Visualising Data Modelling Constructs in an Object-Oriented Database

Ghassan al-Qaimari

Department of Computer Science
Royal Melbourne Institute of Technology
Melbourne 3001, Australia
e-mail: <ghassan>@cs.rmit.edu.au

Alistair C. Kilgour and **Norman W. Paton**

Department of Computing and Electrical Engineering
Heriot-Watt University
Riccarton, Edinburgh EH14 4AS, Scotland, UK
e-mail: <ack,norm>@cee.hw.ac.uk

Abstract

Object-oriented databases are seen as potential successors to relational databases, at least in part because they provide a richer set of data modelling constructs. This paper addresses the challenge to interface designers posed by such constructs to support data browsing and modelling through powerful and perspicuous visualisations, with the additional requirement that the visualisations should be readily updatable when the semantic model is modified, as is possible in some extensible object-oriented databases.

The paper describes a range of visualisations for the semantic modelling constructs of the extensible object-oriented database ADAM, and reports the results of a series of empirical evaluations to assess the effectiveness of these visualisations. These results support the belief that it is possible to make the advanced data modelling constructs of an extensible object-oriented database system accessible to non-specialist users, allowing the advantages of these systems to accrue across a much wider range of application domains than with conventional systems.

Keywords: Visualisation, lucidity, object-oriented databases, user interfaces, data modelling constructs, evaluation.

1 Introduction

A characteristic feature of the relational model of data which has significantly contributed to its widespread acceptance is its support for declarative query interfaces and form-based retrieval/manipulation systems. The design of such systems has been eased by the straightforward data structuring mechanisms supported by the relational model which can be readily and directly depicted in interfaces as tables or forms. The simplicity of the relational model, while a strength for certain tasks, is also one of its principal weaknesses – representing the structural semantics of complex applications is cumbersome using first normal form relations, and the relational model has come to be regarded as inappropriate for a range of data management tasks.

Semantic data models and object-oriented databases (OODBs) have been proposed, in part, to overcome the limited modelling facilities of the relational model. However, such models are necessarily more complex than their relational predecessors, and thus are less readily associated with visual representations which are suitable for the wide range of tasks (data entry, querying, browsing, schema design, etc) associated with a database system. This problem is further exacerbated in some recent database systems which support such modelling constructs as relationship objects, versions and composite objects, and in systems which can be extended to support additional data modelling constructs.

This paper describes experience designing and evaluating visualisations for a range of modelling constructs supported by the OODB ADAM [GKP92]. The modelling facilities provided by ADAM are described in [DG91, PDB93]. The paper is organised as follows: section 2 describes some related work on interfaces to OODBs; section 3 considers some general issues relating to visualisation, and how they are relevant to interfaces to OODBs; section 4 outlines our experience of designing and evaluating modelling constructs, and compares the evaluation techniques used; and section 5 presents some conclusions.

2 Interfaces to object-oriented databases

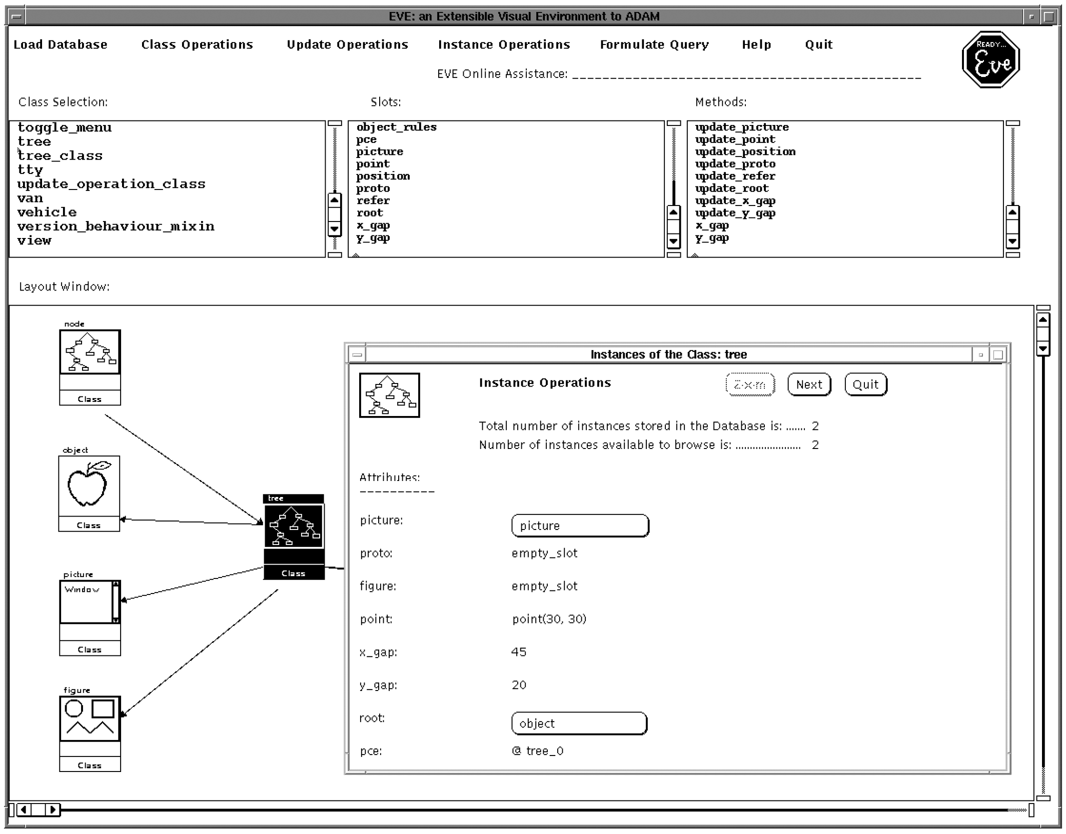
As with other database systems, much of the work to date on interface design for OODBs has concentrated on support for the information retrieval task, through browsers and query interfaces. Browsers for object-oriented databases have often been based on those commonly found in object-oriented programming environments, the classic original of which is the Smalltalk browser. However, the needs of the data modeller using an object-oriented database are not necessarily the same as those of a programmer using an object-oriented language, and recent studies have shown that conventional object browsers are not very effective in supporting the reuse of objects, which is an important part of the programmers task in an object-oriented development environment [Gea92].

