attempted is to identify servers that are frequently reported down or inaccessible. In this case, we begin by extracting all the records, for the one-week period, that indicate the server was not successfully contacted by the probe. Fortunately, this occurs in about one per cent of the cases and thus provides an event set of manageable size for in-memory analysis.

Obtaining a list of servers ordered by total number of "down" reports allows us to investigate as many of the servers near the top of the list as seems appropriate. Most servers are checked by multiple probes. Thus, we use a subset of the data to plot the probes reporting that a single server is unreachable. The results are displayed in Fig. 1.

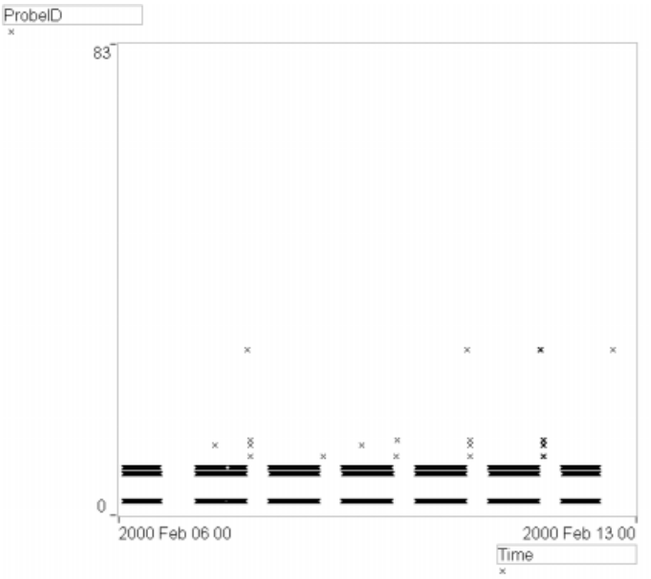

Fig. 1. "Down" Events for an Individual Server: Probe ID versus Time

Three of the probes appear to be reporting the server as down or unreachable nearly continuously and four others have reported it as down or unreachable on a few occasions. (Given the usual probe configuration, it is a reasonable assumption that the regular gaps represent periods that the probes are configured to be inactive.) A small "imperfection" in the second line for one of the three probes seems to indicate that it did succeed in contacting the server briefly. However, it is important to recall that we have simply extracted the "down" events, so

the gap could represent "up" events or no events at all. In this case, extracting all the events describing probe attempts to that one server by that one probe reveals that it simply failed to report on the server at all for about two hours.

In this case, we can conclude with very limited use of event data beyond the "down" events that for most of the intranet there are occasional problems in contacting the server, but likely not frequent enough to be significant. However, either there is a significant network problem in the vicinity of the three probes (they are geographically close to each other) or they are misconfigured so that they are unable to contact the server. In either case, the situation warrants further investigation.

Now consider a second server. As shown in Fig. 2, there is a fairly small number of "down" events, reported by a large number of probes. These events appear to occur regularly at intervals of about one day. We confirm this by changing the x-axis to time-of-day, producing the display shown in Fig. 3. This confirms that the events are indeed clustered in a short daily interval; almost all of the "down" events occur between 8:30 and 9:30 A.M. A possibility is that some activity on either the server or the network occurs regularly at that time and triggers occasional failures. Since the time interval is early in the morning, it may also simply be a large amount of "getting started" activity as employees arrive for the start of the work day.

