Interactive Querying of Temporal Data Using A Comic Strip Metaphor *

Jing Jin Pedro Szekely†

Information Sciences Institute
University of Southern California

ABSTRACT

Finding patterns in temporal data is an important data analysis task in many domains. Static visualizations can help users easily see certain instances of patterns, but are not specially designed to support systematic analysis tasks, such as finding all instances of a pattern automatically. VizPattern is an interactive visual query environment that uses a comic strip metaphor to enable users to easily and quickly define and locate complex temporal patterns. Evaluations provide evidence that VizPattern is applicable in many domains, and that it enables a wide variety of users to answer questions about temporal data faster and with fewer errors than existing state-of-the-art visual analysis systems.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interface—User-centered design; H.3.3 [Information Systems]: Information Search and Retrieval—Query formulation

1 INTRODUCTION

In the modern information age, the quantity and complexity of time-oriented data is rapidly and continuously increasing. The ability to analyze temporal data is fundamental to reasoning and making critical decisions in many domains. We focus on data sets containing temporal data about individuals or objects. The data can be an event (e.g., Patient 1 had a heart attack on day 10), or can be part of a time series (e.g., Patient 1's LDL cholesterol on day 10 was 250).

Our overall objective is to enable users to gain insights held within diverse temporal data. Our progress towards this objective is advanced by achieving the following three goals. Our main goal (G1) is to make it easy for users to answer questions regarding temporal relationships about events or time series data. For example, "how many patients had a downward trend in their cholesterol level within 10 days of the second dose of medicine X?"

Typically, answers to questions lead to additional, related questions. Our second goal (G2) is to make it easy for users to incrementally perturb a question and immediately see how the answer changes. This immediacy enables users to understand the sensitivity of the answer to various elements of the question, and is inspired by Shneiderman's Dynamic Query techniques [17].

Our third goal (G3) is to leverage existing visualizations of temporal data. Many visualizations of temporal data have been created [1], and they are often the starting point for deeper data analysis. However, they typically don't offer the ability to perform systematic analyses because they don't offer the ability to ask complex questions about the data. We want to enable users to ask their temporal questions by using the icons and graphics they see in the visualization. This is in contrast to requiring users to learn and understand the symbols and relation names used in the underlying data, which users don't actually see in the visualizations.

In previous work [6], we showed that this metaphor is easy to learn and understand (G1). Users were more effectively able to translate queries into a comic-based representation and extract the accurate meaning of a query in a comic-based representation when compared with a form-based representation. In this paper, we present *VizPattern*, an interactive visual analytics environment designed to achieve the three goals identified earlier. In this work, we performed a new user study that shows that users can effectively use this metaphor to answer questions about a data set (G1, G2) and we tied our comic strip metaphor to existing visualizations (G3). Four key capabilities of *VizPattern* that enable our effectiveness are:

C1: The ability to manipulate results to refine queries We developed techniques to introduce and edit temporal relationships in the comic strip query by editing instances of query results. Users can edit undesired results to form queries that return desired results. The system infers new relationships and asks users for clarification to resolve ambiguities (G2).

C2: Example-based construction and modification of queries from visualizations. Users can drag-and-drop icons, such as events, directly from the visualization into the comic strip to construct or modify it. A key innovation is a technique to incorporate pieces of a time series into comic strips to represent constraints that the time series must satisfy (G3). The system generalizes a user-selected segment of a time series into a template that is fuzzily matched against all relevant time series. Users control the strictness of the match for each template.

C3: A Dynamic Query interface to explore variations of questions. Temporal queries have several numeric parameters (e.g., time interval between events, strictness of fuzzy matching for time series templates). We enable the Dynamic Query technique [18] so that users can drag any of these parameters and immediately see the matches to the revised query (G2). Thus, users can better understand how answers depend on the parameters of the query.

C4: Integration of event, time series and geospatial data. A significant drawback of existing temporal query systems is that they focus either on event data or time series data, but are not able to combine in the same query both types of data. Our medical example illustrates why combining both types of data is interesting and useful. Our comic strip metaphor can easily represent both types of data, as both of them can be represented as elements of comic strip panels. Events are represented by icons, and time series are represented using a segment of the time series. We developed an algorithm that converts time series constraints into events so that temporal queries combining both types of data can be answered using a generalization of SeqJoin algorithm [9]. Using the same technique, our user interface and algorithms also support a simple, yet powerful geospatial constraint where an object can be constrained to be within a given distance from a point (G1).

*The work presented here is funded by the DARPA COORDINATORS Program under contract FA8750-05-C-0032. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them. Approved for Public Release, Distribution Unlimited.

†e-mail: {jing,pszekely}@isi.edu

2 Related Work

Comic Strip Metaphor: Our use of the comic strip metaphor for temporal queries was inspired by Kurlander's work on Chimera [8], where he uses comic strips to depict commands histories. Since then, several systems have been created that use comic strips to convey events over time, taking advantage of users' familiarity with comic strip conventions. ComicKit [7] allows children to use comic strips to make event-based programs. GIS [20] uses comic strips to demonstrate query procedures to non-specialist users. *VizPattern* is the first system to use this metaphor to help users construct temporal queries.

Visual Analytics Systems: Many visual analytics systems have been created to help users explore, query and analyze data. We sort these systems into those that use GUI widgets, and those that use direct manipulation techniques.

The systems in the first group use traditional GUI widgets to specify temporal patterns/queries. Dynamic Queries [18] is an early visual query system for database. Its instantaneous query modification/update cycle gives users insights into their data traditional modify/submit/view-results systems do not provide. Dynamic Queries was not designed to create temporal queries, but our work shows how its features can be incorporated into a temporal query system yielding similar benefits.

