To appear in *ACM Multimedia'97 Conference Proceedings*, Seattle, WA USA.

User Interface Evaluation of a Direct Manipulation Temporal Visual Query Language

Stacie Hibino †
Bell Labs/Lucent Technologies
1000 E. Warrenville Road
Naperville, IL 60566 USA
hibino@acm.org

Elke A. Rundensteiner
Computer Science Department
Worcester Polytechnic Institute, 100 Institute Road
Worcester, MA 01609-2280 USA
rundenst@cs.wpi.edu

ABSTRACT. As new query interfaces emerge for accessing multimedia data, formal user studies are needed to evaluate the usability of such interfaces. In this paper, we present results from a user interface evaluation of our temporal visual query language (TVQL). TVQL is a novel direct manipulation query interface for specifying temporal relationship queries over temporal events such as video data. In our user study, we compare TVQL to a forms-based temporal query language (TForms). Our results indicate that while subjects took longer to learn TVQL than TForms, they were more efficient and more accurate in specifying temporal queries with the TVQL interface than with the TForms interface.

KEYWORDS

Temporal visual query language, temporal query filters, dynamic queries, user interface evaluation.

1. INTRODUCTION

Recent advances in multimedia databases have introduced new multimedia query languages and interfaces empowering users to access multimedia data in novel ways such as querying images by submitting either color and spatial specifications or sample images as search criteria (e.g., [14, 12]). While the description of such systems usually includes evaluation of their efficiency and accuracy, little work, if any, has been done to evaluate the *usability* of these new query interfaces. Formal user evaluations are needed to demonstrate the usability of such systems—to validate not only that the *system* can efficiently and accurately access multimedia data, but that *real users* can use the corresponding new query interfaces to efficiently and accurately find relevant information in multimedia data sets.

Our MultiMedia Visual Information Seeking (MMVIS) environment represents a new visual paradigm for the temporal trend analysis of video data [7, 9]. In MMVIS, we provide a direct manipulation interface enabling users to dynamically browse through temporal relationships between user-specified subsets of video events stored in a database. This interactive browsing is supported by tightly coupling specialized temporal query filters (referred to as TVQL, our temporal visual query language—see [10] and summary in Section 2.1) with a dynamically updated temporal visualization (TViz) of results.

Consider the case where HCI researchers collect CSCW video data to analyze the process flow of a design meeting between three subjects ("Cindy," "Ron," and "Greg") collaborating from remote sites [11]. The data is coded to indicate when each person speaks as well as to characterize the design activity based on the design rationale (DR) of what is being said (e.g., to indicate when alternatives, digressions, etc., take place in the meeting). Researchers could select two subsets within MMVIS such as: A) person talking events and B) design activities. They could then *explore* various relationships between members of these subsets using TVQL. As they manipulate the temporal query filters, TViz is dynamically updated to visually present the query results by indicating the strength of the temporal relationship specified through TVQL (see [8, 9]). For example, they could use TVQL to investigate temporal trends such as who starts talking at the same time as each of the design activities, who starts talking before or at the same time as each design activity but does not stop speaking until after the given activity starts (i.e., who "initiates" them), etc.

Previously, we have demonstrated the utility of MMVIS through a case study applying it to real CSCW video data such as the design meeting described above [8]. We have recently completed two user interface studies on MMVIS—one evaluating the usability of TVQL and the other on the effectiveness of the fully integrated MMVIS environment [7]. In this paper, we describe the results of our first user interface study on evaluating TVQL.

In the TVQL user study, we compare and contrast the users' ability to specify and interpret various types of temporal queries using TVQL versus a forms-based temporal query language (TForms). The study shows that while users spent more time learning TVQL than TForms, they were also able to specify temporal queries more efficiently (significantly more efficiently for incremental temporal queries) and more accurately with TVQL than with TForms. While TVQL is designed to be part of our integrated MMVIS environment, it is a general interface for specifying temporal relationship queries that could be applied to temporal data other than video (e.g., logfiles or stock exchange data) as well as incorporated into other database frameworks of temporal data.

This paper is divided into six additional sections. In the next section, we provide some background on temporal relationship queries and describe the two query interfaces being compared—our temporal visual query language (TVQL) and a forms-based temporal query language (TForms) interface. In Section 3, we

†This work was completed while the author was at the University of Michigan.

describe the experimental design of the study, presenting the number and types of participants, the overall design, the procedure and materials, and the types of data collected. In Section 4, we then summarize our experimental results and in Section 5, we discuss the implications for changing the TVQL interface based on these results. In Section 6, we present some related work and finally, we provide conclusions in Section 7.

2. TEMPORAL QUERY INTERFACES COMPARED

2.1 Temporal Relationship Queries

Given two events A1 ( ) and B1 ( ) with nonzero duration, there are thirteen possible primitive temporal relationships between them such as: *before, meets, during, starts, overlaps,* etc. [3]. Although there are four pairwise relationships between temporal starting and ending points of the events (e.g., start A1 - start B1), only one to three of these relationships are required to define any one of the thirteen temporal primitives (see Figure 1).

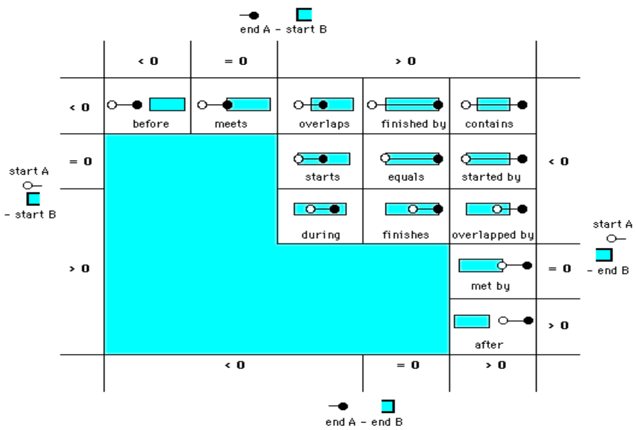

Figure 1. Relationships between temporal primitives and the four defining endpoint difference relations.