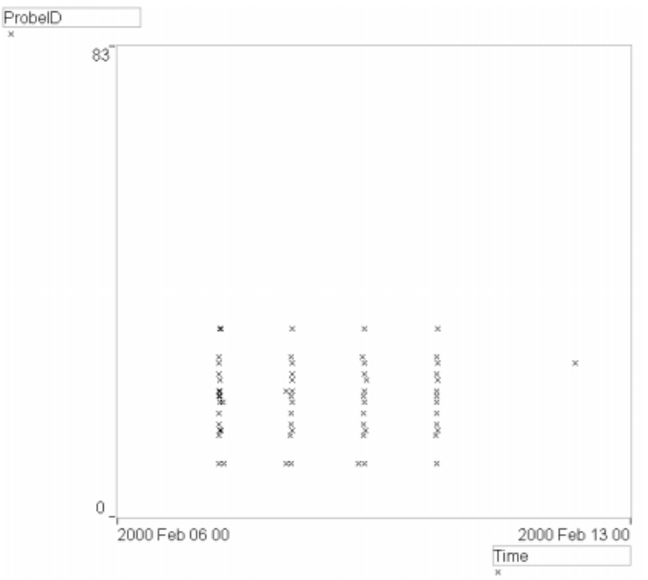

Fig. 2. "Down" Events for Another Server: Probe ID versus Time

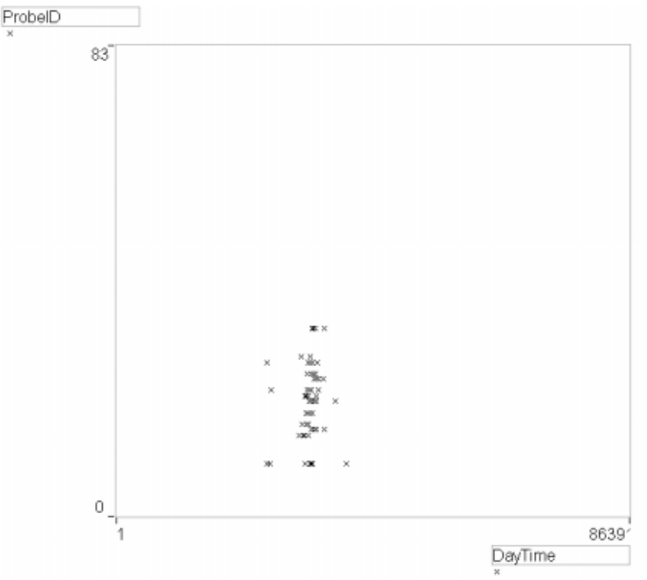

Fig. 3. "Down" Event for the Same Server: Probe ID versus Time-of-Day

A plausible theory for the time-out events is that there is an overload condition occurring regularly each morning. This overload then causes timeouts that produce the "down" reports. Pursuing this theory, it is appropriate to turn to the other aspect of our investigation and examine response times for the server in question. Doing so requires accessing "up" events for the server, so all events were extracted for that server which either reported it down or had above-average response times. Fig. 4 shows a plot of response time against time-of-day, for the "up" events in that set. Note that the x-axis, which in this figure and Fig. 3 gives seconds since midnight, here runs from 25216 (approximately 7:00 A.M.) to 79195 (approximately 10:00 P.M.), because it represents data for only one server and that server is only probed during that interval. In Fig. 3, the x-axis covers essentially the entire 24-hour day because data from all probes and servers is included. We observe that there is indeed a very high response-time peak; unfortunately it occurs from roughly 7:00 to 7:45 rather than coinciding with the "down" events.

The situation is thus rather mysterious, but clearly warrants further investigation. It appears that very early in the work day there is a burst of activity that pushes some response times into the several-minute range and that slightly later in the work day an unknown phenomenon causes occasional apparent server failures. (The reported failures are so brief that it appears unlikely a server could actually crash and restart, but the server does occasionally fail to respond to a probe.) To the extent that these represent an expected burst of activity at the start of the work day, they may simply need to be tolerated, but if any scheduled,

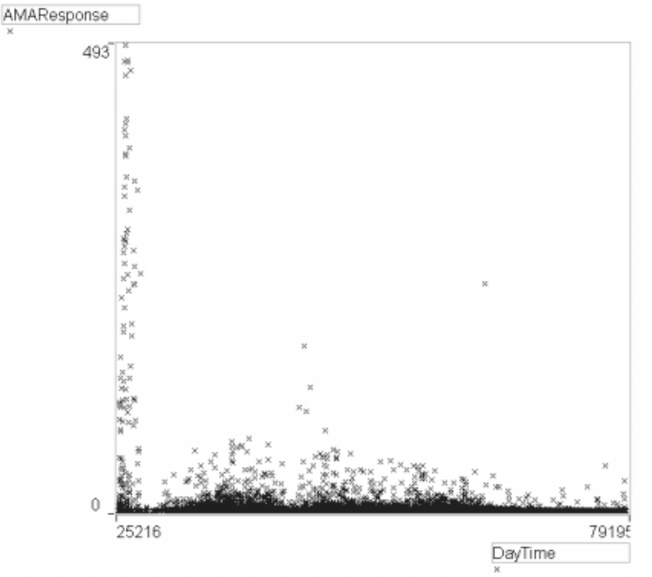

Fig. 4. Response Time versus time of Day, for the Same Server

non-interactive work is taking place during these periods, rescheduling it might alleviate the observed problems.

Finally, we can examine response times across the complete set of servers, rather than for this one server. For this purpose, events corresponding to response times three or more standard deviations above the mean were extracted, for all servers and probes. Fig. 5 shows response time versus (local) time-of-day for all probes and servers. In this case, the mean and standard deviation were computed for each probe/server pair, and response times were determined to be high relative to the statistics for that probe and server. Since some probes are located at the same site as the server and some are on other continents, averaging across probes, even for a single server, is not very useful in determining whether a response time is unreasonably high.