TVQL [16] is a visual query language for identifying relationships between events of interest in video data. TVQL uses four double-sided sliders to allow users to express the relationship between interval endpoints, supporting Allen's 13 relational primitives [2]. The language is expressive, but users studies show that the double-sided sliders are hard to use [16]. PatternFinder [3] [13] is also supports an expressive language to find patterns of events across multiple records. It uses forms to specify queries, and a ball-and-chain visualization to display matching results. As our previous user study shows [6] it is often hard for users to represent queries using forms. Our use of ball-and-chain visualizations for results is inspired by this work. We enhance it by allowing users to manipulate the visualization to refine the query. Lifelines2 [21] inspired much of our work. Its sentinel event concept is complementary to the capabilities in *VizPattern* as it provides additional tools that would be useful to further analyze the query results.

The systems in the second group use visualizations as the starting point for their analysis. They allow users to directly interact with the visual display of the system to specify patterns. In QuerySketch [22], users directly draw a "line shape" which represents a pattern, and the system uses the Euclidean distance as the similarity measure to match the pattern against the data. Our algorithm to match time series segments is similar to the one used here. [5] introduced two relaxed selection techniques for querying time-series graphs. Users sketch over the graph in one stroke, and implicitly define the similarity by deviating from the original graph, or by the speed of sketching. TimeSearchers [4] let users specify patterns of interest on line graphs by creating "time boxes", rectangular query operators that specify the regions of interest to users. Each time box identifies a value range and a time interval. The conjunctive queries are incrementally defined through several time boxes. [23] presented a visual analysis system for network traffic data. In this system, users first select example event sequence from an event diagram, and then iteratively choose one or more predicates returned from the system based on the initial selection. Once such a logical clause is created, it is incorporated into the knowledge base which will be applied to all data. Users can then use the augmented data to continue such analysis cycle.

The following shortcomings exist in the above systems. First, except for PatternFinder, they don't have a separate result output/visualization. Our work shows that the separate results visualization is useful to help users further understand the matching results and to modify the query. Secondly, most of them are example-based. Our work attempts to integrate the example-based methods with direct specification methods to give users more control over their queries. Finally, most of them provide support for single temporal data type (temporal events, or time series data), but don't address the data sets that contain both.

3 Example

In the following sections, we present our work in the context of a synthetic scenario where we compare two medicines for reducing cholesterol. Our data set has information for 8 weeks and 100 subjects, 50 who received blue pills, and 50 who received green pills. It contains events to record the days when patients took pills and the days when they suffered side-effects (nausea, stomach pain and chest pain). It also contains a time series with daily cholesterol readings for each patient. We would like to understand the effectiveness and side-effects of the pill types.

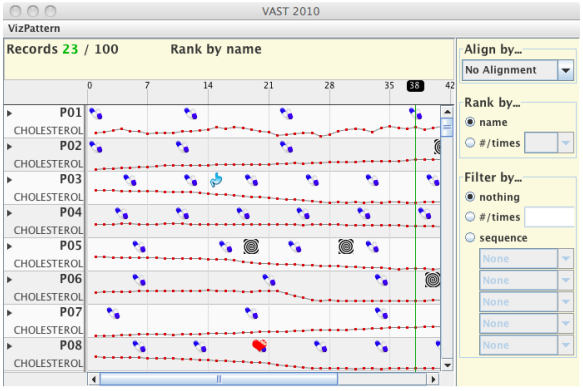

Figure 1: Visualization of patient data.

Figure 1 shows our patient data set using a visualization similar to Lifelines2 [14, 21]. Rows represent patients and the horizontal axis represents time. Events are shown using icons (pills, hearts, stomachs and nausea), and cholesterol readings are shown using a line chart. This visualization is illustrative of the type of visualizations that can be tied to *VizPattern*.

4 VizPattern

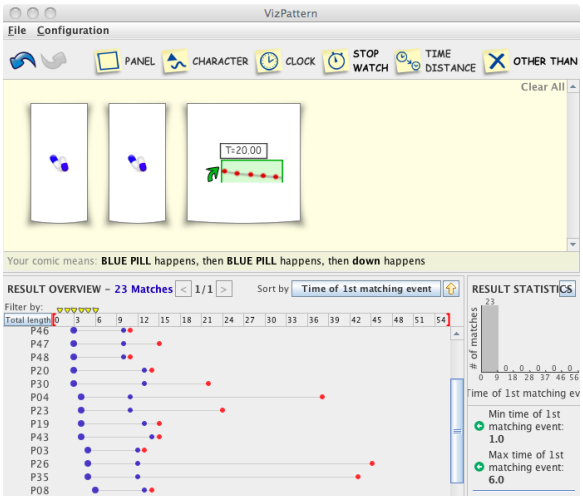

Figure 2: The VizPattern workspace.

Users doing data analysis with *VizPattern* work using two types of views. The first, shown in Figure 2, is the *VizPattern* workspace that consists of two panels, a Comic Strip Editor where users compose their queries and a Results Panel where they view the answers.

The second type of view are visualizations where users see the details of their data. Of special interest are visualizations that implement the *VizPattern* API. Users can drag elements from these visualizations into the Comic Strip Editor in order to specify their queries, and when they select entries in the Results Panel, all visualizations that can show the corresponding entry will be highlighted.

VizPattern supports a rapid, iterative data analysis cycle where users typically compose queries by dragging elements from their data visualizations into the Comic Strip Editor. The Results Panel immediately updates to show the answers, which often gives users insights that leads them to refine their queries. They can do so directly, by editing the comic strip or dragging additional elements from the visualizations, or indirectly, by manipulating the answers in the Results Panel.

In the rest of this section we briefly summarize the comic strip metaphor for composing temporal queries, we describe the Results Panel, and then use our patient data example to illustrate the *VizPattern* data analysis cycle.