The following are among the main commercial and prototype OODB interfaces: ISIS [Gea85], FaceKit [KN89], SKI [Kin86], SIG [MNG90], GLAD [Wu90], SNAP [BH90], Easy-Objects [AA91], O_2LOOKS [Dea90], Iris [Vea88], and GOOD [GPT92]. Among these there is little agreement as to what constitutes a visually effective interface to an OODB. Even classes, attributes and relationships, the facilities supported by all of the systems, are represented in different ways on the screen, including networks, icons, and frames. Perhaps the disagreement within the database community regarding the central characteristics of an OODB has reflected on the design of their interfaces.

The graphical interfaces found in these systems can be divided into two categories: *form-based* interfaces, such as OOPS and SIG, and *graph-based* interfaces, such as ISIS and SKI. Graph-based systems use linked visualisations to target information for retrieval. Other data manipulation operations, for the creation, modification or deletion of data, have been ignored in many graph-based interfaces. Graphs are normally used to model data types, while data manipulation operations range over instances of data types.

Form-based interfaces can be used both for data retrieval and for data manipulation. Problems for form-based interfaces include the representation of hierarchical data structures, and the description of distinct but related concepts. Forms can be used to support the processing of data at the attribute level rather than at the entity type or class level, and nested structures are necessary for displaying hierarchical data relationships.

The most effective graphical interfaces are those which allow the user to interact directly with visual objects in ways that are suggestive of the underlying functionality being provided. Finding representations which effectively achieve this for a broad range of database



Total number of instances stored in the Database is: 2
Number of instances available to browse is: 2

Attributes:
----

picture:	picture
proto:	empty_slot
figure:	empty_slot
point:	point(30, 30)
x_gap:	45
y_gap:	20
root:	object
pce:	@ tree_0

Figure 1: Layout of EVE base window plus a form used for browsing instances.

tasks is a major challenge to the interface designer, one to which the work reported in this paper is principally addressed.

Some systems lack uniformity in the way that common database tasks are expressed. While some tasks can be expressed using direct manipulation techniques, other tasks, say queries, require form filling or even the use of a programming language. GOOD is one of the few systems that offers a uniform methodology throughout the different tasks it supports.

In general, database interfaces differ in three fundamental ways [Eps91]: with respect to the visual metaphors which they employ; with respect to their expressive power; and with respect to their underlying computational mechanisms. It is important for designers to take into account users' preferences and assumptions from the early stages of building an interface. Nevertheless, however skilled the interface designer, and however extensive the preliminary task and user analysis, it is now widely recognised that it is never possible to get the interface right first time. Iterative evaluation and redesign are therefore vital in arriving at a successful and supportive interface, although there are few references in the literature describing current interfaces to OODBs to the importance of evaluation in the design process.

In addition, few current systems support recent ideas in semantic data modelling, such as composite objects and versions. As these new modelling constructs become more widely available, it will become increasingly important to find appropriate visualisations as a basis for effective direct-manipulation interfaces to the full range of modelling constructs. Later sections of the paper describe initial steps towards the development of effective visualisations for recent semantic modelling constructs.

The visualisations presented in this paper are supported by the EVE (Extensible Visual Environment) direct-manipulation interface [PaK92] which is part of a project that aims at addressing the two major weaknesses of existing database interfaces by: developing effective tools for the implementation of tailorable database interfaces, and developing effective visualisations of sophisticated data modelling constructs. To overcome the first of the two weaknesses our approach has been to implement a system which integrates an object-oriented graphical component set with the OODB ADAM. The aim of the integration was to store the interface objects in the database, where they can have the same structure as normal database objects, and therefore, to eliminate the impedance mismatch between the interface and the database [PCE+94].

The ADAM graphical toolkit, known as EDEN [PaD94], can be viewed as an object-oriented widget set that consists of normal database objects, which can be subjected to standard ADAM operations (such as *create, replace, delete*, etc), but which also have behaviour which has visible consequences, (such as *draw, open, display*, etc).

A principal feature of the underlying ADAM data model is that it can be extended with new constructs such as relationships, versions and composite objects [DG91, PDB93]. Such an extensible data model requires an extensible direct manipulation interface in order to provide effective visualisations for the different constructs introduced to the data model. This has been achieved by using ADAM to build its own extensible interface, known as EVE, with the EDEN object-oriented widget set [PaD94].

3 Visualisation of data modelling constructs

3.1 Properties of visualisations

Visualising information, especially complex and intricate information, has been the subject of considerable research, but little of this has related to the external representation of complex data models, and in particular modern object-oriented database systems.

The fundamental hypothesis underlying the work reported here is that the usability of object-oriented semantic data modelling constructs can be enhanced by lucid and expressive visual screen representations. Effective visual representations are of particular importance where modelling constructs are intended to capture both the structural and the behavioural semantics of real-world concepts. The challenge for the interface designer is to find clear, concise and comprehensive representations which achieve in practice the usability gains over textual representations which our hypothesis suggests are possible.

Designing visual representations of data models involves a kind of *projection,* where the knowledge provided by the data model is transformed (mapped) into recognisable and expressive pictorial representations. This process has parallels with knowledge elicitation, as used by knowledge engineers in building knowledge-based systems [McG92]. Knowledge elicitation techniques described in the literature include: document reviews and content analysis, observation, interviews, concept and vocabulary analysis, and job and task analysis. Several of these approaches were applied to provide a starting-point in the iterative search for effective representations of semantic modelling constructs, as described in section 6. Other factors informing the design process included an analysis of earlier theoretical and empirical investigations of factors influencing usability. In [Eas84], usability is defined as "the extent to which the user can exploit the potential utility of a system". Other aspects of usability which have been proposed include *functionality* (how well a system fits the needs of a set of particular tasks [Ben84]), and *acceptability* (how willing users are to use the system in their own organisational context [Ric87]). In addition to these, *productivity increase,* which relates to efficiency in terms of time, money and quality assurance, is of particular concern in commercial environments .

