QueryMarvel: A Visual Query Language for Temporal Patterns Using Comic Strips

Jing Jin, Pedro Szekely
Information Sciences Institute, University of Southern California
{jing, pszekely}@isi.edu

Abstract

In many domains, decision makers want to find and understand patterns of events as these patterns often give insight into the causal relationships among events. Current systems to specify patterns are either too difficult to use or only support simple patterns. We present QueryMarvel, an interactive visual query environment that allows ordinary users to easily and efficiently specify complex temporal patterns. It uses and extends the semantic elements and rules of comic strips with the goal to make specifying and interpreting temporal patterns as easy as reading and writing a comics story. User tests show that using QueryMarvel, users can understand and create temporal patterns faster and with fewer errors than using a forms-based interface.

1. Introduction

In the modern information age, the quantity and complexity of time-oriented data is rapidly and continuously increasing. The ability to specify and detect patterns of temporal events is fundamental to reasoning and making critical decisions in many domains. Temporal events are ubiquitous in medicine, finance, defense, weather, engineering, etc.

Visualizations are often used to help users discover and understand patterns because they leverage the ability of humans to process visual information [1]. Interaction techniques such as brushing, scrolling, zooming enable users to configure and explore the visualizations to better understand the data. However, helping users to "see" patterns in the visualizations is far from sufficient. Finding the first pattern instance is just the first step in an analysis. To more deeply understand the data, users want to find all instances of a pattern, and want to explore variations of the pattern. Doing so by visual inspection is inefficient, error prone and not reusable. Powerful visual analytics techniques are needed to support users in these scenarios.

For example, consider a medical trial to assess the effectiveness of a new drug. Suppose a database has been used to collect symptoms data of 1000 patients who have taken this medicine. A visualization similar to Lifelines2 [6] could be used to show each patient's data displayed in a strip, where different symptoms (e.g., headache, chest pain, stomach pain, swelling joints) are encoded using icons. While browsing a few examples, one might notice that several patients suffered of chest pain within 10 days of taking the medicine. Finding all instances of this pattern in the database of 1000 patients by visual inspection would be cumbersome, time consuming and error prone. The situation would be worse if the pattern were more complex (e.g., patients who exhibit swelling of their joints in 30 to 50 days after taking the medicine, and then have headache and chest pain at the same time).

In this paper, we present QueryMarvel, an easy to use and powerful visual query language to enable ordinary users to specify temporal patterns of events using comic strips. QueryMarvel utilizes and extends the semantic elements and rules of comic strips to construct complex temporal patterns. As a result, interpreting and creating patterns in QueryMarvel is similar to reading and writing a comic. The design of QueryMarvel is generic and can be used in different types of visualizations in different domains. The main contribution of this work is the use of comic strip metaphor in query language specification, which makes QueryMarvel an intuitive, easy to use, powerful and cross-domain visual query language.

The main challenge is to maximize expressiveness while keeping the language easy to use. We show that QueryMarvel represents a significant improvement over the state of the art towards meeting this challenge. Our user tests show that users can both understand and create QueryMarvel queries with a wide range of

complexity, and that they can do so faster and with fewer errors than using a form-based method.

This paper structure is as follows: section 2 presents related work, section 3 describes QueryMarvel in detail, section 4 discusses the evaluation and section 5 presents the conclusions and future work.

2. Related work

To help users easily explore temporal data, many visual methods for specifying temporal pattern/query have been proposed across different research domains. Generally, two visual methods are used to specify patterns: GUI widgets, and direct manipulation.

Dynamic Queries [2] is an early visual query system for database query. Users manipulate sliders and buttons to visually specify query criteria to specify queries consisting of conjunctions of disjunctions, and range queries on numeric values. It supports database very well, but does not support temporal event queries.

TVQL [3] 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 each pair of endpoints among two intervals in order to support Allen's 13 relational primitives [4]. According to the user studies in [3], the double-sided sliders are not intuitive enough.

PatternFinder [5] is designed to find patterns of events across multiple records. It uses forms to specify complex temporal patterns, and a ball-and-chain visualization to display matching results. Although PatternFinder has the ability to construct expressive temporal patterns, it is also complex to use. The learning curve is deep for inexperienced users.