4.1 Comic Strip Metaphor

Our visual language to compose temporal queries is called *QueryMarvel* [6]. It uses a comic strip metaphor to represent temporal queries or patterns, where characters or actors in the comic strip are used to represent events, and the visual language of comic strips is used to represent the temporal relationships among the events. The main advantage of this approach is that it can convey the abstract representation of a temporal query using the visual, concrete familiar elements of a comic strip. The details of *QueryMarvel* and its usability benefits are documented in our prior work [6]. In this section we briefly summarize the *QueryMarvel* constructs by describing how elements of temporal patterns are mapped into a comic strip in *QueryMarvel*. In each mapping, we show a picture of a real life comic example that conveys the same temporal relationship, and the corresponding visual elements in *QueryMarvel*.

Temporal Events: A comic strip consists of a series of panels laid out along an invisible time line. To tell a story, characters inside the panels are shown doing something at some time. Similarly, in *QueryMarvel*, each visual snippet becomes a character. For example, putting a "chest pain" icon into a comic panel means "chest pain" happens at some time.

Event B happens after event A: As a form of sequential arts, comic strips by design show events happening in a sequence. When panel B follows panel A, the events in panel B happen after the events in A. In Figure 3(a), it is clear that the woman reaches inside the crib after having stood next to the crib looking inside. Similarly, the two *QueryMarvel* panels show stomach pain after chest pain.

Event A and B happen at the same time: When a comic strip panel contains multiple characters, we assume they are simultaneously doing something. In *QueryMarvel* we represent multiple events happening at the same time by including them in the same panel. Figure 3(b) shows two players attacking and defending at the same time. Similarly, the *QueryMarvel* panel shows chest pain and stomach pain happening at the same time.

Event A happens after B with specified time interval: Comic strips use a text label at the top of a panel to indicate that the events depicted in it occurs a specific amount of time after the events in the previous panel. *QueryMarvel* uses the same technique to represent time interval constraints. For example, Figure 3(c) shows that the second conversation occurred one minute after the first conversation. Similarly, the *QueryMarvel* panel shows stomach pain happening 10 days after chest pain. The text label can be used to represent a variety of temporal relationships.

Event happens at a specific time: Comic strip panels include a clock to convey that the events inside the panel occur at an absolute time. *QueryMarvel* uses the same technique to represent absolute time constraints. For example, Figure 3(d) shows the man checking the office supplies at 9:30. Similarly, the *QueryMarvel* panel shows chest pain happening on the 10^{th} day of the medial trial.

Event happens in an absolute time range: This is easily represented in *QueryMarvel* using the building blocks defined above. Figure 3(e) shows an example of a chest pain event happening between the 10^{th} and 100^{th} day after the start of the medical trial: two panels with clocks showing 10 and 100 are put before and after the chest pain panel. It is possible to bracket an arbitrary number of panels between the clocks.

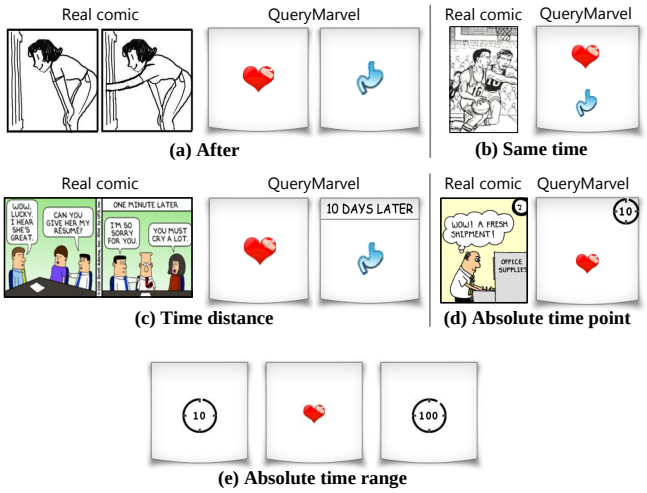

Figure 3: Temporal Constraint Mappings

The mappings described above show how several temporal pattern elements can be mapped directly to the comic strip metaphor. Next we discuss extensions that support more expressive temporal patterns.

Other than: It is often desirable to represent patterns such as "something other than stomach pain happens after taking X". *QueryMarvel* represents it by putting a red cross over an event snippet. Figure 4(a) shows "something other than stomach pain".

Or: *QueryMarvel* supports "Or" constraints using the grid panel. The cells in the grid represent the branches of Or. Figure 4(b) shows chest pain Or stomach pain happen at some time.

Object constraints: The default behavior in *QueryMarvel* is to match all events on the same object. It is often useful to represent temporal constraints that relate events on different objects (owners). For example, "server 1 is down within 10 minutes after server 2 is down". In *QueryMarvel*, we introduce "tags" to represent object constraints. If we want different objects to match events, we attach different color tags to these events icons, and if we want the same object to match these events, we give the tags the same color. If we want a specific object to match an event, we put the identifier of the object on the tag. For example, Figure 4(c) shows a constraint where one server goes down and then a different server goes down. Figure 4(d) shows the case where specific servers go down one after the other. If no tag is attached to event icons, then it means they match to the same object.

Time interval between events from non-consecutive panels: To specify time constraints between non-consecutive panels we use a "stop watch" metaphor. To specify the time interval between panel P1 and P2, we first put a stop watch in P1 to represent the start of a stop watch. Then, we put another stop watch with the same color in P2 with number "X" in it to represent that the watch is stopped here and the time span from the start is X. We can also put ">=x" to represent "over x days", "<=x" to represent "with in x days", or "x-y" to represent "in x to y days." Stop watches are paired with colors. Therefore, users can put multiple pairs of stop watches into the same *QueryMarvel* comic strip. Figure 4(e) shows

the time interval between stomach pain and nausea is 10.

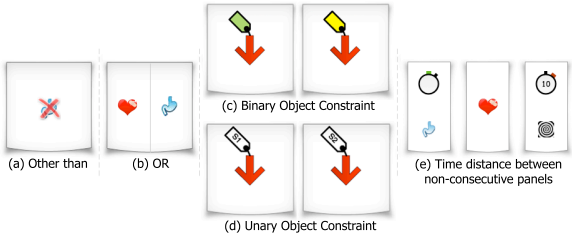

Figure 4: Extensions

4.2 Results Panel

The Results Panel is designed to help users understand the temporal relationships among the data elements that match a temporal query. It uses the well-known information-seeking mantra: overview first, zoom and filter, then details-on-demand [19]. The Results Panel in Figure 5 shows the results for a pattern that identifies patients whose cholesterol went down following two dosages of a blue pill.