A general temporal query language should be able to specify any one of these primitives. In addition, it is also desirable to specify combinations of the primitives (e.g., to see how often events start at the same time but may end at different times, corresponding to combining the *starts*, *equals*, and *started by* primitives). Rather than making arbitrary combinations of such relationships, users may wish to query over *similar* primitives (i.e., *temporal neighborhoods* [6], equivalent to selecting a series of adjacent cells such as a row, column, or grid from Figure 1).

A set or combination of temporal relationships between two events forms a *temporal neighborhood* [6] if it consists of relations that are path-connectable conceptual neighbors. Two primitive temporal relationships between two events are *conceptual neighbors* if a continuous change to the events (e.g., shortening, lengthening, or moving the duration of the events) can be used to transform either relation to the other (without passing through an additional primitive temporal relationship) [6]. Thus, the *before* ( ) and *meets* ( ) relations *are* neighbors, because we can move the ending point of A from before the start of B to B's start without specifying any additional primitive relationship. On the other hand, the *before* ( ) and *overlaps* ( ) relations are *not* neighbors.

2.2 TVQL: Temporal Visual Query Language

While a formal specification of our temporal visual query language (TVQL) can be found elsewhere [10], we review its basic design principles here. In order to define a temporal query interface capable of specifying any individual primitive temporal relationship, we designed TVQL to be a collection of four temporal query filters—one filter for each of the defining endpoint difference relationships (see Figure 1). More importantly, this design also allows us to capture *temporal neighborhoods* (see Section 2.1). In this way, not only can users browse for temporal relationships between two subsets, but they can browse in a *temporally continuous* manner. More specifically, TVQL provides a direct manipulation interface where users can explore within and between primitives as well as within and between temporal neighborhoods through simple mouse manipulations. Figure 2 presents our TVQL [7, 10]. As in the standard double-thumbed slider query filters [1], the thumbs are manipulated to select the endpoints of a range, and a filled or open arrow thumb indicates when the endpoint of a range is included or excluded respectively.

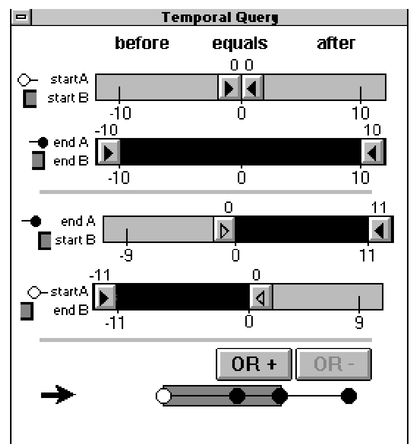

Figure 2. TVQL palette. This query specifies all events of type A that start at the same time as events of type B (and may end before, after, or at the same time as B events).

Similar to standard dynamic query (DQ) filters [1], our temporal DQ filters are bound to one another to prevent the specification of invalid queries. As users adjust one query filter, the other filters are automatically updated accordingly. In the case of Figure 2, the user only has to set the left and right filter thumbs of the top startA-startB query filter to 0. The bottom two filters are automatically constrained as indicated.

To enhance the TVQL user interface, we have incorporated qualitative descriptive labels along the top and side and our dynamic temporal diagrams along the bottom of the palette. The labels allow users to "read" the relationship specified and the diagrams provide visual confirmation of the temporal primitive(s) specified (though not quantitative values as given by the filters). If subset A specified person P1 and subset B specified all Plan design rationales, then Figure 2 illustrates how users could ask the query "show me how often person P1 starts at the same time as a Plan starts." The descriptive labels can be used to "read" the top query filter as "start A equals start B." The relationship between the temporal ending points is unconstrained as indicated by the selection of all values in the second (i.e., endA-endB) query filter. This is also reflected in the temporal diagram, which indicates that the end of A (represented by a filled circle) can be before, equal to, or after the end of B.

The benefit of this direct manipulation design is that it supports specifying *particular* queries as well as *browsing* for temporal trend discovery. That is, users can simply drag DQ thumbs with no particular query in mind and when an interesting visualization appears, they can look at the temporal diagram to see which temporal query was specified.

2.3 TForms: A Forms-Based Temporal Query Language

While many researchers are working in the area of temporal query languages (e.g., [13, 15]), they have focused more on *text-based* approaches rather than visual or forms-based ones. These existing text-based temporal query languages provide users with the power to specify a larger range of queries than TVQL (see [7]), but they also require users to learn the syntax and semantics of the query language as well as to type the queries in by hand. Although the goal of the TVQL user study was to compare TVQL to an existing temporal query language, we did not want to unfairly handicap non-TVQL subjects with extra typing and memorization burdens. We thus chose to provide them with a *forms-based* temporal query language that is based on a text-based one. In this section, we describe this forms-based temporal query language which we refer to as TForms.

The basic syntax of a TForms query is based on a subset of the syntax for the temporal query language TQuel [16] that allows users to specify the same types of queries that are possible with TVQL. Figure 3 presents a sample TForms query that specifies the *starts* temporal query.

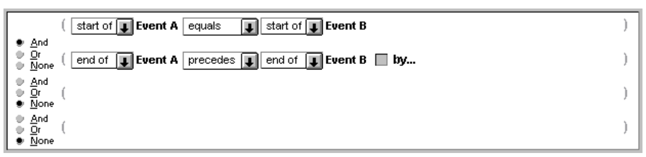

Figure 3. Sample TForms query used to specify the *starts* temporal query.