In [Nor88], the term *visibility* is used to indicate "the mapping between intended actions and actual operations". It is also suggested that there are occasions when too much visibility can be a problem: "It is an excess of visibility that makes gadget-ridden, feature-laden modern audio sets and VCRs so intimidating". The author also distinguishes between *affordance,* referring to whether the design of an object suggests (that is, affords) its functionality, and *perceived affordance,* referring to what a person thinks can be done with that object.

According to [Gil91], visibility should be thought of in terms of three separate dimensions, two of which are static properties of the presentation, while the third is dynamic. The first of the static components, *accessibility,* is associated with availability of information, while the other, *salience,* is associated with its meaning. Accessibility and salience are both properties of the display alone, independent of its use. The third dimension, namely *congruence,* is a property of the display in interaction with the user, and reflects whether the salience of a display is relevant to the user's task.

In a complementary analysis [JDMM91], three distinct components of usability are investigated, which account for how the performance of a system changes with learning. These are: *guessability, learnability,* and *experienced user performance.* Guessability is defined as "a measure of time and effort required to get going with a system. The less time and effort required the higher the guessability."

Through analysis of this work and its relevance to the derivation of visual representations for semantic data modelling constructs in object-oriented databases, we have been led to propose an additional usability dimension, referred to as *lucidity,* defined as the extent to which a representation reflects and reveals the structure and meaning of the underlying modelling construct. This concept incorporates aspects of *visibility* and *guessability* as defined earlier, but we believe it more directly characterises the effectiveness of a visualisation of a modelling construct for use in a database interface, which it is the aim of our work to investigate.

Following from this analysis, sample visualisations were designs aimed at maximising lucidity, and their success in achieving this goal was evaluated empirically. The aim of the evaluation was to measure whether the characteristics of a data model were visible to the user, and whether it was easy for the user to guess the structural semantics of the data model, and the function it was intended to perform. Lucidity in the context of visualisation of modelling constructs may be considered to have the following aspects: clarity, referring to the number and organisation of visual components; accessibility, referring to the ease of which information can be accessed; perspicuousness of visual representations, which indicates if pictorial information can convey the appropriate meaning, and can be easily understood without confusion; and transparency, meaning the ability to see through the pictorial representation to the underlying semantics. Clarity and accessibility are directly related to the speed of performance, that is, the ability of the user to find a particular piece of information, or identify a particular visual clue or instruction that helps navigating through the system, within an acceptable period of time. On the other hand, perspicuousness and transparency directly affect user's goals and his ability to perform the correct task.

The following section describes in detail the initial design of possible visual representations of composite objects, one of the more powerful data modelling extensions supported by the ADAM OODB.

3.2 Visualisations of composite objects

A number of objects related by the *subpart* relationship are collectively called a *composite object* [KBG89]. Each composite object has one or more subparts, each of which may consist of other composite objects, or standard ADAM objects. The semantics of the part-of relationship are discussed in [PDB93, Kim93].

In seeking lucid representations of composite object constructs, one approach which seemed promising was to investigate the application of alternative visual paradigms. Candidate paradigms which immediately suggested themselves were form-based and graph-based. Accordingly, three alternative representations of composite objects were designed. The form-based representation (figures 2 and 3) is very similar to EVE's conventional way of visualising class instances (figure 1), with the exception that two different types of attribute are distinguished, namely composite (subparts) and non-composite.

In figures 4 and 5, the subpart hierarchy is depicted as a tree in a separate window. Note that the slot *subpart* represents a special type of relationship between a composite object class and other classes. Thus the subparts window, shown in figure 4, could be eliminated if (for example) we chose to use a different line style (different from the lines shown in the layout window, figure 1) to represent such system-defined relationships. This would enable the user to differentiate between user-defined relationships and the subpart relationship. The justification for designing a subpart window is similar to the justification for having separate is-a hierarchy window, namely to highlight the special semantics of the relationship, and to prevent cluttering of the main layout window.

The representation shown in figures 6 and 7, has all the characteristics of the form-based representation except for the subparts, which are shown as icons under the heading *Composite Attributes* (Subparts). As in the graphical representation, the user can click on any of the icons in the form to browse the subpart instances.

4 Evaluation

The goal of arriving at lucid representations of advanced semantic modelling constructs cannot be achieved just by following design guidelines, such as those in [MS86] and [Bro88]. An iterative approach must be adopted involving formative evaluation in each iteration [HH93]. Two evaluation cycles have been performed. Firstly, a pre-implementation paper-based evaluation was carried out with the help of experienced users in the fields of databases and interface design. Secondly, a practical (task-oriented) evaluation was conducted using prototype implementations based on the results of the first stage of the evaluation. The particular technique used was *cooperative evaluation* [MWHD93], which brings together designers and users in a cooperative context, so as to involve the users in the design process by giving feedback and identifying weak points at each stage.

The different backgrounds, expertise and experience of the representative users can provide valuable insights when the results of the evaluation are analysed and interpreted. Our subject profiles, in the first stage of the evaluation (25 users), indicate that all of our representative users were either lecturers, researchers, or graduate students, of whom half had considerable knowledge about databases in general, and the other half were knowledgeable about interface design. Expert users can provide critical facts and heuristics at the early stage of the design. However, we appreciate that databases are not exclusively designed for database or interface experts, and so subject profiles in the second stage of the evaluation (25 users) included, in addition to expert users, novice and intermediate users, who are likely to be the primary users of the system. Such users can have different needs, expectations and understanding of terminology. They also tend to draw different conclusions from instructions provided by the interface [McG92].