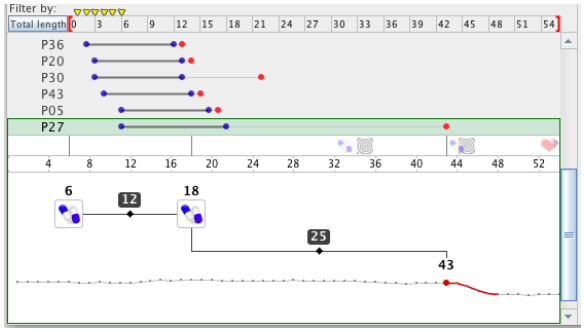

Figure 5: Visualization of pattern matching results.

The results panel shows an overview of the matches using a dot-and-line style [3], where each dot represents a matching event, and each line represents a temporal gap between events. It shows the complete set of matches so that users can quickly gain an overview of the matches at a glance, and see outliers or any patterns that develop within the matches.

Users can filter the results using a time range controller, and in order to better understand relationships among matches, they can sort the results based on particular events, the length of the gap following an event, or the total span of a match. The result statistics panel shown on the right of Figure 2 displays a histogram of the matches according to the criterion used for sorting.

Finally, users can obtain details on demand by clicking on a match to expand the dot-and-line display, revealing the specific events that were matched and the temporal intervals between these events. In addition, the original visualizations from which the icons were dragged are automatically scrolled and highlighted to show the corresponding events.

4.3 Data Analysis

Building Temporal Queries: *VizPattern* is designed to support two types of data analysis use cases. In one type, users approach the system with a question in mind. To support this use case, *VizPattern* enables users to build temporal queries directly in the Comic Strip Editor using the toolbars and menus provided. All interactions are animated to give users a good understanding of the effects of all operations: objects slide to make room for new ones, objects expand or contract smoothly to accommodate the information inside them, puffs of smoke appear when objects are deleted. In addition, all operations offer proactive highlighting to help users understand the effects of the operations that are currently being performed. For example, landing locations for dragged objects are highlighted to guide users to them.

The second use case is one where users are exploring their data in one or more visualizations, and something that they see suggests an analysis to perform. *VizPattern* enables users to start data analysis cycles directly from these visualizations by dragging pieces of these visualizations into the Comic Strip Editor. Users can drag both icons representing events and segments of time series charts. For example, Figure2 shows the state of the editor after we dragged two blue pill icons from our patient data visualization into the Comic Strip Editor, and then dragged a piece of the cholesterol line chart. When dragging time series segments, the system asks whether the Y-values are relevant for matching, or whether users are only interested in the shape of the curve.

By completing these three steps, we started our analysis, focusing on patients whose cholesterol fell following two dosages of blue pills.

Refining Queries Using Examples: After users modify the comic strip to edit their queries, the Results Panel immediately updates to show the answers. Often, the list includes matches that users want to exclude from their current analysis. For example, in our scenario we want to focus on conscientious patients who took their pills on schedule, once per week, so we want to exclude patients where the time interval between the two pill events is over seven days.

VizPattern allows users to refine patterns by manipulating the results. Users can introduce or refine time constraints by dragging the lines sideways in the expanded point-and-line display. While they drag the line, *VizPattern* shows the precise time interval between its end-points. When users release the mouse, the system shows a menu with alternative interpretations of the time constraint (exactly, at least, within). Users can also click on the events in the expanded point-and-line display to change the event type.

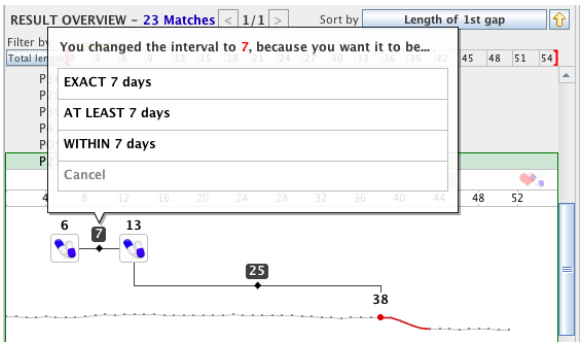

Figure 6: Editing an answer to refine the pattern specification.

Continuing with our example, the Results Panel in Figure 5 shows that patient P27 took the second pill 12 days after taking the first pill. To exclude from our analysis all patients that took more than 7 days between the first and second pill, we simply drag the number 12 towards the left until the interval is what we want. Figure 6 shows that when we release the mouse, the system asks us what we mean, giving us a menu of the possible interpretations.

This paradigm is used consistently for all situations where users can edit examples: the system infers possible interpretations from a single example and asks the user to choose. If none of the interpretations is what the user intends, the user can apply the Comic Strip Editor capabilities to specify more complex constraints.

VizPattern incorporates Dynamic Queries by making all numbers in the Comic Strip Editor draggable. When the user drags them, the number changes and the Results Panel immediately updates to show the answers to the modified query.

In summary, *VizPattern* is designed to support a powerful, intuitive and fast data analysis cycle where users focus on their data,

be it in the original visualizations that show the data in context, or in the Results Panel that shows the relationships between events. Users can define and refine queries by manipulating the data, but when it is more convenient, they can edit the query directly in the comic strip. The unlimited undo/redo feature enables users to quickly take short excursions to investigate variations and side questions. Users can also save queries to files and load them later to reuse useful queries from history, or to compare different queries.

Continuing with our blue/green pill example, we see in the visualization that many of the matching patients have nausea icons. We can drag one of these icons to the Comic Strip Editor to refine the query to show only those patients who had nausea. To compare responses to the green pill, we can simply click on the blue pill icons in the Comic Strip Editor and replace them by green pills. The Results Panel immediately updates, show us the patients whose cholesterol lowered after they took two green pills, and had nausea side effects. We compare the effectiveness of the pills by reading the number of matches.