The *buttons along the left side* of TForms allow users to add a temporal query predicate (i.e., to include an additional temporal constraint per line) to their query via conjunction (*AND*) or disjunction (*OR*), or to remove a temporal predicate by selecting the *NONE* button at the left of the corresponding line. *Parentheses* allow users to group lines of their query together. Users can click the left mouse button directly on a parenthesis to toggle it on and off. The *first part* of each line of the query—up to and including "Event B"—specifies the *qualitative* relationship between temporal endpoints. Users specify this first part of a query predicate (e.g., query line 1 in Figure 3) by selecting a value from each of the three drop-down boxes. The *second part* of each line allows users to specify *quantitative* ranges, if necessary. Users can toggle the quantitative option on and off by clicking on the "*by...*" checkbox button. This quantitative option is only available when the relationship of the second drop-down box of the line is *not* set to "equals." Figures 6 and 7 include examples of TForms that use a quantitative option.

3. EXPERIMENTAL METHOD

3.1 Participants

Twenty undergraduate and graduate students (fifteen male and five female) participated in the study. The study was advertised through the student newspaper, flyer postings, and electronic mailing lists. Subjects were selected based on their expertise and experience in either video analysis (VA) or databases (DB). They were paid ten dollars an hour for a maximum of thirty dollars. All subjects had at least five years of computer experience and were familiar with the Macintosh and/or Windows operating systems.

3.2 Design

A 2x2 between subjects, counterbalanced design was used to compare the two interfaces (TVQL versus TForms) for each type of subject (DB versus VA). The type of the interface used was alternately assigned to each subject within each group. Thus, subjects were placed in one of four groups, depending on which interface they used and what type of expertise they had: TVQL-VA, TVQL-DB, TForms-VA, and TForms-DB (see Table 1).

Table 1. Description of the four user groups.

		Interface Tested	
		TVQL	TForms
Subject's Expertise	Video Analysis	Group TVQL-VA	Group TForms-VA
	Database	Group TVQL-DB	Group TForms-DB

Subjects in each group performed all tasks for the given user interface used, with the only constraint being the maximum time limit of three hours.

3.3 Procedure and Materials

At the start of each testing session, subjects were asked to complete background information and consent forms. Then, for each interface—TVQL and TForms, there were four primary parts:

- Part I. *Training Session* where users were presented with computerized training materials,
- Part II. *Query Interpretation* section where users were presented with a series of multiple choice queries for which they had to select the best English text query that matched a query specified in the given query interface,
- Part III. *Query Specification* section where users specified a series of temporal queries in the given interface, and
- Part IV. *Post Questionnaire on User Satisfaction* which was essentially a slightly shortened version of the Generic User Interface Questionnaire (also referred to as QUIS), version 5.0 [5] consisting of twenty-three questions.

In Part I, the online training materials consisted of 12 information screens for the TForms interface and 16 screens for the TVQL interface. The training materials for each interface followed the same outline, beginning with an introduction and scenario, providing some background on temporal queries and the different types of temporal relationships (primitive, neighborhood and disjunctive), describing the interface and how to use it, providing some hands-on practice time, and presenting a series of examples. Subjects were allowed to go back and forth through the screens of the training materials.

Part II, the query interpretation section, was not only used as a test of the users' understanding of the given interface, but also as a prerequisite to the query specification section (Part III). When answering the multiple choice questions, subjects clicked on the letter corresponding to their answer and then clicked on a "Check Answer" button to receive immediate feedback on the accuracy of their answers. In order to better test whether subjects really were able to correctly interpret a query (versus correctly answering the multiple choice question by guessing), we required subjects to correctly answer each question twice. In the TVQL interface, users had two types of interpretation tasks—one to test their interpretation of temporal diagrams (Figure 4), and one to test their interpretation of TVQL queries (Figure 5).

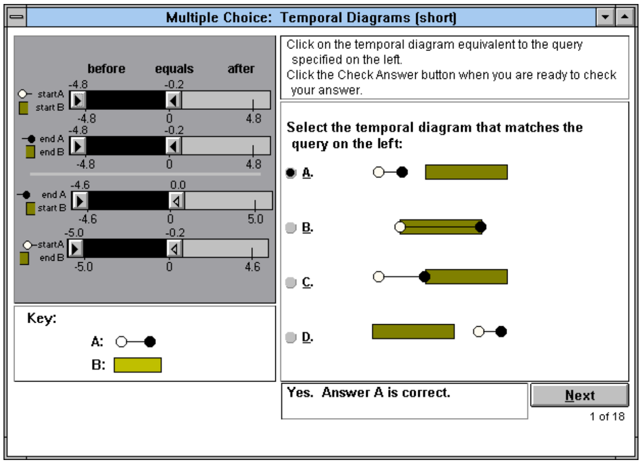

Figure 4. Sample multiple choice question for interpreting temporal diagrams.

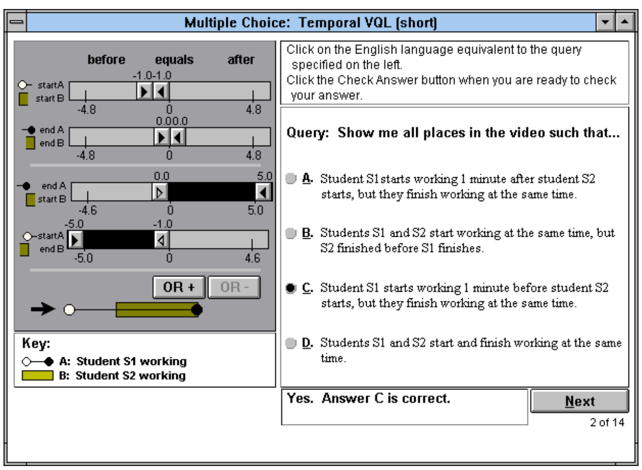

Figure 5. Sample multiple choice question for interpreting a TVQL query.

As temporal diagrams are not part of the TForms interface, subjects only had to interpret TForms queries during the query interpretation section (Part II). A sample query interpretation question for the TForms interface is presented in Figure 6.

In the temporal diagram query interpretation task of Part II for TVQL subjects, two types of temporal queries were tested—temporal *primitive* queries and temporal *neighborhood* queries. In the TVQL and TForms query interpretation tasks, *disjunctive* temporal queries were tested in addition to temporal primitive and neighborhood types of queries. (Since the temporal diagrams of a disjunctive query are a combination of primitive and neighborhood queries, they were not tested during the temporal diagram interpretation task.)