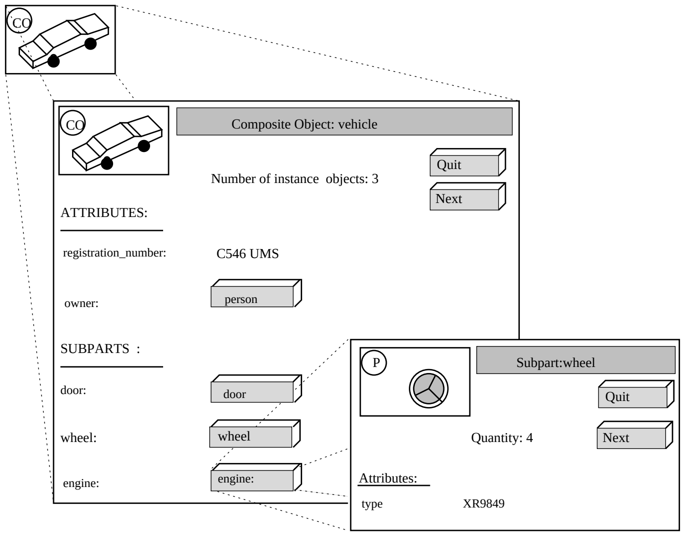
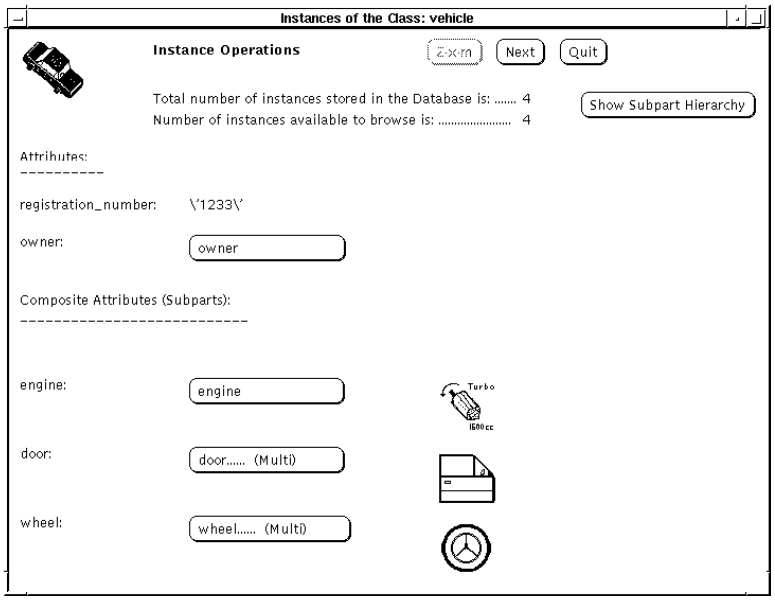


Figure 2: Paper design 1: a proposed form-based visual representation of composite objects.



Instances of the Class: vehicle

Instance Operations

Total number of instances stored in the Database is: 4
Number of instances available to browse is: 4

Attributes:
----

registration_number: \'1233\'

owner: owner

Composite Attributes (Subparts):
----

engine: engine

door: door...... (Multi)

wheel: wheel...... (Multi)

Figure 3: Visualisation 1: an implementation of paper design 1, shown above.

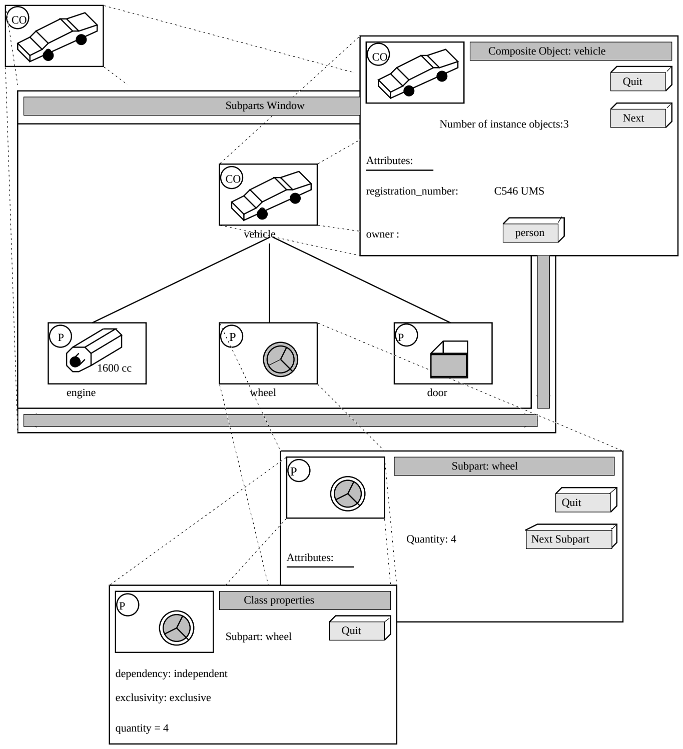


Figure 4: Paper design 2: a proposed graph-based visual representation of composite objects.