4.4 Expressivity

The expressivity of *QueryMarvel* can be understood using a graph, as illustrated in Figure 7. **Nodes** represent a disjunction of conjunctive event types. They can be labeled with object names to represent unary object constraints. **Solid edges** represent temporal constraints. Each such edge is labeled by the time interval between two ends of the edge. **Dotted edges** represent binary object constraints. Each such edge is labeled by either "=" or "$\neq$" to represent that the objects at two ends of the edge are equal or not equal.

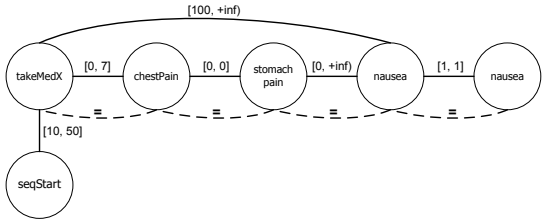

Figure 7: Graph representation of example pattern.

QueryMarvel can represent temporal patterns whose graphs have at least one solid path consisting of solid edges that goes through all the nodes. Every pair of nodes can be additionally connected by one solid edge and one dotted edge. The solid line can be labeled with arbitrary time intervals.

The "solid path" requirement is limiting as it prevents tree structures or arbitrary graphs. Consequently, it is not possible to represent branching timelines, and events that occur simultaneously with multiple other events. It is not possible either to represent that an event should not happen. The "other than" construct is represented as a disjunction of all other event types, and thus represents the constraint that something else happens.

Our pattern matcher uses a variant of the SeqJoin algorithm [9]. We studied the scalability of our algorithm using synthetic data sets where we varied the number of objects, event types, events and average gap between events. Our experiments show that the number of events is the main limiting factor. When processing a data set with 2 million events on a Intel Core-Duo 1.66GHz PC with 2 GB memory, the algorithm takes 30 seconds to complete the one-time index building process, and then can produce query results in about 1 second. However, the current algorithm builds its index in memory, and will run out of memory on data sets with 5 million events.

5 Evaluation: User Study

In our previous work [6], we presented a detailed user study about *QueryMarvel*, the foundation of *VizPattern*. In this section, we evaluate *VizPattern* as a whole. The objective of our user study was to measure how well users who are completely unfamiliar with *VizPattern* are able to use it to do data analysis. To do this, we conducted a comparison study. The motivation was to gain better insights about the pros and cons of our system through the comparison, and hopefully learn how to utilize the results in our future research. The other system we picked was Lifelines2 [21] because 1) like *VizPattern*, Lifelines2 focuses on answering questions about temporal events, 2) its user studies show that using its "Align, Rank and Filter" (ARF) framework improved the ability to answer questions by 60% compared to visual inspection, which represents state-of-the-art research in this field. We were unable to obtain the Lifelines2 source code for use in our evaluation, so we reimplemented it according to the description provided in the literature [21]. We also considered comparing *VizPattern* with other well-known systems, such as TimeSearcher. However, TimeSearcher is mainly used for value analysis in time series data sets, whereas *VizPattern* focuses on temporal events.

Figure 1 shows a screenshot of our own implementation of Lifelines2. The panel on the left shows the events, and the panel on the right shows the controls for the ARF framework. Using the "align by" tool, users can choose any occurrence of any available event type and force all the records to be aligned by the chosen event, and the time scale becomes relative to the aligned event; using the "rank by" tool, users can order rows by the number of occurrences of a selected event type; using the "filter by" tool, users can filter the records to show only those with a given number of occurrences of an event or those where the events appear in a specified sequence. Users can combine and use these tools together.

Our implementation of Lifelines2 supports the *VizPattern* API. Users can drag events and time series segments from it and deposit them in the Comic Strip Editor, and results to queries are highlighted in it, using the same conventions as in the original Lifelines2. When we use our Lifelines2 with *VizPattern*, we remove the ARF tools from the screen so users see only the event list.

5.1 Participants and Evaluation Procedure

We carried out our evaluation over the Internet using the Amazon Mechanical Turk service, a marketplace for tasks with over 100,000 users in over 100 countries [15]. We recruited 50 users, and discarded responses from 8 subjects who did not attempt to perform all the required tasks or took less that 30 minutes to complete the evaluation. Our remaining 42 responses included subjects from America, India, Singapore, and Canada. 16 were graduate students, majoring in Computer Science, Electronic Engineer, Statistics, Law, MBA, and Civil Engineering. 12 work for government, and the rest are from various fields, such as graphic designer, accountant, and system engineer. Along with 22 graduate students who participated in our first user study a year earlier, we had 64 participants in total.

The data set used in this user study consists of synthetic scholastic records for 50 graduate students. Data are academic events such as "submitting a paper," "submitting a thesis proposal," "thesis defense," "taking a class exam," or "submitting a job application." The time span for all the records is 1 year. We intentionally used the same kind of data described in [21] to enable a meaningful comparison. However, our data set included data for 50 students, whereas the Lifelines2 study included data for only 20 students.

In order to carry out the user study over the Internet, we developed an evaluation program instrumented to record all user actions and times, and deployed it online as a Java WebStart application.

The evaluation was divided into 2 independent sections: one to evaluate Lifelines2, and the other to evaluate *VizPattern*. Users were asked to complete both sections, which the evaluation system presented in random order. Each section had three parts:

Part 1: Training. We prepared multimedia training materials for each system, including screenshots and tutorial videos. To ensure that participants completed and understood the training, the training materials included 4 quizzes testing various knowledge

points. Participants had to answer all correctly before entering the next part, so we can make sure the participants didn't abandon the training earlier. For a fair comparison, the training materials for both systems contained the same application domain, same examples, and same quizzes, but with a different data set.