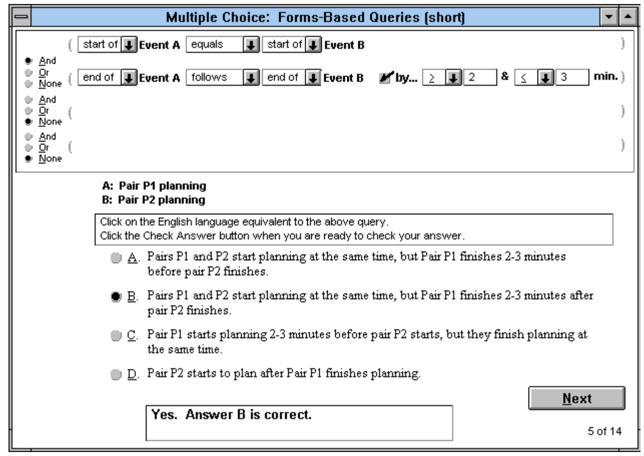

Figure 6. Sample multiple choice question for interpreting a TForms query.

All questions in Part II were grouped by query type (i.e., by temporal primitive, neighborhood, or disjunction). Each group of questions was randomly presented to the subjects once, then randomly presented in a different order a second time, and then followed by any questions that the subjects incorrectly answered. The multiple choice answers were randomly presented to the users so as to discourage them from memorizing only the letter of the correct answer. Table 2 summarizes the minimum number (i.e., when users do not make any errors answering the multiple choice questions) and type of temporal query interpretation questions included in Part II for each interface. Note that TVQL subjects answered the 30 temporal diagram interpretation questions followed by the 32 TVQL interpretation questions (62 questions total) while the TForms subjects only answered the 32 TForms interpretation questions.

Table 2. Minimum number and type of temporal query interpretation questions included in Part II.

UI Tested	# of Primitive Queries	# of Neighbor-hood Queries	# of Dis-junctive Queries	Total # of Queries
Temporal Diagrams	9x2	6x2	-	30
TVQL	7x2	6x2	3x2	32
TForms	7x2	6x2	3x2	32

During query specification (Part III), subjects specified the same three types of temporal queries that were tested in Part II (i.e., temporal primitive, neighborhood, and disjunctive temporal queries) as well as an additional group of *incremental* queries (i.e., a series of queries representing temporal browsing). Two sequences of incremental queries, presented in the same order to all subjects, were included for both interfaces. In each sequence of incremental queries, users specified the first query in the sequence from scratch and then specified each subsequent query by starting from the previously specified query.

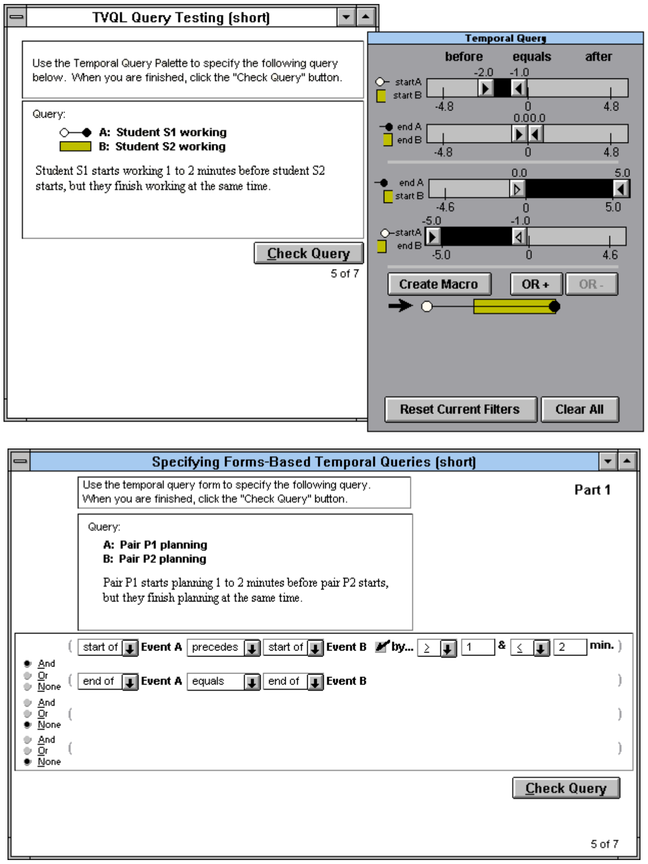

Figure 7. Sample isomorphic TVQL and TForms queries.

The TVQL and TForms queries were isomorphic so that equivalent, though not identical, queries were specified in each interface (see Figure 7). In both interfaces, some of the queries to be specified in Part III were exact repetitions of those interpreted in Part II thereby testing users on both familiar and new queries. The number and type of queries specified during Part III for each interface is summarized in Table 3.

Table 3. Summary of the number and type of temporal queries specified in Part III (Rep=Repeated queries).

UI Tested	# of Primitive Queries		# of Neighbor-hood Queries		# of Disjunctive Queries		# of Incr. Qs	Total # of Qs
	Rep.	New	Rep.	New	Rep.	New		
TVQL	4	3	3	3	1	2	7	23
TForms	4	3	3	3	1	2	7	23

At the end of the testing session, the same post-questionnaire survey, based on twenty-three questions of QUIS [5], was provided for each interface. Finally, a brief informal interview was conducted to clarify or add to any comments on the post-questionnaires and to allow subjects to ask any other questions regarding the study.

Hardware and Software Setup. All computerized materials, including the training modules and both user interfaces, were developed in *Asymetrix Multimedia ToolBook 3.0*. The testing sessions were conducted on Dell Pentium 90 (90 MHz Pentium) desktop machines equipped with 17-inch SuperVGA monitors and running Microsoft Windows NT.