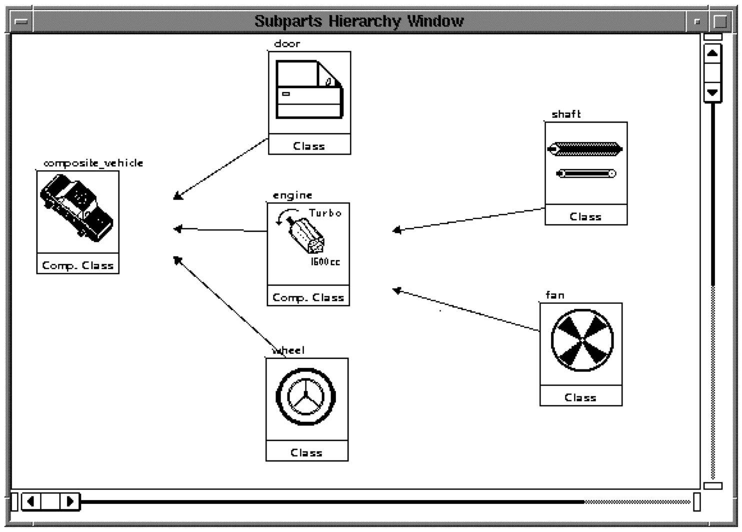
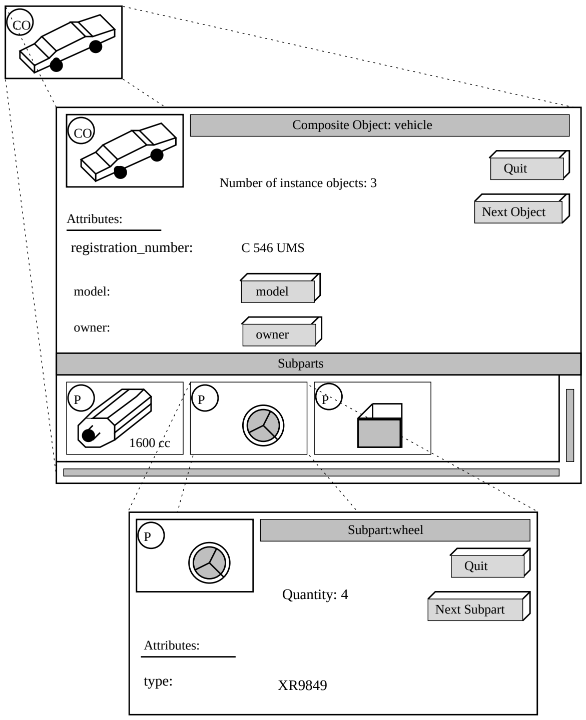


Figure 5: Visualisation 2: an implementation of paper design 2, shown above.



Number of instance objects: 3

Attributes:

registration_number: C 546 UMS

model:

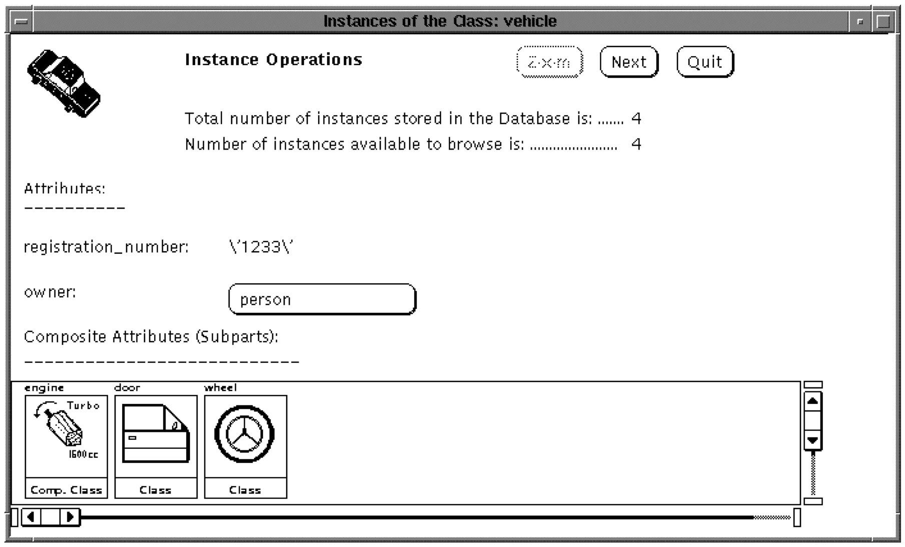
owner:

Quantity: 4

Attributes:

type: XR9849

Figure 6: Paper design 3: a proposed mixed-mode visual representation of composite objects.



Total number of instances stored in the Database is: 4
Number of instances available to browse is: 4

Attributes:
----

registration_number: \'1233\'

owner:

Composite Attributes (Subparts):

Figure 7: Visualisation 3: an implementation of paper design 3, shown above.

For each checklist question, please tick the column which best describes your answer to the question. Then write any comment which you feel you could make when answering a checklist question, in the column *Comments.*

Questions Related To Common Features.

Question	Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree	Comment
1. The concept of two levels of "Next" button, in one class representation, is confusing to me.						
2. Having two levels of counters in one form is confusing to me.						
3. The link between the top left-hand corner icons and the class level properties is a good idea (see figure 4).						
4. I prefer the class level properties to be shown in a popup menu style.						
5. I prefer the class level properties to be shown in a dialog box form, as in figure 4.						
6. I find using icons to represent class-specific visualisation, as well as construct-specific visualisation to be a good idea.						
7. I find using icons for quick identification, whereby an icon is positioned at the top left-hand corner of every dialog box to be a good idea.						
8. For browsing class instances, I like the idea of giving the system the choice between displaying a graph based or a form based representation based on size of the counter.						
9. Designing the "Quit" buttons to cause recursive deletion is a good idea.						

Are there any comments (good or bad) you wish to add regarding the above issues?

Figure 8: Sample Checklist, presented to the representative users regarding the features common to all visualisations.

4.1 Pre-implementation evaluation: (stage one)

The first stage of evaluation involved *checklists,* questionnaires answered by potential users (see sample checklist, figure 8), which were used to obtain feedback on paper mock-ups of visualisations (figures 2, 4 and 6). Such an evaluation technique offers a flexible strategy, whereby checklists can be modified according to the nature of the evaluation, for the purpose of reaching practical quantifiable evaluation results. This evaluation strategy followed a similar approach to that presented in [RJ89, Shn92].

Following consistent guidelines in the early stages of interface design ensures a consistent *look and feel* of the system as far as fonts, placement of menus, and wording of titles and messages are concerned. In addition to the design guidelines, the pre-implementation evaluation aimed to explore the extent to which the initial alternative visualisations succeed in capturing the structure and meaning of the data modelling constructs. This was achieved by encouraging the experienced representative users to explore all initial designs of visualisations, to give constructive critiques, and to suggest possible alternatives.