Part 2: Evaluation Questions. The evaluation asked subjects to answer the same 5 questions using each system (users were warned that the data sets for the two sections were different). We used 3 questions from the Lifelines2 paper [21], slightly modified to accommodate our data set, but not to alter their difficulty. For the other 2 questions, we interviewed two staff members from the UCLA graduate school, and created questions that they needed to answer as part of their work.

- **Q1** (Lifelines2): How many students submitted a paper within 60 days after proposal?
- **Q2** (Lifelines2): How many students submitted at least 2 papers between proposal and defense?
- **Q3** (UCLA Staff): How many students had a class test and submitted a paper at the same time in the last 100 days of the year?
- **Q4** (Lifelines2): What occurred most often within 40 days of student's first paper submission?
- **Q5** (UCLA Staff): How many students had their defense within 80 days after submitting the job application? What about 60 days? What is the minimum number of days within which more than 1/5 of the students had their defense after submitting the job application?

The evaluation asked users to answer the questions and to describe their rationale for using the different system capabilities (the time participants spent in typing was ignored in the results.) The evaluation program recorded every action subjects took on the interface, and the time to answer each question, enforcing a 10 minute time limit. We instructed participants to use the pause button to stop the time recorder whenever they were interrupted during the evaluation.

Part 3: Survey. This part presents a 16 question survey from the Questionnaire for User Interaction Satisfaction (QUIS 7) [12].

5.2 Results and Discussion

We organize our results into an objective part where we report time and accuracy measurements, and a subjective part concerned with the user satisfaction survey.

We measured both the amount of time each participant spent answering each question, and the accuracy of the answers. We graded answers to each question as correct (1.0), incorrect (0.0) or skipped/give-up (0.0). For question 5, which has 3 sub-questions, we assigned scores 0.25, 0.25 and 0.5 respectively.

User Study Data. Figure 8 shows completion time results in seconds. The last column shows the times in percentage between VizPattern and Lifelines2. An asterisk symbol " * " marks statistically significant differences based on the paired-difference t-test with a 95% confidence level (p-value < 0.05).

Figure 9 shows that averaged over all users, *VizPattern* enabled users to answer every question faster and more accurately. *VizPattern* accuracy scores are higher on all questions, and except for question 2, the differences are statistically significant. More importantly, the number of incorrect answers for *VizPattern* users is below 20%, whereas except for Question 2, the number of incorrect answers for Lifelines2 users is above 50%.

Discussion. *VizPattern* required more training than Lifelines2. On average participants spent twice as much time learning *VizPattern* (23 minutes vs 7 minutes) . This is because *VizPattern* has a non-traditional interface design, which requires more time to master than a form-based interface as used in Lifelines2. Interestingly, participants made up the training time difference answering the 5 evaluation questions.

	Lifelines2	VizPattern	%
Training	424.7	1390.8	327.47% *
Question 1	305.6	83.4	27.3% *
Question 2	158.7	97.3	61.3% *
Question 3	398.2	133.8	33.6% *
Question 4	402.4	274.6	68.2% *
Question 5	509.9	188.4	36.9% *
All (Average)	**355.0**	**155.5**	**43.8% ***

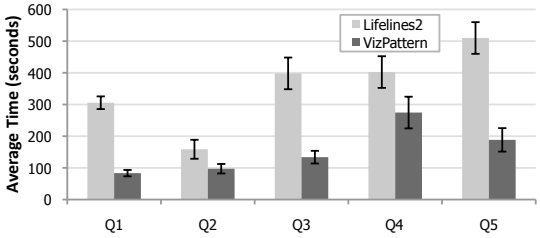

Figure 8: Average **time** to answer questions over all participants.

	Lifelines2	VizPattern	%
Question 1	0.46	0.97	210.9% *
Question 2	0.80	0.98	122.5%
Question 3	0.13	0.83	638.5% *
Question 4	0.46	0.79	171.7% *
Question 5	0.28	0.85	303.6% *
All (Average)	**0.41**	**0.87**	**212.2% ***

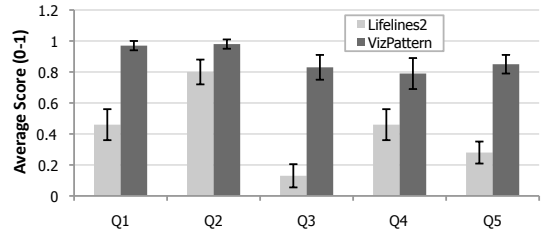

Figure 9: Average answer **accuracy** over all participants.

The large differences in time and accuracy are the result of the need to visually inspect records to determine answers. When users are able to represent the question in the system, they can simply read the answer off the screen. Sometimes, users are only able to use the features of the system to trim down the list of relevant records, and then they need to visually inspect the remaining records to determine the answer. Visual inspection takes time and is error prone.

Questions 1, 2, 3 and 5 can be answered in *VizPattern* without visual inspection, but only Question 2 can be answered in Lifelines2 without visual inspection. The data shows strong evidence that users of *VizPattern* were able to determine how to use the comic strip metaphor to encode the questions. They were able to answer questions 1, 2 and 3 in under two minutes, and answered the 3-part question 5 in about three minutes.

The *VizPattern* accuracy results for question 3 are relatively low, 79% compared to 97% and 98% for questions 1 and 2. This question is more complex because it combines two constraints, a same-time constraint for the events and an absolute time constraint. To answer it correctly, users must drag a "paper" and a "test" icon into the same panel, and insert a panel with a clock to represent the absolute time constrain. 80% of the participants were able to do this.

The Lifelines2 accuracy results for question 3 are surprisingly low. Lifelines2 users can use the align tool to make the counting process easier, but they must do so creatively. They must align by either first "class test" or "paper submission" event and then count. They must repeat this align and count procedure for the second,

third, and subsequent events. Only 13% of participants did this correctly. Question 1 also required visual inspection in Lifelines2, but the use of the alignment tool was significantly simpler, and 46% of participants were able to do it correctly.