Lifeline2 [6] introduces the idea of sentinel events (such as "the first heart attack"). Using sentinel events, users can easily spot precursor, co-occurring and after effect events, thus providing a way to explore patterns in medical data. It also provides ways to filter temporal data by specifying number of event occurrences, and simple sequence of events. However, it does not support expressing temporal relationships among events, which are useful in many domains.

TimeSearcher [8] lets users specify patterns of interest on line graphs by creating time-boxes to identify a value range and a time interval. The conjunctive queries are incrementally defined using multiple time-boxes. TimeSearcher2 [9] supports patterns with multiple variables and iterative flexible pattern searching. TimeSearcher3 [10] adds the river plot view and the forecasting tool. Systems using similar ideas also appear in visual data mining field. IVQuery [11] and IPBC [12] allow users to select an area of interest directly in visualizations as input. Based on the selection's attributes, it runs a similarity search and highlights the results. These systems are all intuitive to use, but not powerful enough to handle more complex temporal patterns. More importantly, they only work with time-series data, while we focus on categorical temporal data (events).

Comic strips are very popular in different cultures around the world. Its narrative grammar and universal and intuitive rules are used in several systems to provide a friendly user experience. A graphical editing system Chimera [13] uses comic strips to depict commands histories, showing the graphical state of the interface changing over time. A video summary system [15] introduces a video summarization and browsing algorithm that produces a comic-like representation of analyzed videos. A GIS [14] uses comic strips to demonstrate query procedures to non-specialist users. Like our system, these systems have a common design goal: using comic strips to convey a story which consists of a series events happening in a time order.

The comic strip metaphor has also been used in visual programming. ComicKit [17] allows children to use comic strips to make event-based programs. Characters, inner panels, signs, motion markers and voice balloons are mapped to program elements, and some extensions to comic strips are also introduced.

3. QueryMarvel

Comic strips are a form of sequential arts, defined as "juxtaposed pictorial and other images in deliberate sequence, intended to convey information" in [16]. As we all know, comics tell people stories. When people are reading comics, an implicit timeline is formed in reader's mind. The picture panels in a comic are viewed as a sequence of events positioned along this timeline, and the various comic elements, such as clocks, text bubbles, decorations and etc, tell readers extra information about those events.

In QueryMarvel we think of each event as a character in a comic, and each temporal relationship as certain comic element. This converts a highly abstract temporal pattern specification into a concrete comic strip that everybody is familiar with. Understanding a temporal pattern is similar to reading a comic strip, and creating a temporal pattern is similar to creating a comic strip using a given cast of characters.

Throughout the rest of this paper we will make reference to our simple medical scenario of 1000 patients participating in a drug trial for Medicine X. The events of interest are chest pain, stomach pain, and nausea as Figure 1 shows:

Figure 1: Four example event types

In the next section we introduce the formal temporal event pattern language underlying QueryMarvel.

3.1. Temporal event pattern

The semantics of the temporal event pattern language are captured in the following definitions, which are adapted and extended from [18].

Definition 1: *Let t be a real value timestamp, S be a set of event types, and O be a set of objects. An* ***event*** *is defined as a triple* (t, o, s)*, where* $o \in O$*, and* $s \in S$*. An* ***event sequence*** T *is defined as a series of event* E_i *ordered by* $t(E_i)$*, and a maximum time value* t_M*, such that* $\forall E_i \in T$, $0 \le t(E_i) \le t_M$*. One special event is defined: a sequence start event* E_s=(0, *seq, seqStart*)*. It is always included in every event sequence.*