Some of the response times are quite extreme (almost 25 minutes), but the two most prominent features of the graph are a sparsely filled rectangular area and a prominent spike at the left side of the rectangle. The rectangle is easily explained as corresponding to the business day. It is hardly surprising that long response times tend to occur when most people are at work. The spike appears to be quite similar to that observed in Fig. 3. Here, the events in the spike have been colored so that they can be examined in other views, in particular to determine what servers they correspond to. In this case, the rectangular region selected as representing most of the spike (minus its base), contained 207 events, spread across 31 servers. However, somewhat closer examination reveals that 24

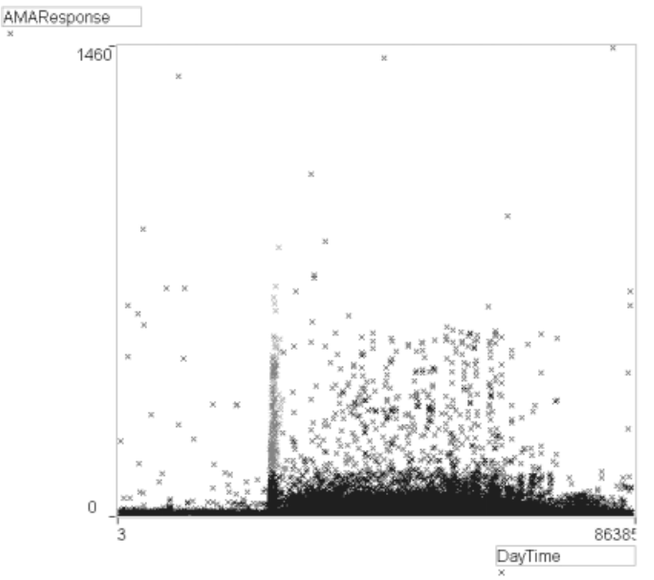

Fig. 5. High Response Times versus Time-of-Day, All Servers

of these servers, representing 196 of the events, are in the same country, but involve all three of the data centers in that country.

Thus, it appears that either the network or a substantial collection of servers in that one country is being subjected to some form of overload. A plot by time rather than time of day (not included here) indicates that the response-time peak is a five-day-a-week phenomenon. Thus, the problem with response times first observed for a particular server that generates an odd pattern of "down" reports turns out to be common to numerous servers, but only within a particular country.

This section has briefly described some of the approaches used in extracting useful information from the event-data warehouse for the corporate network. All of the work described was performed by formulating a SQL query, using it to extract events from the database, reformatting the data to make them suitable for use with the Event Browser, and then using the Event Browser on the data. Clearly, a more tightly integrated approach is highly desirable. The next two sections derive general principles from this scenario, first for the process to be followed in analyzing such large event sets and then for the architecture of a tool to support that analysis.

3 Process Principles

The scenario described in the preceding section suggests a general process that can be followed in dealing with large volumes of event data. The process can be briefly outlined as follows:

1. Apply any basic restriction (such as a date range) and obtain summary statistics.
2. Using the summary statistics together with domain information and the desired goal for exploration, formulate an initial set of restrictions and extract the corresponding subset of events. If all obvious initial restrictions yield sets that are too large, a further possibility is to extract a statistical sample of the specified events, for initial analysis.
3. Repeat steps 4 through 6 until analysis is complete.
4. Explore the current subset of events, using any appropriate interactive facilities available. Look for unusual (extreme) values, look for patterns in appropriate visualizations of the data, and so on.
5. If phenomena of interest are discovered but cannot be fully investigated in this event subset, determine (a) what common properties are possessed by the events of interest and (b) what restrictions used in extracting this subset need to be relaxed in order to obtain further information relevant to the phenomenon discovered.
6. Based on the analysis of the preceding step, formulate a new set of restrictions, extract the corresponding subset of events, and (a) replace the current set of events with the new set, (b) form the union of the current and new sets, or (c) keep both new and old sets for concurrent but independent examination.

Most of the above follows quite directly from the scenario described in the preceding section except for the first step and the three-way choice in the last step. In the scenario, the first step was effectively replaced by an understanding that "down" events and events with high response times are quite rare.

For the last step, choice (a) was used throughout, that is, a new set of events was identified and used in place of the previous set of events. A difficulty with that strategy is that a user will be trying to accumulate information about a problem until the problem is fully understood or all available data are exhausted. Simply discarding one event subset and replacing it by another can cause a loss of context and accumulated information.

