DESIGN AND USER TESTING OF A MULTI-PARADIGM QUERY INTERFACE TO AN OBJECT-ORIENTED DATABASE

Dac Khoa Doan, Norman W. Paton and Alistair Kilgour
Department of Computing and Electrical Engineering
Heriot-Watt University
Riccarton, Edinburgh EH14 4AS, Scotland, UK
e-mail: <vibama,norm,ack>@cee.hw.ac.uk
phone: +44-31-449-5111 ; fax: +44-31-451-3431

Abstract: This paper reports on experience obtained during the design, implementation and use of a multi-paradigm query interface to an object-oriented database. The specific system which has been developed allows equivalent data retrieval tasks to be expressed using textual, form-based and graph-based notations, and supports automatic translation of queries between these three paradigms. The motivation behind the development of such an interface is presented, as is the software architecture which supports the multi-paradigm functionality. Feedback from initial user trials with a dual-paradigm version of the system indicates that users can use it to perform complex query tasks without difficulty, that given the choice users overwhelmingly prefer the graph- based to the text-based interaction style, and that graphical visualisation of textual queries appears to aid users in query construction.

Keywords: query interface, multiparadigm, object-oriented database, user testing.

1 Introduction

The application of object-oriented techniques to the area of database management has been extensively researched, and a range of commercial systems have been developed which are used to support a variety of tasks relating to data intensive applications. This level of activity, however, has not been matched by comparable progress with query interfaces, and in many commercial systems query language support is limited to a subset of the facilities offered by SQL [9]. Such limitations are particularly unfortunate because the rich schema structures of object-oriented systems present opportunities for effective visualisation of data model constructs [1].

The work presented in this paper seeks to overcome some of the weaknesses in commercial query interfaces to object-oriented databases (OODBs) through the development of novel graphical query interfaces, and by exploiting the potential for automatic translation between alternative paradigms. The decision to pursue a multi-paradigm approach was motivated by a range of observations: while there is an emerging standard textual query language for OODBs based upon SQL, there are alternative query languages described in the literature, and no such consensus exists with respect to graphical interfaces; different proposals have been made for direct manipulation interfaces to OODBs, but there is little empirical evidence as to the effectiveness of alternative approaches; experience from other areas suggests that different users or tasks may be best served by different interaction styles, and that support for alternative paradigms may make a system accessible to a wider range of users; textual languages are often more powerful than their graphical counterparts, and there is the hypothesis that users can learn about a textual language as a side-effect of using a graphical interface.

This paper is organised as follows. Section 2 outlines the principal design issues addressed when developing the multi-paradigm interface, and presents its overall architecture; section 3 describes the experiments which have been conducted to evaluate the system with users; section 4 summarises the principal results of the user trials; section 5 indicates how this work relates to other research on the design and

use of database interfaces; section 6 presents some conclusions and areas for future research.

2 Interface Design

This section describes the design of a multi-paradigm query interface [7] for the Object-Oriented Database ADAM [11], in which three different query paradigms including the textual (Daplex) [12], form-based and graph-based have been implemented. Each query interface is a separate component, but the option is provided to translate between any of the above query paradigms. The architecture of the multi-paradigm query interface is presented in figure 1. All of the interfaces use the same object-oriented structures to describe queries internally. These structures are constructed incrementally in response to user operations, and can be mapped into the internal form of the optimiser, which is used to plan efficient evaluation strategies. The object-oriented internal form is defined as a collection of ADAM objects associated with methods to support the automatic translation of queries between different paradigms.

The visual representations required by the three existing paradigms are very different – the textual (Daplex) interface requires an editor window into which queries can be typed; the form-based interface requires a class browser where all the classes of the database to be queried are displayed; the graph-based interface is based on two picture windows, one depicting the database schema graph and the other the query graph. All the interfaces require an operation panel which consists of a number of buttons providing facilities for constructing, editing and executing the query, or for translating the current query into other query paradigms.

The visual representations of queries in the textual, form-based and graph-based interfaces are distinctively different– the textual query is simply a string; in the form-based interface, each class of interest to a query is represented as a form; in the graph-based interface, each class or scalar attribute of interest to a query is represented as a box, and subsequently, the boxes are linked to each other to form the query graph. The inter-paradigm query translation facility enables a query to be constructed in the multi-paradigm query context. For example, users can start to build a query using the graph-based query interface, then translate it into the textual language so that the query can be completed by editing the translated query in the textual query window.