3.4 Types of Data Collected

We collected several types of data during user testing, including: background questionnaires, logfiles, observational data, post-questionnaires on user satisfaction, and informal post-interviews. The background questionnaires were used to gather information on subjects' educational background, knowledge and use of computers, and experience with databases and video analysis. The logfiles recorded the time spent on training and testing materials as well as the answers given during testing. The post-questionnaires included 23 of the 26 rating scales from QUIS version 5.0 [5] on overall reaction, learning, features of the screen, system terminology, and system capabilities (3 of the 26 ratings were excluded since they were not applicable). In addition, the post-questionnaires contained open-ended questions and were used to collect information on what subjects liked or disliked about the system, what they thought was easy or difficult about the interface, and any final comments they had about the system such as specific suggestions for improving the interface. Finally, the informal post-interviews were conducted to clarify subjects' comments on the post-questionnaires, allow them to provide any additional comments, and give them an opportunity to ask any questions about the study.

4. RESULTS AND DISCUSSION

We compare TVQL to TForms for both the video analysis (VA) and database (DB) subjects based on three criteria: efficiency, accuracy, and user satisfaction. We thus divide the presentation and discussion of our results based on these criteria.

4.1 Efficiency Evaluation

We evaluate the efficiency of the interfaces by comparing and contrasting the times taken by each group of subjects to complete each of the first three parts of the study, including time taken to: 1) review training materials, 2) interpret temporal queries through the online multiple choice test(s), and 3) specify temporal queries. The first two columns of numbers in Table 4 summarize the average times in minutes that subjects in each interface group used to complete their assigned tasks in Parts I to III. In analyzing task times, we found no significant differences between the VA and DB subjects within each type of interface and thus grouped these subjects by interface. The final column of the table summarizes the differences in times between the same types of subjects using the two types of interfaces, where a *positive* difference indicates the percent *more* time required by TVQL over TForms and a *negative* difference indicates the percent *less* time required by TVQL. An asterisk (*) is used to indicate significant differences ($p < 0.05$) based on a Scheffe post-hoc analysis.

Recall that during the query interpretation part of the study, TVQL subjects had to answer a series of questions on interpreting temporal diagrams as well as those on interpreting TVQL queries (e.g., Figures 4 and 5). Referring back to Table 2, we see that the temporal diagram interpretation task thus required TVQL subjects to answer a minimum of thirty questions more—almost double the number of questions—than the TForms subjects during the query interpretation part of the study. Thus, in Table 4, we provide two summaries to compare average times for the total time spent on Part II (query interpretation) for each interface—one for comparing the average combined times on temporal diagram (TD) and TVQL multiple choice tests to the average times spent on the TForms multiple choice test, and one for comparing a summary of average multiple choice test times for *TVQL only* to TForms.

Table 4. Average total task times (in minutes). (TD=Temporal Diagram; *=statistically significant ($p < 0.05$) for Scheffe post-hoc analysis)

	TVQL (time in min.)	TForms (time in min.)	TVQL-TForms (% diff.)
Part I. Training Time			
	26.7	**15.4**	**73.1 %***
Part II. Query Interpretation			
	(TD + TVQL)	**(TForms only)**	
Primitive	16.2	12.6	28.4 %
Neighbor	11.5	9.0	27.7 %
Disjunctive	3.4	4.9	-31.5 %
Total	**31.1**	**26.5**	**17.0 %**
	(TVQL only)	**(TForms only)**	
Primitive	6.7	12.6	-46.6 %
Neighbor	5.1	9.0	-43.8 %
Disjunctive	3.4	4.9	-31.5 %
Total	**15.2**	**26.5**	**-42.8 %***
Part III. Query Specification			
Primitive	9.1	9.9	-8.5 %
Neighbor	7.8	9.6	-18.7 %
Disjunctive	5.7	5.8	-2.2 %
Incremental	4.0	8.1	-50.4 %*
Total	**26.5**	**33.3**	**-20.5 %**

Using Table 4 to compare TVQL to TForms on average task times, we see that:

- TVQL subjects spent significantly more time on training materials than TForms subjects (73.1% more time),
- the differences between the total query interpretation times (i.e., times for TD + TVQL versus TForms multiple choice questions) spent during Part II are *not* statistically significant,
- TVQL subjects spent significantly less time (42.8% less time) than TForms subjects on the same number and types of query interpretation questions (i.e., Part II, TVQL only versus TForms only task times), and
- TVQL subjects spent significantly less time (50.4% less time) than TForms subjects when specifying incremental queries during Part III.

These results indicate that although TVQL subjects spent more time learning TVQL than TForms subjects spent learning TForms, the TVQL subjects were able to use TVQL more efficiently than the TForms interface in interpreting all types of queries and in specifying incremental queries. The largest efficiency advantage of TVQL over TForms is in specifying *incremental* queries, where TVQL subjects took about half the time of TForms subjects to specify these types of queries. TVQL was especially designed to handle such incremental queries and we thus expected TVQL subjects to perform better on these types of queries than TForms subjects. The results confirmed our expectations and showed that TVQL provides a major efficiency gain over TForms for such incremental queries.

The longer training time required for TVQL over TForms could be a reflection of the complexity of TVQL and/or potential problems with the training materials. We expect the learning time to decrease as we iterate on improving the TVQL interface and training materials and plan to continue evaluating TVQL training time in the future.

4.2 Accuracy Evaluation

In this section, we compare TVQL subjects to TForms subjects in terms of accuracy. We grouped the results of VA and DB subjects by interface, based on a Scheffe post-hoc analysis that revealed no significant differences between VA and DB subjects within each interface. Table 5 summarizes the average number of errors made by each interface group for each part of the query interpretation (Part II—multiple choice) task. No significant differences were found between TVQL and TForms subjects. The table shows that all subjects made an average of less than three errors (out of a minimum total of 30 questions). Thus, subjects were able to use their given interface to *interpret* temporal queries with fairly high accuracy.

Table 5. Average number of errors in multiple choice (Part II—query interpretation) task.