The somewhat complicated specification for the events used to create Fig. 4 is essentially intended to maintain context. By extracting the "down" events previously examined as well as the events with long response times, the new events can be seen in the context of the old events. A user would not need to write such specifications, which could become progressively more intricate as analysis progressed, if it was possible to add new events to the existing set or to display two (or more) event sets simultaneously. The ability to display two sets simultaneously also facilitates taking a quick exploratory look at a new subset and then returning to the set currently being examined.

4 Architectural Principles

An effective tool for examining very large sets of event data must provide good support for the process outlined in the preceding section. The remainder of this section lists a number of principles that appear necessary or at least very desirable in designing a tool that will support the kind of analysis discussed.

Principle 1: Provide a general, easy-to-use mechanism for specifying event subsets, allowing all event attributes to be used in the specification.

Repeatedly selecting event subsets is a central feature of the process discussed in the preceding section. The scenario indicates that there can be a great deal of variation in the criteria used to specify a subset of events. At minimum, all the attributes of an event need to be useable in the selection criteria. For categorical attributes it may be sufficient to allow selection of the individual categorical values to be included, but see Principle 3 for categorical attributes with a very large number of values. For numeric attributes, in addition to ranges with constant limits, range end-points that are simple functions of properties like minimum, maximum, mean, and standard deviation are also needed. It is also necessary to provide a mechanism for specifying over what set the statistics are to be computed. In the scenario of Sect. 2, a specification would indicate that means and standard deviations were to be computed for each combination of probe, server, and up/down status (the last in order to exclude fictitious response times reported when no response was received). The mechanism for specifying new subsets should also be appropriately integrated with other mechanisms for examining event data, so that selections made in examining the current event subset can be easily used in selecting the next subset.

It is somewhat less clear what facilities need to be provided in combining restrictions on different attributes. The subsets used in the scenario were almost all specified as the intersection of restrictions on the relevant attributes. Only one involved a union: all the events for a particular week and a particular server that either reported the server down or had a high response time. The scenario does not provide a strong case for using a union since the "down" events were filtered out of the graph in Fig. 4, but a graph could have been presented in which the "down" events were marked with color and their position could then be directly compared with the position of the response-time peaks. If a general facility is provided for adding newly extracted events to the current set (as required by the principles of the preceding section), then a union operation need not be provided directly. This will make the use of union somewhat less convenient than using specifications based purely on intersection, but that may be a reasonable tradeoff given the likely frequency of use.

In most cases, the complete collection of event data will be in a database with a SQL interface, so it is also possible to allow a user to write a SQL query directly and use that for data extraction. While it would be undesirable to require ordinary users of an event-browsing facility to write SQL, such an escape mechanism could be useful to avoid overburdening the graphical user interface with seldom-needed capabilities.

Principle 2: Event-subset specifications should allow direct specification of the size of the result subset.

One issue not explicitly discussed in Sect. 2 was the varying definition of "high response time." In one case it is simply response times greater than the mean and in another response times three standard deviations above the mean. Not surprisingly, the underlying reason in each case is to provide a restriction that appropriately limits the size of the specified subset. In one case, only a single server is involved so a fairly loose interpretation of "high" yields a subset of reasonable size, but when extracting events without restriction by probe or server, a very stringent definition of "high" is required in order to limit the subset size. Some trial and error went into the determination of these specifications, which it would be desirable to eliminate. A facility that allowed a specification like "the events with the highest response times, which also satisfy this set of selection criteria" could be used to obtain a set of the appropriate size for interactive browsing. Unfortunately, this raises yet another difficulty: what does "highest" mean? It could be highest in actual value, highest as a multiple of the mean, or highest in number of standard deviations above the mean. Again, it is necessary to give the user considerable flexibility in specifying constraints because no one form will be suitable for all purposes.

Principle 3: The hierarchical structure of categorical attributes needs to be made readily accessible to the user.

Another issue is implied by observations in Sect. 2 like "they are geographically close to each other" and "24 of these servers ... are in the same country." In this case, stereotyped naming of most servers and all probes allows a knowledgeable user to deduce geographic information, with some difficulty, directly from the names. It is undesirable to ask users to assimilate such a scheme and in other cases, names will not be stereotyped, requiring access to on-line or off-line reference material to determine relationships. In this case, the event-data warehouse includes a six-level geographic hierarchy; similar hierarchical information is likely to be available whenever a categorical attribute has a very large number of values. Making such hierarchy information directly accessible when using an event browser would simplify the selection of events using categorical attributes with many values, as well as being helpful in determining the significance of patterns observed.