Definition 2: *A* ***query pattern*** *is defined as a triple* $<Z, C_T, C_O>$ *where Z is a set of* ***event types*** $Z=\{Z_0, Z_1, \ldots Z_m\}$, $Z_i \in S$, $i=0,1,..,m$. C_T *is a set of* ***temporal constraints*** C_{Tij}, $i,j=0,1,\ldots,n$, *each of which is defined by an interval* $[a_{i,j}, b_{i,j}]$, *and* C_O *is a set of* ***object constraints*** C_{Oi} *defined over a set of event E.* C_{Oi} *could be a unary equity U(m,o), or a binary equity/inequity BE(m,n)/BIE(m,n). U(m,o) is satisfied if* $o(E_m)=o$*. BE(m,n)/ BIE(m ,n) is satisfied if* $o(E_m)$ *is equal to (not equal to)* $o(E_n)$*.*

The above pattern definitions can be used to define a complex temporal pattern like: "patients who have chest pain within 10 days after taking medicine X, followed by a stomach pain and nauseous at the same time between the 80th and 100th day".

Using our definitions, the event type set Z := {Z_0=seqStart, Z_1=take med X, Z_2=chest pain, Z_3=stomach pain, Z_4=nauseous}

The temporal constraints between events are C_{T12}=[0,10];C_{T34}=[0,0];C_{T03}=[0,80];C_{T04}=[100, +inf)

Finally, the object constraints are:

C_{O0}=U(0,seq);C_{O1}=BE(1,2);C_{O2}=BE(2,3);C_{O3}=BE(3,4)

3.2. Visual metaphor

In this section, we describe how temporal patterns are mapped into a comic strip in QueryMarvel. For each element of the pattern language 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: comic strips consist of a series of panels laid out along an invisible timeline. To tell a story, characters inside the panels are shown doing something at some time. Similarly, in QueryMarvel, each event type icon becomes a character.

Temporal constraint 1: Event B happens after event A: As a form of sequential arts, comic 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 2, it is clear that the woman reaches inside the crib after having stood next to the crib looking inside it. Similarly, the two QueryMarvel panels show stomach pain after chest pain.

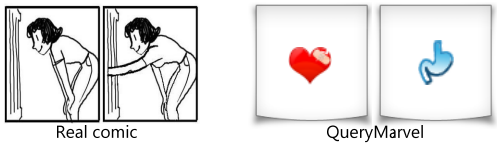

Figure 2: "after" temporal constraint

Temporal constraint 2: 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 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.

Figure 3: "same time" temporal constraint

Temporal constraint 3: Event A happens after event B with specified time distance: 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 distance constraints. For example, Figure 4 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.

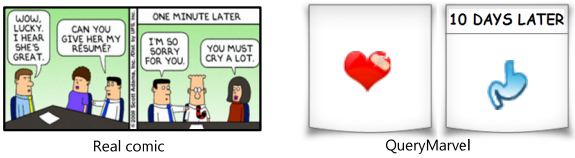

Figure 4: "time distance" temporal constraint

The text label can be used to represent a variety of temporal relationships such as "in over 10 days", "within 10 days", or "10-50 days", which correspond to the intervals [10, +inf), [0, 10], and [10,50].

Temporal constraint 4: 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. QueryMavel uses the same technique to represent absolute time constraints. For example, Figure 5 shows the man checking the office supplies at 9:30. Similarly, the QueryMarvel panel shows chest pain happening on the 10th day of the medial trial.

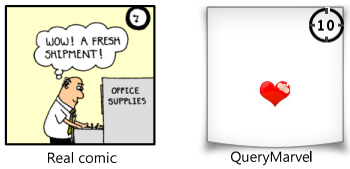

Figure 5: "absolute time" constraint

Temporal constraint 5: Event happens during an absolute time range: This is easily represented in QueryMarvel using the building blocks defined above. Figure 6 shows an example of a chest pain event happening between the 10th and 100th 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.

Figure 6: "absolute time range" constraint

This technique is very flexible as it can be used to define flexible before and after constraints. For instance, if we remove the first panel from Figure 6, then its meaning becomes "having a chest pain before the 100th day". Besides, it allows bracketing an arbitrary number of panels between the clocks.

The mappings described above show how several constraint types can be mapped directly to the comic strip metaphor. However, in order to fully support our temporal pattern language described in section 3.1, we need to extend the existing comic strip concept.