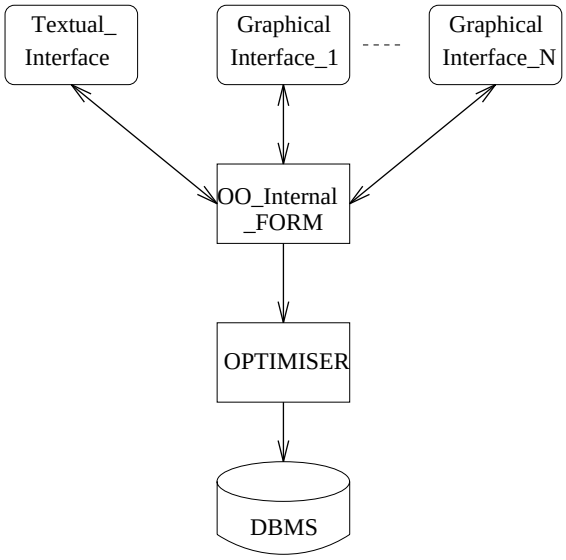

Figure 1: Overall architecture of the multi-paradigm query interface.

3 Experiment Design

3.1 Experimental interface and database

The interface used in the experiment was a dual paradigm query interface which consists of the textual and graph-based query interfaces displayed in a single window as illustrated in figure 2.

In the dual-paradigm query interface, graph queries are formulated by direct manipulation using the popup menus attached to the boxes in the query graph window which are created as a result of clicking on boxes in the schema graph window. Textual queries are entered by typing in the textual query window. The basic structure of a textual query involves iteration over classes of objects, and the application of functions to those objects to retrieve scalar or object attribute values. Automatic translation between the graphical and textual queries can be invoked by clicking on the corresponding translation buttons (i.e –> *Text* or –> *Graph*) in the dual-paradigm query interface.

In this experiment, queries for use in training and evaluation were drawn from a *graduate student ad-*

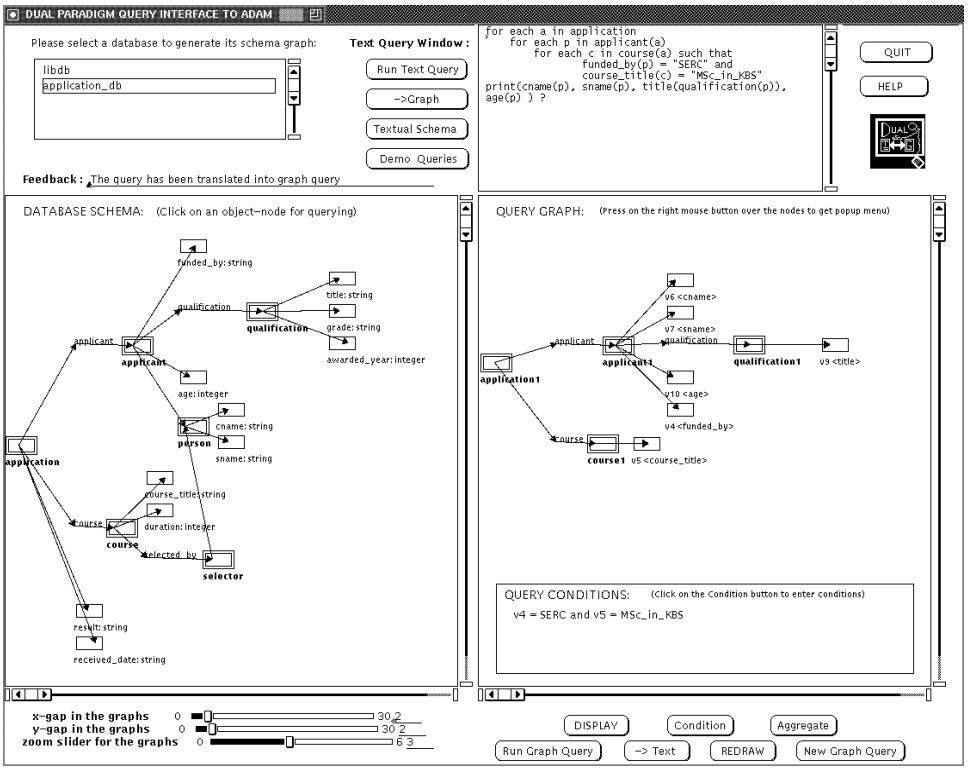

Figure 2: Dual paradigm query interface to the OODB ADAM.

mission database which concerns applications for admission to postgraduate study at the University.