A facility for accessing such hierarchical information needs to be quite flexible. In particular, multiple hierarchies can reasonably exist for a single attribute. For example, in the case of the servers in the corporate intranet, in addition to the geographic hierarchy there are separate, shallower hierarchies corresponding to the function of the server (database, mail, web, etc.), hardware architecture and speed, and operating-system type and version.

Principle 4: Provide summary data, using an interface similar to that used for displaying detail data.

In many cases, some information about the overall set of events will be needed before it is possible to formulate a reasonable specification for the initial set of

events. For example, in the analysis described in Sect. 2, a SQL query could have been used to determine the number of "up" and "down" events for each server. Then, rather than extracting all "down" events and working with them, the events for servers frequently reported down could have been examined for each server in turn. If the set of "down" events had been substantially larger for any reason, including the desire to examine a period longer than a week, such summary data would have been vital.

As much as possible, the usual interface for examining detailed data should be used with summary data, rather than providing a separate interface. Although only a subset of the usual functionality can be provided for such summary data, it will clearly be helpful to the user if it is not necessary to learn the use of a separate interface. Such summaries should be available for event subsets as well as for the entire set of stored events. For example, if all subsequent analysis is for events in a particular week, then summary information needs to be restricted to that week as well.

Principle 5: Multiple event subsets should be simultaneously accessible to the user.

The discussion in Sec. 3 indicates that simultaneous access to multiple event subsets can be very helpful. It might appear that providing such a facility would require that the total size of the event subsets be no larger than a single subset would be, to obtain good performance. Fortunately, this may not be true. In most cases, the central performance issue is not likely to be the total virtual-memory footprint but the use of virtual memory when scanning through an event set to perform operations like coloring all events (potentially in several displayed views) that have a specified property.

Principle 6: User actions in selecting event subsets should be recorded and made accessible.

As a user performs a lengthy sequence of selecting and examining event subsets, the current state of the present and previously examined event subsets may no longer be clear to the user. This is particularly likely if multiple subsets are retained for concurrent examination or subsets are modified by adding additional events after they are first created. To allow a user to understand the state behind the current displayed views and also to allow a user to retrace previous steps if necessary, a "history" facility is highly desirable.

5 Conclusions and Further Work

In this paper, we have determined both an effective process for working with large sets of event data and a set of architectural principles for building a tool that properly supports that process. With the exception of concurrently examining multiple event sets, the process described can be used with the current Event Browser, but the process is quite inconvenient. The chief problem is that extracting an event subset requires writing a SQL query, executing it, running

a program to reformat the output of the query, then opening the resulting file in the Event Browser.

Our plan is to extend the Browser to allow the suggested process to be performed interactively during a single Browser session. Most of the work required is straightforward, for example, the SQL queries required can be readily generated from standard patterns, with user input essentially used to "fill the blanks" in those patterns. The major challenge is likely in devising a user interface that provides enough flexibility to implement all the principles described in the preceding section without overwhelming the user. Additional facilities not covered by the principles of Sect. 4 might also be added, for example a recording facility that could be used to document the actions taken or to create a "macro" that could be used by a less experienced user in later performing a similar analysis.

At least two additional challenging problems remain to be addressed. One is that this paper has only considered scaling issues with respect to the number of events available. Another potential problem is scaling in the number of attributes per event. An implicit assumption in the above is that the number of attributes per event is small enough that the user can individually inspect any that appear of potential interest without special support from a software tool. We have recently been given access to a new set of data in which the number of events is not particularly large but there are more than 250 attributes per event. Different techniques are required to help the user extract useful information from such a large set of attributes.

Another problem is the development and integration of techniques for mining the event data for patterns that are not immediately apparent in standard views such as scatter plots and bar charts. Some mining techniques have already been developed for use with the set of data currently being examined by the Event Browser, but further techniques are required and they also need to be made workable in a two-level environment as proposed here. That is, it should be possible to execute a mining algorithm against an entire event-data warehouse or a large subset of it, probably as an off-line activity because of the time required, and then interactively use the results of the mining together with the other "basic" information available for the events.

References

1. Robert F. Berry, Joseph L. Hellerstein: A flexible and scalable approach to navigating measurement data in performance management applications. Proceedings, Second International Conference on Systems Management, Toronto, Canada (June 19–21, 1996).
2. M. Derthick, J. A. Kolojejchick, S. F. Roth: An interactive visualization environment for data exploration. Proceedings of Knowledge Discovery in Databases (1997).
3. Sheng Ma, Joseph L. Hellerstein: EventBrowser: A flexible tool for scalable analysis of event data. Proceedings, Distributed Systems, Operations and Management, Zurich, Switzerland (October 11–13, 1999).
4. Edward R. Tufte: Envisioning Information. Graphics Press (1990).