The *VizPattern* accuracy results for question 5 suggest that many users understood the *VizPattern* capabilities and could use them creatively. This question cannot be encoded in the comic strip, but the capabilities of *VizPattern* can be used to answer it without visual inspection. The issue is that the "minimum number of days" part of the question cannot be encoded in the comic strip. However, the question can be answered by using *VizPattern* "backwards", i.e., using the dynamic query feature to vary the time constraint on the "defense" event until the number of matches is 10 (1/5 of the 50 students). The answer to the question is the value of the time constraint that results in 10 matches. 80% of participants answered this question correctly. Of those who answered correctly, 80% used this technique. In contrast, 90% of those who answered incorrectly used a different approach. Question 5 was difficult for the Lifelines2 users. 62.5% of them gave up whereas only 7.8% of the *VizPattern* participants gave up.

Question 2 was the only question that could be answered in Lifelines2 without visual inspection. Participants needed to use the "exact sequence" feature, but only 66% did. The rest used the "align and count" strategy, resulting in higher times and lower accuracy.

Question 4 required visual inspection in both systems. Lifelines2 participants used the align feature to align record by the first paper submission and visually inspected the results to count the different events that occurred within 40 days. In contrast, most *VizPattern* participants repeatedly queried for X happening within 40 days of paper submission, substituting X for the various events, and quickly inspecting the results to subtract those that were not for the 1st paper submission, a constraint that *VizPattern* cannot represent. The *VizPattern* strategy yielded a 79% accuracy (276 seconds) compared to the 46% accuracy (402 seconds) of the more burdensome visual inspection method used by Lifelines2 users.

The evaluation shows that increased expressivity is useful. Of course, increased expressivity is useful only if users are able to exploit it. The evaluation shows that users were able to exploit relative and absolute time constraints, and that the comic strip metaphor enables them to do so. Furthermore, when queries could not be fully represented, *VizPattern* participants devised creative uses of the comic strip capabilities to reduce the burden of visual inspection. The evaluation also shows that visual inspection is time consuming and error prone, which is the core factor that made the differences between two systems. We believe this factor won't change too much even if the users of Lifelines2 were experts, since most of people are likely to have the same problem no matter what their expertises are.

For the user satisfaction evaluation we used 16 questions from QUIS 7 [12], covering the categories shown in Table 1. *VizPattern* scored better in all categories, and the score difference is statistically significant (marked by "*"). The scores show that participants found *VizPattern* easy to use. Many reported that using *VizPattern* is "interesting" and "fun".

	Lifelines2	VizPattern
Overall Reaction *	3.54	8.00
Screen *	5.44	8.01
Terminology *	5.22	8.00
Learning *	6.04	6.94
System Capabilities *	4.40	7.54
Total Average *	**4.92**	**7.71**

Table 1: Average QUIS scores (9-point scale).

6 Applications

VizPattern is applicable in many domains. In our own work we used it to understand the behavior of agents in multi-agent simulations. In this section we present two additional case studies.

6.1 Case Study: Effects of Nifedipine

This case study is based on a published adverse drug reaction analysis (ADR) for nifedipine, a drug used to treat chest pain and high blood pressure [11]. The medical facts and analysis questions are taken from the published study. The data we used are synthetic.

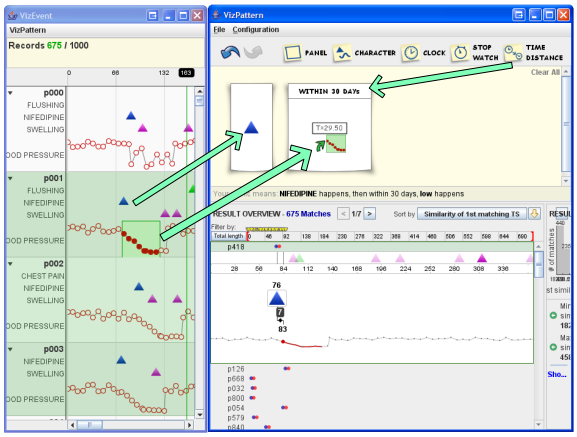

Figure 10: Visualizing patients' medical history related to nifedipine.

Figure 10 shows two of the windows used in the analysis. On the left, we show a visualization similar to Lifelines2 [21] to visualize 1000 patients' medical history data related to nifedipine, In the visualization, the prescription of nifedipine, chest pain, flushing, and swelling are denoted by blue, red, green, and magenta triangles. Each patient's blood pressure values are also recorded and visualized in line charts. On the right is the *VizPattern* window focusing on an analysis of the effects of nifedipine on blood pressure.

Nifedipine's Effect on Blood Pressure In the visualization we see that first patient had an uneven response to nifedipine, while the second through fourth exhibit a nice smooth response. The *VizPattern* window shows how we can find all patients who exhibit a similar smooth response. We drop a nifedipine icon in the Comic Strip Editor, then we drop in the segment of the blood pressure line chart that exhibits the smooth down-trending response, and then we impose a time interval constraint to require the lowering of the blood pressure to follow closely after the nifedipine event. 675 patients exhibited this response.

Nifedipine Adverse Drug Reactions To understand whether flushing is a common ADR of nifedipine, we drop nifedipine (blue triangle) and flushing (green triangle) icons into *VizPattern* workspace to construct a pattern "having flushing after taking nifedipine". 584 matches are returned, which means flushing is a common ADR of nifedipine.

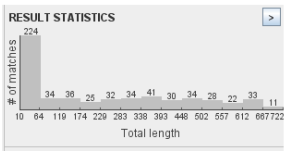

Figure 11: ADR analysis.

To further understand the relationship between nifedipine and flushing, we can sort the results by the gap between nifedipine and flushing. From the histogram view shown in Figure 11, we see that the majority of matches have a gap smaller than 64.