3.2 Experiment

The experimental procedure is illustrated in figure 3.

Subjects: Subjects were recruited from among the Computer Science postgraduate students at Heriot-Watt, and thus all had significant computing experience. A total of 12 subjects participated in the evaluation, of which two were chosen for videoing. The subjects are assumed to have some familiarity with the procedure for admitting students to postgraduate degrees, the information which is stored in the example database.

Tasks : There are four queries presented to the subjects in the training session. A set of instructions on how to perform each of the training queries was provided to the subjects. In the evaluation session, subjects were given four additional queries to complete without external assistance.

Training session: Subjects were given the opportunity to learn the facilities supported by the interface to be evaluated, and to enhance their understanding of the database schema being used. During the training session, the subjects were free to ask for guidance on using the system, and were encouraged to think aloud, giving comments and suggestions on the usability of the interface. The training session was designed to ensure that subjects had similar levels of exposure to each of the two paradigms.

Evaluation session: During this session, the subjects' activities were carefully observed and recorded, and a logfile kept to help with identification of recurring problems when using the interface. The *completion time* and the *number of errors* encountered for each query task were measured to yield quantifiable results. At the end of the evaluation session, subjects were asked to put down their comments and suggestions for improving the evaluated interface, and to grade the *intuitiveness, suitability* and *effectiveness* of the query interface.

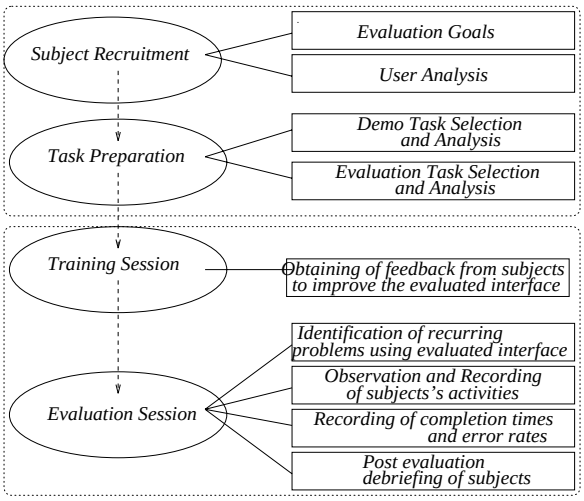

Figure 3: The evaluation process.

4 User Feedback

4.1 User Behaviour

Preference for the graph-based paradigm : There is evidence of an overwhelming preference for the graph-based query interface in the evaluation session. Subjects used only the graph-based query interface to perform their tasks; none of them used the textual query interface even though they received similar training in both paradigms. It was noticeable that it took subjects more time to gain comparable levels of familiarity with the textual paradigm than the graph-based paradigm. Subjects also appeared to be more at ease with the graph-based paradigm in the training session.

Preference for the graphical translation : During the training session, only 6 out 12 subjects opted to view the textual translation when using the graph-based paradigm, whereas 10 out of 12 subjects chose to see the equivalent query graphs when using the textual paradigm. It is interesting that subjects indicated that they only wanted to see the textual translation *out of curiosity*, and they expressed *minimal interest in learning the syntax of the textual language* in the absence of any well-defined reasons to do so. On the other hand, subjects using the textual paradigm indicated that they wanted to see the graphical visualisation of the textual query in order to confirm and enhance their understanding of the query.

Tendency to start querying from destination class : In OODBs, relationships are often described in a directional manner, and thus depicted in a graph by an arrow from one class to another. Users tended to develop queries by navigating from the class associated with the answer to the query, even though this may involve going 'backwards' along many relationship links. It is clear that interfaces must support straightforward navigation through relationships in both directions.

Tendency to exploit query reuse : About half of subjects built a new query by editing the previous one. This emphasizes the potential of a query history mechanism which allows past queries to be saved or recalled.