The first stage of the evaluation helps the designer to avoid, as much as possible, prototyping poor visual representations that might require many revisions and modifications later at the prototyping phase. However, some ideas might seem very appealing when evaluated on paper, and yet due to unexpected limitations in the implementation environment, either cannot be implemented effectively, or turn out to be less effective in practice than anticipated.

4.1.1 Results of pre-implementation evaluation

The results of the first stage of evaluation (see figure 9) indicated t hat graph-based visual representations are not necessarily the preferred way of representing modelling constructs. The representative users' comments helped highlight ambiguous designs and wording of titles. Mixed mode representations (figure 6, paper design 3) were appreciated, as was the idea of hiding class properties behind icons to avoid confusion (see *class properties* window in figure 4). They also stressed the importance of building consistent interfaces that do not change the type of display without users' instructions.

4.2 Prototype evaluation: (stage two)

The second stage evaluated prototyped implementations of the proposed visual representations. For the purpose of testing whether the results of the two evaluations correlate, and whether the two stage evaluation approach is worth the time and effort involved, not only the winning paper designs for these two modelling constructs have been implemented, but all the proposed paper designs.

The second evaluation was task-oriented - representative users were asked to perform certain specific tasks, and while looking at the information available to them on screen (that is, the three visualisations), they were asked post-experience questions the aim of which was to assess the usability of the system in general, and the lucidity of the visual representations in particular (see figure 10). Questionnaire forms were divided into different sections, each of which was based on certain criteria which are expected to be met by a well-designed user interface.

4.2.1 Designing an evaluation task

As indicated earlier, one of our aims was to measure the *lucidity* of the visual representations of the data modelling constructs supported in the EVE interface, in addition to assessing the usability of the system in general. To do so it was necessary to devise an evaluation task that reflects what is meant by lucidity and how it could be measured.

The task performed by our representative users in the second stage of the evaluation aimed at exploring the following interface issues:

- Do the visual representations succeed in suggesting to the user the meaning of the data model, and do they reveal the semantics supported by the modelling constructs?
- Is it clear to the user that a construct, such as a composite object, is a normal object in the database?
- How useful are icons for quick identification of related visual representations?
- How helpful is a hierarchical overview of special types of relationships?
- To what extent is the distinction between class level information and instance level information recognised by the users?

It was equally important after deciding on the tasks to appraise them in order to check their representativeness and the extent to which they explore the interface issues under

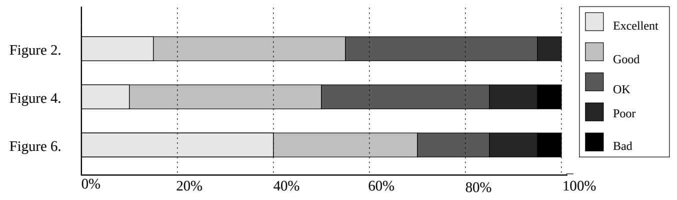


Figure 9: Relative preference scores from the three paper designs in the pre-implementation evaluation.

evaluation. To do so the proposed tasks were carried out by an experienced user in a pilot investigation.

4.2.2 Conducting evaluations

During evaluation sessions the emphasis was placed on the reason behind *how* the user found a certain answer, and *why* a certain conclusion was reached or a decision made. Also, an emphasis was placed on finding out whether the representative users performed subtasks within a reasonable period of time, which was an important factor in determining the effectiveness of a visual representation. Users were encouraged to to think aloud, in order to generate as much feedback as possible. This was achieved by asking questions after performing each subtask (such as *How did you know that?* or *What makes you think so?*, etc). For example, by asking the representative user a question like: *How many vehicles are stored in the database?*, the designer is really interested in finding out whether the *counter* in the representation is visible to the user, and whether or not the wording is confusing.

The representative users were kept unaware of the fact that the time taken to perform each subtask was being monitored. Such an informal evaluation strategy created a relaxed atmosphere, and enabled representative users to concentrate better on the screen without interruption.

While the representative user is carrying out the task, the designer (evaluator), records feedback information on a *Think aloud protocol recording sheet.* Such a recording sheet contains questions such as: What does the task user notice? What is the task user thinking now? What kind of clues is he/she looking for? What are the problems encountered at this subtask? What suggestions are being made by the task user?

After the representative users performed the tasks using the different prototypes, they were asked to rate the visualisations (figure 11) according to preference.

4.2.3 Results of prototype evaluation

The following are some of the results and the issues raised by the second evaluation.

- The representative users were almost evenly divided in their preferences between figures 3 and 7, even though in the pre-implementation evaluation figure 6 was the clear favourite. Figure 3 is considered by many to be simpler and more consistent with EVE's conventional way of browsing instances. Figure 12 compares the relative performance scores from both, paper (top) and practical (bottom), evaluations of the three visualisations.
- The idea of showing a special hierarchy window to give an overview of special types of relationship, such as the subpart relationship, was appreciated.
- The majority of the representative users like the idea of having a special button (say *Show Subparts Hierarchy*), which can be selected if the user is interested to see the

Your task is to find the *engine_number* of the *engine* of the *vehicle* which has the *registration_number: 1234.*

After carrying out the task, use the information available to you on the screen to answer to each of the following post-experience questions.

1. Who is the owner of this vehicle? How did you know?
2. How many vehicles are stored in the database?
   How did you know?
3. How many doors does the vehicle have?
   How did you know?
4. Can two vehicles share the same engine?
   Why/Why not?
5. What will happen to the vehicle if you try to remove an *owner* from the database?
   How did you know?
6. What fundamental difference is there in the way the *registration_no* and the *engine* are modelled as properties of vehicle?
   How did you know?
7. What is the fundamental difference between the way in which the *engine* of a vehicle is modelled compared with the *owner* of a vehicle?
   How did you know?
8. Is an engine a normal object in the database?
   Why do you think it is/Why do you think it is not?
9. Is it a good idea to have a special button in the dialog box (figure 3 and 7) which gives the user the choice of whether to display the *Subparts Hierarchy Window* (figure 4) or not.
   Explain why it is/isn't a good idea?