	Temporal diagrams	TVQL	TForms
Multiple Choice			
Primitive	0.6	0.9	1.0
Neighbor	1.0	0.7	0.4
Disjunctive	n/a	0.7	0.4
Total	**1.6**	**2.3**	**1.8**

In evaluating subjects' accuracy during query specification (Part III), we characterized each user query according to the following coding scheme:

- *correct:* query that was correctly specified.
- *qualCorrect:* query that was *qualitatively* correct (e.g., was the same temporal primitive) but *not quantitatively* correct.
- *partCorrect:* query that is not *qualCorrect* nor a *neighbor* when compared to the actual correct query, but for which *part* of the query has been correctly specified.
- *neighbor:* query that represents a temporal "neighbor" to the actual answer (e.g., specifying the *before* relationship instead of the *meets* relationship).
- *reverse:* query that is the reverse or symmetric query to the actual answer (e.g., specifying the *meets* relationship instead of the *met by* relationship).
- *syntaxError:* code for TForms subjects only. An incorrect query that is *not qualCorrect* nor a *neighbor* compared to the actual answer. A *syntaxError* is based on missing or incorrect

syntax (e.g., missing parenthesis, use of OR rather than AND, etc.).

- *incorrect:* query incorrectly specified that does not match any of the above codings.
- *none:* no query specified.

The bar chart in Figure 8 compares the answer accuracy of subjects using TVQL to those using TForms, based on the above coding scheme. Again, no significant differences were found between the VA and DB subjects within each interface, so these subjects were grouped together by interface. A Scheffe post-hoc analysis showed that the differences in accuracy between TVQL and TForms subjects were significant ($p < 0.05$) for qualCorrect, partCorrect, and neighbor answers. That is, TVQL subjects specified a significantly larger number of queries that were qualCorrect than TForms subjects, while TForms subjects specified a significantly larger number of queries that were evaluated to be partCorrect or neighbor than TVQL subjects.

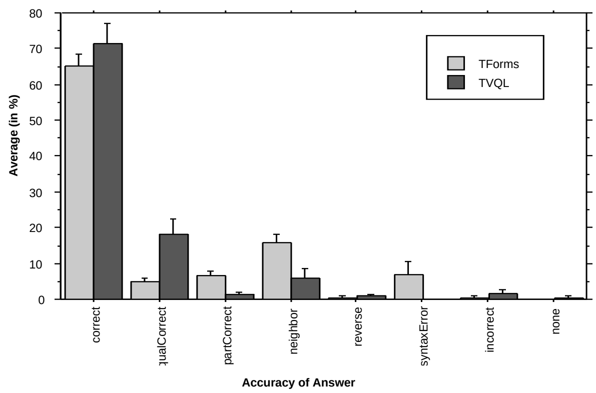

Figure 8. Comparison of average answer accuracy (in %) and standard error bars for TVQL versus TForms subjects.

Overall, TVQL subjects reached an average correct + qualCorrect accuracy of almost 90 percent. A Scheffe post-hoc analysis also revealed that the difference between TVQL and TForms subjects on correct + qualCorrect accuracy is significantly in favor of TVQL ($p < 0.05$). The advantage of TVQL over TForms for specifying a query with qualCorrect accuracy (i.e., specifying the query *qualitatively* correctly) indicates that TVQL subjects were able to successfully use the temporal diagrams as feedback to their specified query.

Comparing all types of errors made by TForms subjects, we see that the top three types of errors made by TForms subjects were *neighbor*, *syntaxError*, and *partCorrect*. The range and quantity of errors without visual feedback during query specification supports the use of visual feedback (e.g., a temporal diagram) as an aid to query specification accuracy.

4.3 User Satisfaction Evaluation

QUIS version 5.0 [5] organizes user satisfaction ratings into five categories: overall reaction, learning, screen, terminology, and system capabilities. Since our post-questionnaires are essentially a slightly shortened version of QUIS, we thus use these categories to compare and contrast user satisfaction ratings for TVQL and TForms.

Table 6 summarizes the average ratings of the TVQL and TForms interfaces based on a five-point scale. Again, no significant differences between DB and VA subjects were found within each interface and subject ratings were thus grouped by interface. A post-hoc Scheffe analysis indicated no significant differences for user satisfaction ratings between TVQL and TForms subjects. The largest difference between the interfaces is in the category of learning and is in favor of TForms. This again indicates the need to decrease the learning curve for TVQL.

Table 6. Average user satisfaction ratings per QUIS category and for QUIS total, based on a five point scale.

	TVQL	**TForms**
Overall Reaction	3.4	3.5
Learning	3.6	4.1
Screen	4.3	3.9
Terminology	4.2	3.9
System Capabilities	4.5	4.2
QUIS Total Average	**3.9**	**3.9**

5. PROPOSED CHANGES TO TVQL

TVQL subjects had three salient comments and reactions to the interface which they expressed in both the post-questionnaires and during the informal post interviews:

- they liked and understood the temporal diagrams,
- they had a hard time deciding which temporal query filter to manipulate, even though they knew the temporal diagram they wanted to ultimately obtain, and
- they wanted to manipulate the temporal diagram directly.

The advantage of the temporal diagram is that it provides a compact, visual, *qualitative* representation of the temporal query specified. The difficulty and danger in pushing the *quantitative* values from the query filters directly down to the temporal diagram is that the diagram could become more cluttered, more complex, and potentially less helpful. Thus, rather than attempt to make radical changes to the TVQL interface, we propose a set of smaller changes to provide users with better starting points of reference.

Figure 9 shows the new TVQL interface that we propose and have incorporated into MMVIS for the second user study (see [7]). In this new interface, we made two types of enhancements—we added color to the sliders to help users determine which query filter to manipulate and we provided shortcuts to key positions of the query filters and temporal diagram. In terms of color enhancement, since *startA* is represented by a *white* circle in the temporal diagram, we emphasized the two query filters related to *startA* (i.e., the top startA-startB query filter and the bottom startA-endB query filter) by coloring the corresponding labels, outlining the corresponding filters, and filling the range between the filter thumbs of the corresponding filters with the color white. Now, rather than having a 25 percent chance of selecting the desired query filter to manipulate, users have a 50 percent chance (provided that they know which circle of the temporal diagram they want to attempt to move).