6.2 Case Study: DNA Microarray Data Analysis

A DNA microarray is an expression matrix that stores the expression level of genes in experimental samples. Each row is a gene, and each column is an experimental sample. Table 2 shows part of an Yeast expression matrix.

One practical problem in this field is identifying genes with similar differences in their expression levels. One example query is:

NAME	Primig SK1a/alpha 1h	Primig SK1a/alpha 2h
YHR055C	140	70
YCRX05W	NULL	258
...	...	...

Table 2: A DNA expression matrix example.

"find all genes where the level of sample s_1 is lower than that of s_2 at some value between 10 and 300, and the level of sample s_2 is lower than that of s_3 at some value between 200 and 400" [9].

We use a gene expression dataset downloaded from the Gasch Lab at University of Wisconsin-Madison. The dataset contains over 2000 genes and 90 samples. To simplify parsing the file, we renamed genes to $g1$, $g2$, ... and renamed samples to $s1$, $s2$, ... The insight to use *VizPattern* for this non-temporal data set is to treat the gene expression values as timestamps. We use the sample names (column) as event types, and gene names (row) as objects.

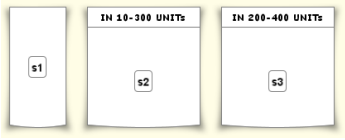

Figure 12: Gene query.

Figure 12 shows the comic strip that represents the sample query given above. It is simple, especially when compared with the equivalent SQL statements required to pull the data from a relational database. The result set contains 8 genes. Using the dynamic query capabilities users can analyze the effects of the thresholds.

7 Conclusion And Future Work

Our user study provides significant evidence that novice users can easily learn the comic strip metaphor for composing queries supporting our first goal (G1) of making it easy for users to answer questions. It also shows that they can creatively use it to help them answer queries even when it is not possible to fully represent the desired queries. Users utilized the capabilities of our second goal (G2) to incrementally perturb a question and immediately see how the answer changes. Our case studies provide further evidence that this metaphor is applicable in multiple domains where we leverage existing visualizations of temporal data (G3).

The user and case studies provide multiple illustrations of the benefit of using examples to define queries. We overcome limitations of example-based techniques by making the query specification explicit in the comic strip and by allowing users to directly edit it. The user and case studies show how Dynamic Query techniques add a new dimension of capability, enabling users to quickly answer questions that cannot be represented in the query language. The user and case studies show that integration of event and time series data is essential for many analyses.

The *VizPattern* work is by no means complete. Modern data sets often contain billions of data elements, and our current algorithms scale to one million elements. New algorithms and techniques are needed. This scalability problem is further complicated by the desire to increase the expressivity of the query language.

Increasing expressivity without compromising usability is particularly challenging. *VizPattern* already has features that enable more expressive queries than those evaluated in our user study. Additional user studies are needed to evaluate their usability. In addition, there are several expressivity limitations that we have not yet tackled. The most critical is the issue that the comic strip represents a single timeline, but many interesting queries need to be represented as a graph. To address this problem we are investigating the use of hyper-comics [10] that have some interesting features, such as non-linear breakdowns and spatial expansion.

We are also investigating new data analysis methods, such as comparing results from different queries, conducting queries directly on results, and performing set operations on different results.

References

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008.

[2] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[3] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. *Symposium On Visual Analytics Science And Technology*, 0:167–174, 2006.

[4] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[5] C. Holz and S. Feiner. Relaxed selection techniques for querying time-series graphs. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 213–222, New York, NY, USA, 2009. ACM.

[6] J. Jin and P. Szekely. Querymarvel: A visual query language for temporal patterns using comic strips. In *VLHCC '09: Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 207–214, Washington, DC, USA, 2009.

[7] M. Kindborg and K. Mcgee. Comic strip programs: Beyond graphical rewrite rules. In *DMS*, pages 327–332, 2005.

[8] D. Kurlander. Chimera: example-based graphical editing. pages 271–290, 1993.

[9] N. Mamoulis and M. L. Yiu. *Non-contiguous Sequence Pattern Queries*, pages 569–570. Springer Berlin / Heidelberg, 2004.

[10] S. Mccloud. *Reinventing Comics : How Imagination and Technology Are Revolutionizing an Art Form*. Harper Paperbacks, August 2000.

[11] G. N. Norén, A. Bate, J. Hopstadius, K. Star, and I. R. Edwards. Temporal pattern discovery for trends and transient effects: its application to patient records. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 963–971, New York, NY, USA, 2008. ACM.

[12] K. L. Norman and B. Shneiderman. The questionnaire for user interaction satisfaction. http://www.lap.umd.edu/quis/index.html.

[13] C. Plaisant, S. Lam, and B. Shneiderman. Searching electronic health records for temporal patterns in patient histories: A case study with microsoft amalga. In *Annual Symposium proceedings / AMIA Symposium*, pages 601–605, 2008.

[14] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1996. ACM Press.

[15] J. Pontin. Artificial intelligence, with help from the humans. http://www.nytimes.com/2007/03/25/business/yourmoney/25Stream.html.

[16] Rudensteiner. A visual query language for identifying temporal trends in video data. In *IW-MMDBMS '95: Proceedings of the International Workshop on Multi-Media Database Management Systems*, page 74, Washington, DC, USA, 1995.

[17] B. Shneiderman. Direct manipulation: A step beyond programming languages. pages 461–467, 1987.

[18] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Softw.*, 11(6):70–77, 1994.

[19] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer-Interaction, 3rd edition*. 2005.

[20] C. Traynor and M. G. Williams. End users and gis: a demonstration is worth a thousand words. pages 115–134, 2001.

[21] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 457–466, New York, NY, USA, 2008. ACM.

[22] M. Wattenberg. Sketching a graph to query a time-series database. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 381–382, New York, NY, USA, 2001. ACM.

[23] L. Xiao, J. Gerth, and P. Hanrahan. and p.hanrahan. enhancing visual analysis of network traffic using a knowledge representatio. In *IEEE Symposium OnVisual Analytics Science And Technology*, 2006.