Figure 10: Post-experience Questionnaires

hierarchy window (see figure 13). This means that the hierarchy window is not forced on users who are not interested in looking at it.

- Icons in the EVE interface represent classes in the database. In EVE, each class has a *class specific visualisation* (such as the picture of a car to represent the class *vehicle*), as well as a *construct specific visualisation* (being the object type, such as *composite class*). Users preferred using words (such as *composite class*, to indicate that the class vehicle is a composite class) over the use of symbolic graphical representations, since the latter adds another level of abstraction which may or may not be easily recognised. While expert users might easily recognise a symbol, novices might find it too abstract.
- Icons played an important role in suggesting to the users that data modelling constructs are normal database objects. This is because the regular classes and the constructs are displayed in the layout window using the same style of iconic representation. However, some users were confused about the difference between an *is_a*

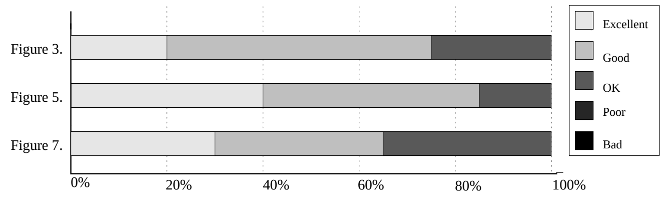


Figure 11: Relative preference scores from prototype evaluation of visualisations for composite objects.

hierarchy window and a *part_of* hierarchy window.

- The performance of our users improved significantly when they repeated the task using the second prototype of visual representations, which conforms to the results of the experimental study in [JDMM91], which describes components of usability that account for how performance with a system changes with learning. Therefore, in order to ensure better results, half of the representative users were asked to carry the evaluation task using the first prototype first (figure 3), while the other half were asked to start with the second prototype (figure 7).

- The users appreciate the presence of icons in the top left-hand corner of every dialog box for the purpose of quickly identifying dialog boxes generated for browsing instances.

- Further investigation is needed regarding the object-oriented terminology and whether it should be simplified by using a more familiar expressions at the interface level. Some users found it hard to understand concepts such as *slot, exclusivity, dependency*, etc. For example, novice and intermediate users prefer the use of the word *attribute*, rather than the word *slot*. While using these terms at the interface level matters if the terms are relevant to the task, using simpler terms can increase the lucidity of a visualisation in particular and the performance and acceptability of a system in general. However, designers must be careful when using different or simplified terminologies because they can confuse experienced users.

- When users were asked questions such as *Can two vehicles share the same engine?* or *What will happen to the vehicle if we delete the engine?*, the first reaction of many was to try to give logical answers. Some users tried to delete the engine to see the results of such an action, and others tried to look at the screen searching for a clue. This suggests that prior assumptions of the users matter and should be taken into account at each stage of the evaluation.

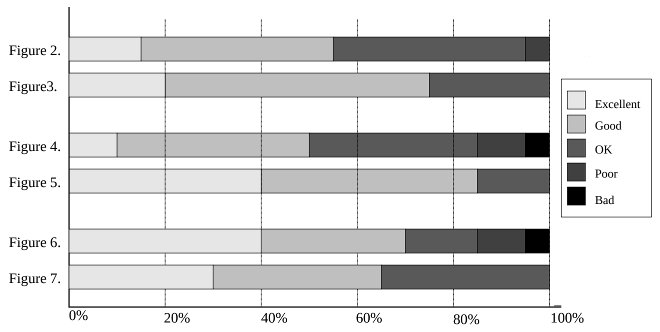


Figure 12: Relative preference scores from paper (top) and practical (bottom) evaluations of visualisations for composite objects.

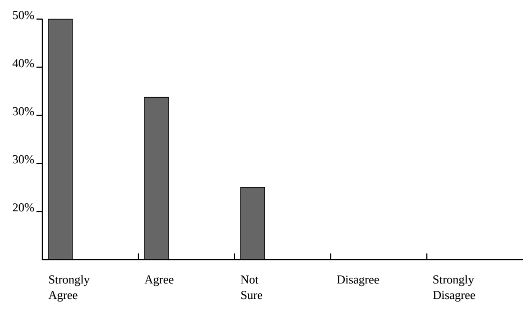


Figure 13: Do you agree with including a button in visualisation 1 for the purpose of optional display of the subpart hierarchy window (vis. 2).

- The distinction between class level information and instance level information was not always clear to some users - it is important to separate these in the display to avoid confusion. Further study is needed to determine the best way to show class level information and constraints.

4.3 Comparison of pre and post implementation evaluations

In [CHP90], the authors compare four different evaluation approaches used by software designers to evaluate the usability of prototype interfaces. These approaches are summarised as follows: *formal analysis*, paper based evaluations which are used to evaluate theoretical models, system specifications, and to measure a hypothetical expert users' performance; *empirical approaches*, which test the interface system in an experimental or semi-experimental setting such as in a usability laboratory; *contextual research*, which is ethnographic, and attempts to understand (observe) users' attitudes towards the interface in ecologically valid situation (users' normal environment); and finally, *construct elicitation*, which is a post-event elicitation of users' opinions, used to compare several prototypes.

Our pre-implementation evaluation followed the first approach for the purpose of testing initial design ideas with the help of expert users, while the second evaluation followed the fourth approach, comparing several prototyped visualisations by eliciting users' opinion after completing evaluation tasks. The second stage confirmed most of the results of the pre-implementation evaluation regarding our initial design decisions and ideas.