We added shortcuts to the “0” or “equals” position of the temporal query filters and corresponding shortcuts to the temporal diagram. Each of the four temporal query filters has one “0” shortcut below it, and a corresponding “0” shortcut in the temporal diagram. More specifically, the top left white “0” above the temporal diagram corresponds to the “0” below the top startA-startB query filter, the top right white “0” of the diagram corresponds to the “0” below the bottom startA-endB query filter, the bottom left black “0” below the diagram corresponds to the third endA-startB filter, and finally, the bottom right black “0” below the diagram corresponds to the “0” below the endA-endB filter.

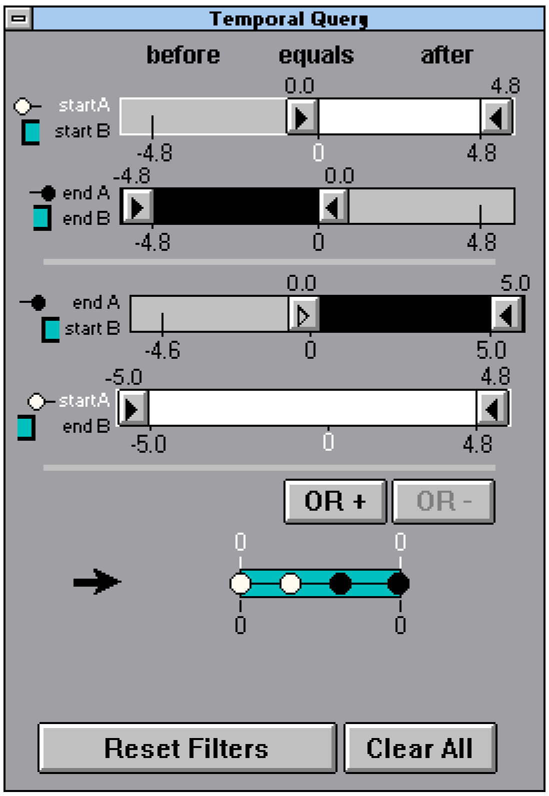

Figure 9. New TVQL interface with color and shortcut enhancements.

The “0” shortcuts below the query filters allow users to set the left thumb of the corresponding filter to zero with a left click on the “0” marker below the filter and to set the right thumb to zero with a right click on the “0” marker. Left or right clicking on a “0” in the temporal diagram has a one-to-one correspondence to left or right clicking the “0” of its corresponding query filter. Clicking on a “0” in the temporal diagram also has the visual effect of moving all circles of the same color as the “0” in towards the “0” marker—where a left click brings circles to the left of the marker in and a right click brings circles to the right of the marker in towards it. Thus, a right-click on the top left white “0” above the temporal diagram in Figure 9 is equivalent to a right-click on the “0” marker of the top startA-startB query filter. Such a right-click would move the right thumb of the startA-startB filter to the “0” position and would be visually reflected in the temporal diagram by bringing all white circles in the temporal diagram to the right of that top left “0” marker in to the point below it (i.e., it would hide the startA position that appears after or to the right of startB). The result of such a right click is depicted in Figure 10. These shortcuts provide mechanisms for users to “zoom” endpoints in towards a desired position.

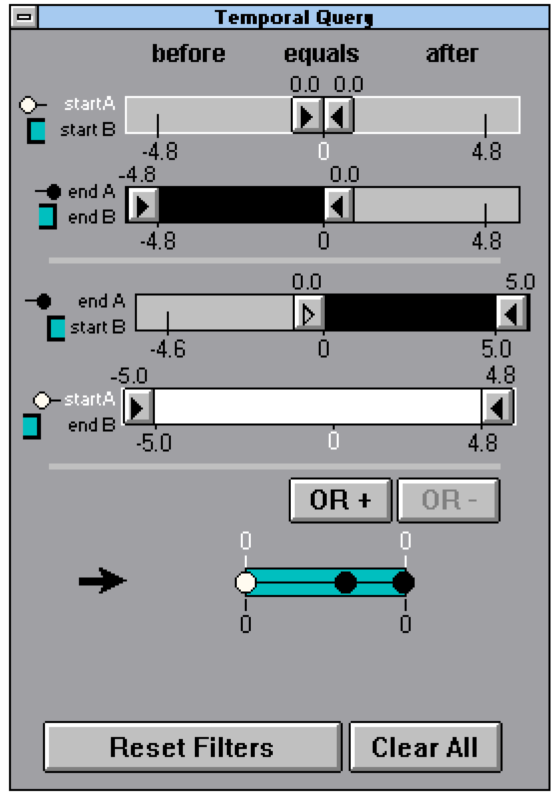

Figure 10. State of TVQL after subject uses a right-click on the first white zero above temporal diagram in Figure 9.

6. RELATED WORK

TVQL and MMVIS represent unique extensions to dynamic query filters and visual information seeking (VIS, [1]) for the purpose of temporal trend analysis of video data. Previous studies evaluating VIS and dynamic query interfaces based on applications of the VIS paradigm to chemistry data (periodic table of the elements [2]) and real estate data [17] have demonstrated the power of this direct manipulation approach for various query tasks ranging from finding a particular data item to searching for data trends and exceptions to trends.

In the VIS user study over chemistry data, Ahlberg et al [2] compared the use of a dynamic queries interface to a forms-based query language with a visualization of results (FG) and a forms-based query language with textual output (FT), in a within-subjects design. Their overall results showed that on average, subjects were significantly faster at query tasks when using the dynamic queries interface over the FG ($p < 0.005$) and FT interfaces ($p < 0.001$). They also showed that subjects made fewer errors using the dynamic queries interface than when using the FG or FT interfaces.