Extension 1: negation: based on our experience, the negation of event plays an important role in temporal event patterns. For example, it is desirable to represent patterns such as something other than stomach pain happens after taking medicine X. QueryMarvel represents negation by putting a red cross over an event icon. The left picture in Figure 7 shows "something other than stomach pain happens".

Extension 2: OR: As described before, putting several event icons into one panel means they happen together at some time (AND). QueryMarvel supports OR constraints using the grid panel. The cells in the grid represent the branches of the OR, and it is possible to insert multiple events in each cell. This represents a disjunction of conjunctions. The right picture in Figure 7 shows an example where chest pain OR stomach pain happen at some time.

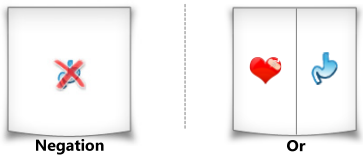

Figure 7: "Negation" and "Or" extension

Extension 3: object constraints: The default object constraint in QueryMarvel is to match all events on the same object. However, it is often useful to represent temporal constraints that relate events on different objects (owners). For example, in a system monitoring domain, each object might represent different equipment, such as "server 1" and "server 2". Users want to specify patterns across different objects, such as "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, 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. Figure 8 illustrates the use of tags. The first part shows a constraint where one server goes down and then a different server goes down. The second part shows the case where specific servers go down one after the other.

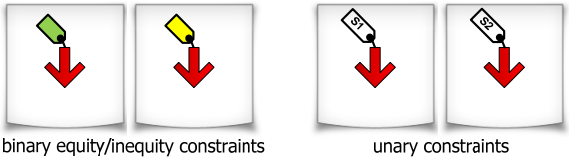

Figure 8: Object constraints

For convenience, if no tag is attached to event icons, then it means they match to the same object.

We have introduced the building blocks of QueryMarvel. To illustrate the expressiveness and intuitive nature of QueryMarvel we show how it can be used to specify the following complex temporal pattern: "patients who take medicine X between 10th and 50th day, and then within 7 days have chest pain and stomach pain at the same time, followed by feeling nausea for two consecutive days". Figure 9 shows the QueryMarvel comic strip. The panels occupy several rows, as do the panels on a page of a comic book.

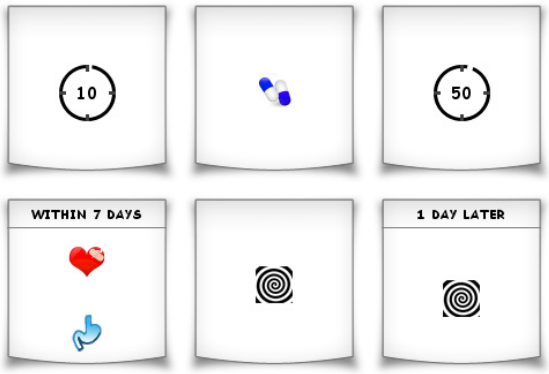

Figure 9: QueryMarvel Example

3.3. The QueryMarvel user interface

As shown in Figure 10, the QueryMarvel interface consists of four major parts:

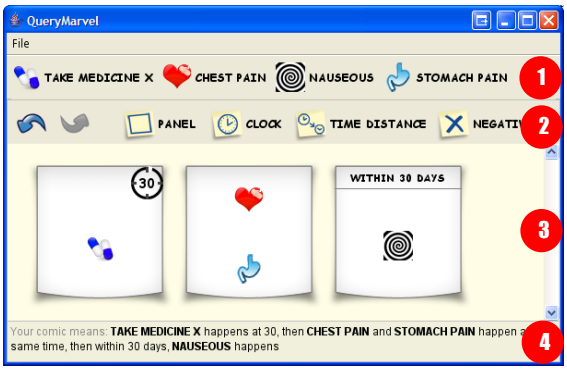

Figure 10: QueryMarvel Interface

1. Event picker panel: It displays all the available event icons that represent the event types relevant to a domain. Users drag icons from here to create their own patterns.
2. QueryMarvel creation toolbar: A set of comic creation tools are provided, including "Undo/Redo", "adding empty panel", "adding clock", "adding time distance", and "adding negation".
3. QueryMarvel content panel: the QueryMarvel comic strip is displayed and created here.
4. QueryMarvel info panel: This panel translates the displayed comic into English. This helps users verify that their comic strip matches their intention and helps users learn the system.