Tendency towards schema querying : Subjects tended to construct a query graph by focusing upon and selecting items in the schema graph rather than the query graph.

4.2 User Expectations

Subjects suggested that the following be added to the interface used in the experiment:

- A clearer boundary between the textual and graphical paradigms.
- An on-line hypertext help system.
- User control over the initial amount of information provided.
- User control over the formatting of query answers.
- The use of different colors to represent different data types.

4.3 User Satisfaction

In general, subjects satisfactorily used the interface and successfully performed the query tasks in the evaluation session. An error was recorded when subjects did not formulate the query correctly (type 1), or became stuck and asked for assistance (type 2), or failed to complete query after 15 mins (type 3). There were only 2 errors of type 2 recorded during the evaluation session with 12 subjects over four query tasks. User preference scores for the *intuitiveness, suitability* and *effectiveness* of the interface were graded on a scale from 0 to 5, and averaged 4.0, 4.3 and 4.5 respectively. These quantitative results have shown the overall success of the experimental

system in meeting users' requirements and expectations. The extent to which this success derives from the dual-paradigm nature of the interface is, however, less clear.

5 Related Work

This section discusses to related work on single or multi paradigm query interfaces to OODBs, and on database interface evaluation. The review of single paradigm query interfaces to OODBs in this section covers only textual and graph-based query interfaces, which are the paradigms used in the experimental interface.

The textual query paradigm is the most widely reported approach to querying OODBs. Most of the textual query languages are extensions of SQL [2]. The O_2 query language [6], like that of the textual query language in the experimental interface, has a functional flavour. Typical graph-based query interfaces for OODBs are [8, 5, 10], in which attributes of a class in the database to be queried are displayed along the edges of graph, and query graphs are constructed by direct manipulation of the schema graph.

In [4], a framework for a multiparadigmatic visual query environment is proposed which supports form-based, diagrammatic and iconic paradigms, and which also allows switching between different paradigms during the querying process. However, no associated working prototype has been described. A multimodal querying approach for an Object-Oriented Database is presented in [14], in which queries can be independently expressed either in graphic, iconic or textual paradigms, but translation between different query paradigms is not supported.

A number of authors have conducted practical evaluations of query interfaces for relational databases. For example, [3] reports on an experiment in which the textual language isql was compared with the form-based language Simplify and the natural language interface Data Talker in the context of the relational model. The evaluation showed that no interface was best overall. An evaluation of an application-specific *dynamic query interface* is presented in [13], where it was shown that the application-specific graphical interface was able to consistently out-perform general purpose query interfaces. To the best of our knowledge, there are no existing publications which present evaluations of multi-paradigm query interfaces to OODBs.

6 Conclusions and Future Work

The first outcome of the user testing was to confirm the power and elegance of a multiparadigm interface system supporting automatic translation between a query expressed in one form and any other supported form. The guarantee of consistency between the different forms, whichever was the initial one chosen, was an essential prerequisite for any meaningful investigation of user preferences and behaviour. However, in the event user preferences were such that the power of translation was seldom exercised in practice, except during the training sessions. Given a free choice users showed an overwhelming preference for the graph-based interface. Further, this preference was backed up by performance, in that users were almost completely successful in carrying out the prescribed queries with the method of their choice.

Secondly, when pressured into working with the textual representation, users often made use of the ability to translate the query into a graph-based equivalent as a means of checking out their understanding of the query. The availability of the graph-based visualisation appeared to add confidence and insight, but these impressions need to be tested more thoroughly in a context where users use the textual language as their primary vehicle (either because of its power, sole availability on a particular system, or because they are required to learn it). A comparison between a group using only the textual language, and another where the graph based visualisation was available, but no execution or editing of the graph-based query was allowed, could throw light on the extent to which the graph-based representation aided comprehension, or supported correct expression of the required query.

Once users had seen and used both of the available paradigms, it was difficult to persuade them to use the text-based formalism for any purpose at all. In the "text versus graphics" debate, these results reinforce the view that users new to the system (though not necessarily new to the domain) show a dominant preference for the graphical method, and perform extremely well with it. However, further work is needed to identify the extent to which this conclusion might be modified by experience with the system, by the nature of the task, or by the background and expectations of the users.