When comparing a dynamic queries interface to natural language and paper-based interfaces in a real-estate example, researchers also found results supporting the VIS paradigm over alternative query interfaces [17]. Their results show subjects were significantly more efficient in using the dynamic query interface for four out of five query tasks ($p < 0.005$). They also show that subjects gave the dynamic queries interface a significantly higher rating in a user satisfaction post questionnaire ($p < 0.005$).

Catarci and Santucci [4] present another user study evaluating a direct manipulation query interface that also demonstrates advantages of a visual query language over a text-based query interface. In their study, they compare QBD*, a diagrammatic query language based on a conceptual data model to SQL, a traditional textual query language. In this study, their overall results show that naive and intermediate users were significantly more accurate when using QBD* over SQL ($p < 0.05$) and that intermediate and expert users were significantly more efficient (i.e., had a significantly lower completion time) in specifying queries using QBD* than SQL ($p < 0.05$).

7. CONCLUSION

In this paper, we used four criteria to compare and contrast the usability of TVQL to a forms-based temporal query language (TForms): learning time, efficiency and accuracy in specifying temporal queries, and user satisfaction ratings. We compared the interfaces in a 2x2 between subjects experimental design and found no significant differences between the two types of subjects who participated in the study (i.e., video analysis (VA) and database (DB) subjects) for any of the criteria tested. We thus grouped the two types of subjects together and focused our evaluation on comparing the TVQL and TForms interfaces, independent of the subjects' expertise.

Our results showed that while TVQL subjects spent more time on training materials and thought that TVQL was relatively difficult to learn, they also scored better in efficiency and accuracy than TForms subjects. More specifically, TVQL subjects were greater than 40 percent more efficient than TForms subjects in answering the same number and type of questions during query interpretation. TVQL subjects were also more than 50 percent faster than TForms subjects in specifying incremental temporal queries. In terms of accuracy, TVQL subjects reached almost 90 percent accuracy in specifying queries correctly or qualitatively correctly (i.e., average of correct + qualCorrect queries specified), a significantly larger number than that of the TForms subjects.

We proposed to enhance TVQL with additional color aids and shortcuts available directly on the query filters and temporal diagram. These changes were based on users' comments on the use of the temporal diagrams and query filters. The new TVQL was integrated into the full MMVIS environment before a second user interface study was conducted on MMVIS [7].

While TVQL is part of our integrated MMVIS environment designed for the temporal trend analysis of video data, we re-emphasize that it is a general interface for specifying temporal relationship queries that can be applied to temporal data other than video as well as incorporated into other database frameworks of temporal data. The results of this user interface study are thus not only relevant to our MMVIS environment, but also to other applications requiring a temporal query interface for investigating temporal relationship queries or temporally browsing data in search of data trends.

ACKNOWLEDGMENTS

This work was supported in part by a University of Michigan Rackham Thesis Grant.

REFERENCES

1. Ahlberg, C., and Shneiderman, B. (1994). Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *CHI'94 Conference Proceedings,* NY:ACM Press, 313-317.
2. Ahlberg, C., Williamson, C., and Shneiderman, B. (1992). Dynamic Queries for Information Exploration: An Implementation and Evaluation. *CHI'92 Conference Proceedings*, NY:ACM Press, 619-626.
3. Allen, J.F. (1983). Maintaining Knowledge About Temporal Intervals. *CACM*, 26(11), 832-843.
4. Catarci, T. and Santucci, G. (1995). Diagrammatic vs. Textual Query Languages: A Comparative Experiment. In *Visual Database Systems 3*, Elsevier Science Publishers, 69-83.
5. Chin, J., Diehl, V. and Norman, K. (1988). Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. *CHI'88 Conference Proceedings*. NY:ACM Press, 213-218.
6. Freksa, C. (1992). Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54, 199-227.
7. Hibino, S. (1997). *MultiMedia Visual Information Seeking.* University of Michigan PhD dissertation.
8. Hibino, S. and Rundensteiner, E. (1997). "Interactive Visualizations for Temporal Analysis: Application to CSCW Multimedia Data." In *Intelligent Multimedia Information Retrieval* (Mark Maybury, Ed.), Cambridge, MA:MIT Press, 313-335.
9. Hibino, S. and Rundensteiner, E. (1996a). MMVIS: Design and Implementation of a Multimedia Visual Information Seeking Environment. *ACM Multimedia'96 Conference Proceedings.* NY:ACM Press, 75-86.
10. Hibino, S., and Rundensteiner, E. A. (1996b). "A Visual Multimedia Query Language for Temporal Analysis of Video Data," *Multimedia Database Systems: Design and Implementation Strategies* (K. Nwosu, B. Thuraisingham, and P.B. Berra, Eds.). Norwell, MA: Kluwer Academic Publishers, 123-159.
11. Olson, J., Olson, G., and Meader, D. (1995). What Mix of Audio and Video is Important for Remote Work. *CHI'95 Conference Proceedings,* NY:ACM Press, 362-368.
12. Pass, G., Zabih, R., and Miller, J. (1996). Comparing Images Using Color Coherence Vectors. *ACM Multimedia'96 Conference Proceedings.* NY:ACM Press, 65-73.
13. Seshadri, P., Livny, M. and Ramakrishnan, R. (1994). Sequence Query Processing. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*. NY: ACM Press, 430-441.
14. Smith, J.R. and Chang, S. (1996). VisualSEEk: A Fully Automated Content-Based Image Query System. *ACM Multimedia'96 Proceedings*, NY: ACM Press, 87-98.
15. Snodgrass, R. (1995). *The TSQL2 Temporal Query Language* (Snodgrass, R., Ed.). Norwell, MA: Kluwer Academic Publishers.
16. Snodgrass, R. (1987). The Temporal Query Language TQuel. *ACM Trans. on Database Systems*, 12(2), 247-298.
17. Williamson, C. and Shneiderman, B. (1992). The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. *SIGIR'92 Proceedings*. NY: ACM Press.