Using the QueryMarvel interface, users can easily read, create and edit temporal patterns. Due to the page limitation, we are unable to describe the interface in detail. A demonstration and video are at: http://coordination.isi.edu/QueryMarvel.

4. Evaluation

4.1. Evaluation design

The goal of our evaluation is to show that ordinary users can use QueryMarvel to both understand and create temporal event patterns with a wide range of complexity, and that they can do it faster and with fewer errors than an alternative form-based method.

Our evaluation compares QueryMarvel with PLForm, a forms-based interface to specify temporal patterns. The syntax of PLForm is based on the entire model described in section 3.1. By using PLForm, we are able to free participants from remembering the complex syntax of the pattern language and typing. While PLForm is our own creation, it is representative of existing form-based query systems such as [2][5][6].

Figure 11 shows an example of PLForm. The left panel is used to define events: each line defines an event, and has a menu to select the event type. Buttons allow users to easily add and delete event types. To keep things simple, PLForm does not provide a mechanism to specify object constraints. The right panel is used to define temporal constraints: each line represents a temporal constraint. Menus allow users to select the events related by the constraint, and a text field enables users to enter the relationship as an interval (in our evaluation we did not penalize users for syntax errors). Buttons allow users to easily add and delete temporal constraints.

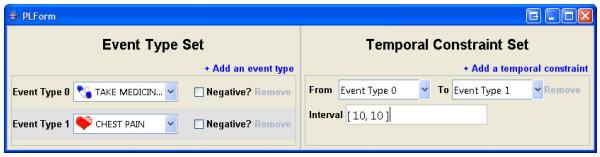

Figure 11: PLForm

26 people participated in the evaluation: 22 graduate students and 4 faculty members from various departments including Electrical Engineering, Computer Science, Economics, Education, Biology, Business, Theater, Law, Civil Engineering, History and Management. All participants use computers every day. Only 7 came from the computer science department.

The evaluation was conducted over the Internet. The evaluation software was instrumented to record all user interface actions, and was packaged so that it could be launched from a Web page. The evaluation software included a pause button so that users could interrupt the evaluation if they needed to do so. Upon completion, the evaluation software automatically uploaded the evaluation results to a server.

The data and domain used in evaluation is similar to the example presented in section 3. The only difference is that the evaluation used 8 events instead of the 4 used in the examples above.

4.2. Evaluation procedures

The evaluation is divided into two sections, one for PLForm and another for QueryMarvel. The order of them is randomly selected (15 participants got the QueryMarvel section first). Each section has four parts:

Part 1: Training. In this part, training materials are provided in web pages. Text, images and video clips are used to explain the basic concepts of each system, and to present examples to help participants understand the concepts better. In order to make the evaluation as accurate as possible, the example data we used in the training is different from the data used in the tests. We record the total time each participant takes to finish reading the training materials.

Part 2: Pattern Interpretation. In this part, 9 questions with three difficulty levels are given to the participants. Each screen contains only one question. In each question, a system screenshot is shown, and participants are asked to write down the meaning of the patterns specified by the screenshot in plain English. We ask participants not to worry about the spelling, structure and use of words in the final answer.

Questions are divided into three levels: 3 questions for easy patterns containing 1-2 events and simple temporal constraints (no absolute time constraints); 4 for medium level patterns containing 2-3 events and moderate temporal constraints (including absolute time constraints); and 2 for hard patterns containing all temporal elements. The questions are ordered from easy to hard. The time spent in each screen is accumulated and recorded. A sample screen from the PLForm section is shown in the top part of Figure 12.

Part 3: Pattern specification. In this part, 9 questions of three levels are given to the participants. In each question, a temporal pattern is described in plain English, and participants are asked to specify it using PLForm or QueryMarvel. The criteria of different levels are the same as part 2. To make the results more accurate, the patterns in sections 2 and 3 are different. The time spent in each question is recorded**.** A sample screen from the QueryMarvel section is shown in the bottom part of Figure 12.