Another important issue is query reuse. The present system does not support a history mechanism; only

the most recent text-based and the most recent graph-based query are available when the user starts work on a new query (and the two extant query relics may or may not be equivalent). Given the addition of an effective mechanism for saving and browsing past queries, it would be of great interest to discover whether a preference might appear for editing a previous textual query in cases where only simple changes, e.g. to attribute values, were required to obtain the new query. It would be important to investigate this in the context of real tasks, rather than artificial test situations where the choice and ordering of the prescribed queries could bias the results.

A possible hypothesis concerning such a multi-paradigm system is that given the automatic generation of the textual equivalent of commands or queries generated by some graphical means, users might gradually absorb the structure and conventions of the textual language by a process akin to osmosis. The results of the present investigation do not look promising for this hypothesis, in that as far as we could tell users totally ignored the textual representation whenever possible. However, further tests would be needed to ascertain whether, in spite of themselves, users had nevertheless acquired some knowledge, or even understanding, of the textual query language which they would not have gained in a single paradigm environment.

Acknowledgements: The first author is supported by a studentship from the UK Engineering and Physical Science Research Council (EPSRC). We are greatly indebted to the students who gave up time to participate in the experiment.

References

[1] G. Al-Qaimari, N. W. Paton, and A.C. Kilgour. Visualising Advanced Data Modelling Constructs. *"Information and Software Technology"*, 36(10):265–281, 1994.

[2] D. Beech. A foundation for evolution from relational to object databases. In J.W Schmidt et al, editor, *Proc. EDBT*, pages 251–270. Springer-Verlag, 1988.

[3] J. E. Bell and L. A. Rowe. An Exploratory Study of Ad Hoc Query Language to Databases. In *The 1992 Proc. of Data Engineering Conf*, pages 606–613. IEEE Press, 1992.

[4] T. Catarci, S.K Chang, and G. Santucci. Query Representation and Management in a Multiparadigmatic Visual Query Environment. *Journal of Intelligent Information Systems, (3)*, pages 299–330, 1994.

[5] M. P. Consens, I. F. Cruz, and A. O. Mendelzon. Visualizing queries and querying visualization. In *SIGMOD RECORD*, pages 39–46, March 1992.

[6] O. Deux and et al. The Story of O_2. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), March 1990.

[7] D. K. Doan, N. W. Paton, A. C. Kilgour, and G. Al-Qaimari. A Multi-Paradigm Query Interface To An Object-Oriented Database. In *to appear in Interacting with Computers*, 1994.

[8] M. Gemis, J. Paredaens, and I. Thyssens. A Visual Database Management Interface Based on GOOD. In *The 1st International Workshop on Interfaces to Database Systems, Glasgow*, pages 25–31. Springer-Verlag, 1993. R. Cooper (Ed).

[9] U. Hohenstein, R. Lauffer, and P. Weikert. Object-Oriented Database Systems: How Much SQL do they Understand? In D. Karagiannis, editor, *Proc. 5th Int. Conf. on Databases and Expert Systems Applications (DEXA)*, pages 15–26. Springer-Verlag, 1994.

[10] A. Papantonakis and P. King. Gql, a declarative graphical query language based on the functional data model. In *Proc. of the 2nd International Workshop on Advanced Visual Interfaces*, 1994.

[11] N. W. Paton. ADAM: An Object-Oriented Database System Implemented in Prolog. In M. H. Williams, editor, *Proceedings of the Seventh British National Conference on Databases (BNCOD 7)*, pages 147–161, Edinburgh, July 1989. Cambridge University Press.

[12] D. W. Shipman. The Functional Data Model and the Data Language DAPLEX. *ACM Transactions on Database Systems*, 6(1):140–173, 1981. Also in [15], pages 95-111.

[13] B. Shneiderman. *Designing The User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, second edition, 1992.

[14] S. Tallard. A multimodal approach for an Object-Oriented Database Query. In *Basque International Workshop on IT (BIWIT)*, pages 209–222. CEPADUES Press, 1994. C. Chrisment (ed).

[15] S. B. Zdonik and D. Maier, editors. *Readings in Object-Oriented Database Systems.* Morgan Kaufmann, San Mateo, CA, 1990. ISSN 1046-1698, ISBN 1-55860-000-0.