Part 4: Survey. This part presents a survey with 16 questions from the Questionnaire for User Interaction Satisfaction (QUIS 7) [19]. We also asked participants to provide feedback about their experience of using the two systems.

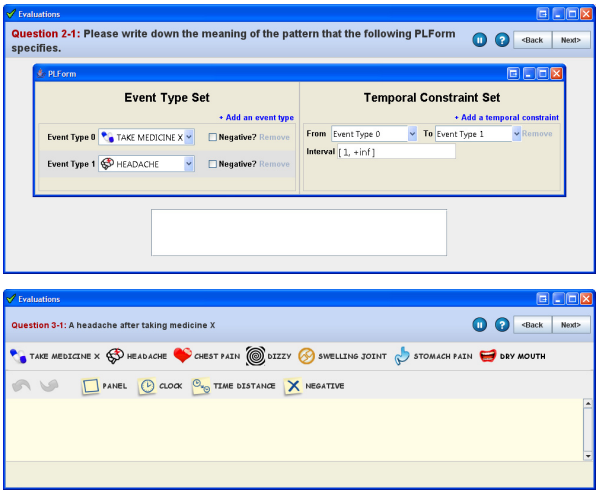

Figure 12: Evaluation question screens

4.3. Results and discussion

Our evaluation compares the effectiveness of QueryMarvel and PLForm by comparing the times to complete the training, interpretation and pattern specification sections of the evaluation, by comparing the error rates in the interpretation and specification sections, and by comparing the user satisfaction scores.

Efficiency evaluation: Table 1 shows the average of the times (in seconds) that users spent on the training, interpretation and specification sections of the evaluation. The table breaks down the interpretation and specification data into the easy, medium and hard task levels. The last column shows the percentage difference of the PLForm minus QueryMarvel times (negative numbers indicate that users completed the tasks faster using QueryMarvel). An asterisk symbol "*" marks statistically significant differences based on the paired-difference *t*-test with a 95% confidence level (p-value < 0.05). Figure 13 plots the mean values of time spent in each group of table 1 (except training) with standard error bars.

Table 1: Time summary

	PLForm	QueryMarvel	% diff
Training	550.3	752.9	36.7% *
Pattern Interpretation (average time in each level)			
Easy	48.44	38.72	-20.0%
Medium	177.52	66.56	-62.5% *
Hard	265.69	125.57	-52.8% *
All	**154.09**	**70.39**	**-54.5% ***
Pattern Specification (average time in each level)			
Easy	72.34	33.23	-54.1% *
Medium	92.90	49.90	-46.3% *
Hard	185.95	79.25	-57.3% *
All	**114.17**	**55.90**	**-51.8% ***

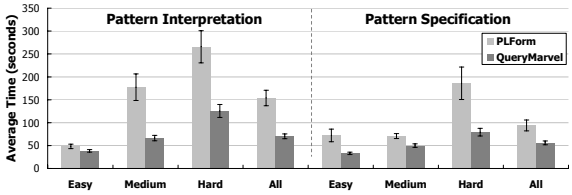

Figure 13: Time means with error bars

The results show that participants spent 3 to 4 additional minutes learning QueryMarvel than PLForm (36.7% more). This is because QueryMarvel uses a new interaction paradigm that the training materials had to cover, whereas participants were familiar with forms, so there was no need to train them on how to operate the PLForm user interface.

The time differences for the interpretation and specification tasks are all statistically significant except for the easy interpretation tasks.

Users did not make up the training time difference during the 9 interpretation and specification tasks, but if the differences hold as users perform more tasks, it would require only 40 interpretation or specification tasks to make up the training time difference. However, more importantly, as the next section shows, users of QueryMarvel made significantly fewer errors.

The results also suggest that QueryMarvel effectiveness increases as the complexity of the patterns increases. The time difference between the easy level and the hard level using QueryMarvel is only 46 seconds, whereas in PLForm this number is 113 seconds. We believe the reason is that PLForm users must bridge a bigger representation gap between a pattern written in English and a form, where as the gap to the QueryMarvel comic strip is much smaller.

Accuracy evaluation: In this part, we graded answers as correct (1.0), partially correct (0.5) and incorrect (0.0). We gave partial score when the temporal events are all correct, but when temporal constraints were reversed, when there were off-by-one errors or there were redundant temporal constraints.

Table 2 shows the accuracy results. The data is broken down and analyzed for statistical significance as in the previous section. Figure 14 plots the mean values of the scores with standard error bars.

Table 2 and Figure 14 show that QueryMarvel achieved higher accuracy scores in interpretation and specification for all levels of difficulty. The differences are statistically significant on the aggregate scores and in 4 out of the 6 levels.

61.1% of all errors occurred because participants treated absolute time ranges as relative temporal constraints. This happened both in PLForm and QueryMarvel, especially in the specification tasks. The reason may be that the English descriptions of these two kinds of temporal constraints use similar words, and may have been confusing to many participants.

Table 2: Accuracy summary

	PLForm	QueryMarvel	% diff
Pattern Interpretation (average score)			
Easy	0.82	0.97	18.3% *
Medium	0.75	0.93	24.0% *
Hard	0.61	0.68	11.7%
All	**0.74**	**0.90**	**20.3% ***
Pattern Specification (average score)			
Easy	0.86	1	16.5% *
Medium	0.70	0.78	12.9%
Hard	0.29	0.56	96.4% *
All	**0.66**	**0.81**	**22.7% ***

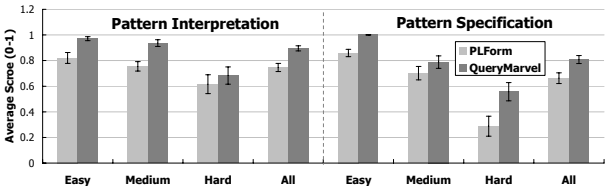

Figure 14: Score means with error bars

13.9% of all errors occurred because many participants wrote down the temporal intervals in reverse order in PLForm. This is likely to happen if participants didn't pay enough attention to the detailed definitions in the training materials. However, this kind of error was never made in QueryMarvel, where temporal intervals are clearly displayed using easy to understand text boxes instead of abstract intervals.

Another interesting observation is that in one of the PLForm interpretation questions, we checked two negation checkboxes in 2 of 4 events, and 42% of the participants didn't see them. However, in a similar question in QueryMarvel, 100% of the participants see the negation sign (the red cross) and provided the correct answers. This shows that the use of visual elements in QueryMarvel can effectively help users understand temporal patterns better and more accurate.

User satisfaction evaluation: In this part, we surveyed participants on 16 questions taken from the QUIS 7 [19]. The questions covered the user satisfaction categories shown in Table 3. QueryMarvel scored better in all categories, and the difference in score is statistically significant (marked by "*").

The largest difference comes from the Learning category, which suggests that users found QueryMarvel easier to learn, even or perhaps because they spend more time training on it.

Several users reported using QueryMarvel was fun, and many provided suggestions, which we discuss in the next section.

Table 3: average QUIS scores (9-point scale)

	PLForm	QueryMarvel
Overall Reaction *	5.58	7.94
Screen *	7.02	8.52
Terminology *	6.47	7.78
Learning *	5.52	8.40
System Capabilities *	6.76	8.12
Total Average *	**6.14**	**8.08**

5. Conclusion and future work

In this paper, we proposed a new visual query language QueryMarvel, which uses and extends conventions and concepts from comic strips to enable users to intuitively and efficiently specify temporal patterns of events. A user study shows that using QueryMarvel, ordinary users can interpret and specify temporal patterns faster with less errors compared to an alternative form-based method PLForm. The user experiences are very satisfactory.

Several participants observed that certain patterns are impossible to specify in QueryMarvel, but can be easily specified in PLForm. For example, "within 20 days of taking medicine X, patients get a headache. And within 30 days of taking medicine X, patients get stomach pain". This example shows several events happening simultaneously on more than one timelines. QueryMarvel cannot represent parallel timelines because the default panel layout implies a single timeline. We will investigate extensions to QueryMarvel to support parallel timelines without destroying its easy of use and intuitive nature.

Further user studies are needed to evaluate the effectiveness and ease of use of the tags concept to encode object constraints.

In ongoing work we are integrating QueryMarvel into various visualization systems to enable users to easily compose and make temporal queries from within those systems.

6. References

[1] J. J. Thomas and K. A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005.

[2] Shneiderman, B. 1994. "Dynamic Queries for Visual Information Seeking". *IEEE Softw.* 11, 6 (Nov. 1994), 70-77.

[3] S. Hibino and E. Rudensteiner. "A visual multimedia query language for temporal analysis of video data". In *Multimedia Database Systems: Design and Implementation Strategies*, pp. 123-159. Kluwer Academic Publishers, 1996.

[4] Allen, J. F. 1983. "Maintaining knowledge about temporal intervals". *Commun. ACM* 26, 11 (Nov. 1983), 832-843.

[5] J. A. Fails, A. Karlson, L. Shahamat, B. Shneiderman, "A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories", *in Proc. of IEEE Symposium on Visual Analytics Science and Technology*, Baltimore, USA, 2006.

[6] Wang, T. D., Plaisant, C., Quinn, A. J., Stanchak, R., Murphy, S., and Shneiderman, B. 2008. "Aligning temporal data by sentinel events: discovering patterns in electronic health records." *In Proc. of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy, April 05-10, 2008). CHI '08. ACM, New York, NY, 457-466.

[7] Wattenberg, M. 2001. "Sketching a graph to query a time-series database". In *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (Seattle, Washington, March 31 - April 05, 2001). CHI '01. ACM, New York, NY, 381-382.

[8] Hochheiser, H. 2003. "Interactive Graphical Querying of Time Series and Linear Sequence Data Sets". *Doctoral Thesis*. University of Maryland at College Park.

[9] Buono, P., Aris, A., Plaisant, C., Khella, A., and Shneiderman, B., (2005). "Interactive Pattern Search in Time Series", *In Proc. of Visualization and Data Analysis*, VDA 2005, SPIE, Washington DC (2005) pp. 175--186.

[10] Buono, P., Plaisant, C., Simeone, A., Aris, A., Shneiderman, B., Shmueli, G., Jank, W., "Similarity-Based Forecasting with Simultaneous Previews: A River Plot Interface for Time Series Forecasting" *Proc. 11th International Conference on Information Visualisation*. Zurich, Switzerland; 2-6 July, 2007.

[11] Ming C. Hao, Umeshwar Dayal, Daniel A. Keim, Dominik Morent, Joern Schneidewind, "Intelligent Visual Analytics Queries", *VAST 2007*, 91 – 98

[12] Chittaro, L., Combi, C. and Trapasso, G., "Data mining on temporal data: a visual approach and its clinical application to hemodialysis". *J Visual Lang Comput*. v14 i6. 591-620.

[13] Kurlander, D. 1993. "Chimera: example-based graphical editing". *In Watch What I Do: Programming By Demonstration*, MIT Press, Cambridge, MA, 271-290.

[14] Traynor, C. and Williams, M. G. 2001. End users and GIS: a demonstration is worth a thousand words. *In Your Wish Is My Command: Programming By Example* Morgan Kaufmann Publishers, San Francisco, CA, 115-134.

[15] Janko Calic, David Gibson, Neill Campbell, "Efficient Layout of Comic-Like Video Summaries". *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7). ISSN 1051-8215, pp. 931–936. July 2007.

[16] S. Mccloud, *Understanding Comics*. New York: HarperPerennial, 1994.

[17] Kindborg, M., McGee, K. (2005). "Comic Strip Programs: Beyond Graphical Rewrite Rules". *International Workshop on Visual Languages and Computing*, Banff, Canada, 5-7 September 2005.

[18] Mamoulis, N., Yiu,M.L.: "Non-contiguous sequence pattern queries". *In: Proceedings of the 9th International Conference on Extending Database Technology*. (2004)

[19] QUIS the Questionnaire for User interaction Satisfaction, http://www.lap.umd.edu/quis/index.html.