5 Conclusion and future directions

Designers and evaluators use different evaluation methods to achieve an objective measurement of the user/system interaction, and yet it is in fact the subjectivity of the evaluation experience of the individual users which they are often after [CHP90]. This paper attempted to exploit another dimension in interface evaluation, that is, the degree of *lucidity* of visual representations, and has described our practical experience in evaluating visual representations of semantic data modelling constructs in object-oriented databases. The need for expressive visual representations becomes important as the underlying systems support increasingly rich semantics, and we believe that cooperative evaluations play a crucial role in bringing the designers and the users together in a context that involve the users in the design (pre-implementation) phase, and in the prototyping phase.

The advantages of the object-oriented paradigm became very clear during the design and evaluation process. The evaluation of alternative visualisations required that rapid prototyping and modification of existing displays was supported by the underlying system.

In practice, a uniform object-oriented approach to the implementation of both data model extensions and interface constructs facilitated rapid revision to both the data model and its visualisations.

Acknowledgement We are grateful to Carmel Smith for helpful comments relating to pilot testing evaluation tasks, and to our colleagues who participated in the evaluations.

References

[AA91] J. Almarode and T. L. Anderson. GemStone Visual Interface Designer: A Tool for Object-Oriented Database Design. In *Object-Oriented Databases: Analysis, Design and Construction (DS-4)*, pages 73–94. North-Holland, 1991. W. Meersman et al (Eds).

[Ben84] J. L. Bennet. Managing To Meet Usability Requirements; Establishing And Meeting Software Development Goals. In *Visual Display Terminals*, pages 161–183, 1984. J. Bennet et al (Eds).

[BH90] D. Bryce and R. Hull. SNAP: A Graphics-based Schema Manager. In *Readings in Object-Oriented Database Systems*, pages 537–550. Morgan Kaufman Publishers, 1990. S. Zdonik and D. Maier.

[Bro88] C. M. Brown. *Human-Computer Interface Design Guidelines.* Ablex Publishing Co, NJ, USA, 1988.

[CHP90] J. Crellin, T. Horn, and J. Preece. Evaluating Evaluation: A Case Study Of The Use Of Novel And Conventional Evaluation Techniques In A Small Company. In *Human-Computer Interaction - INTERACT'90*, pages 329–335, (North-Holland), 1990. Elsevier Science Publishers. D. Diaper et al (Eds).

[Dea90] O. Deux and et al. The Story of O2. In *IEEE Transactions on Knowledge and Data Engineering, (2)*, pages 91–108, 1990.

[DG91] O. Diaz and P. M. D. Gray. Semantic-rich User-defined Relationship as a Main Constructor in Object-Oriented Database. In *Object-Oriented Databases: Analysis, Design and Construction DS-4*. Elsevier Science Publishers (North-Holland), 1991. R. A. Meersman and W. Kent and S Khosla (Eds).

[Eas84] K. D. Eason. Towards the Experimental Study of Usability. *Behaviour and Information Technology, 2(3)*, 1984.

[Eps91] R. G. Epstein. The TableTalk Query Language. *Journal of Visual Languages and Computing*, pages 115–141, February 1991.

[Gea85] K. J. Goldman and et al. ISIS: Interfaces for a Semantic Information System. In *ACM SIGMOD Conference on the Management of Data*, pages 328–342, 1985.

[Gea92] T.R.G. Green and et al. Towards a cognitive browser for OOPS. *International Journal of Human Computer Interaction, 4(1)*, pages 1–34, 1992.

[Gil91] D. J. Gilmore. Visibility: A Dimensional Analysis. In *People and Computers VI, Proc. of the HCI '91 Conference*, pages 317–329. Cambridge University Press, 1991. D. Diaper and N. Hammond (Eds).

[GKP92] P. M. D. Gray, K. G. Kulkarni, and N. W. Paton. *Object-Oriented Databases: A Semantic Data Model Approach.* Prentice-Hall, 1992.

[GPT92] 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 (IDS92), Glasgow*, pages 25–31. Springer-Verlag, 1992. R. Cooper (Ed).

[HH93] D. Hix and H. R. Hartson. *Developing User Interfaces.* Wiley, 1993.

[JDMM91] P. W. Jordan, S. W. Draper, K. K. MacFarlane, and S. McNulty. Guessability, Learnability, and Experienced User Performance. In *People and Computers VI, Proc. of the HCI '91 Conference*, pages 237–245. Cambridge University Press, 1991. D. Diaper and N. Hammond (Eds).

[KBG89] W. Kim, E. Bertino, and J. F. Garza. Composite Objects Revisited. In *Proc. of the 1989 ACM SIGMOD, 18(2)*, pages 337–347, 1989. J. Clifford, B. Lindsay and D. Maier (Eds).

[Kim93] W. Kim. Object-Oriented Database Systems: Promises, Reality, and Future. In *Proc. of the 19th VLDB*, pages 652–687, Dublin, Ireland, 1993. R. Agrawal and et al (Eds).

[Kin86] R. King. A Database Management System Based on an Object-Oriented Model. In *Expert Database Systems: Proc. of the 1st International Workshop*, pages 443–467. The Benjamin/Cummings Publishing Co, 1986. L. Kerschberg (Ed).

[KN89] R. King and M. Novak. FaceKit: a Database Interface Design Toolkit. In *Proc. of the 15th VLDB*, pages 115–123, 1989.

[McG92] K. McGraw. *Designing and Evaluating User Interfaces for Knowledge-Based Systems.* Ellis Horwood Publishers, 1992.

[MNG90] D. Maier, P. Nordquist, and M. Grossman. Displaying Database Objects. In *Readings in Object-Oriented Database Systems*, pages 551–566. Morgan Kaufman Publishers, 1990.

[MS86] J. Mosier and S. Smith. Guidelines for designing user interface software. Technical Report MTR-10090, ESD-TR-86-278, The Mitre Corporation, Beford, MA, USA, 1986.

[MWHD93] A. Monk, P. Wright, J. Haber, and L. Davenport. *Improving Your Human-Computer Interface: A Practical Technique.* Prentice Hall, 1993.

[Nor88] D. A. Norman. *The Psychology Of Everyday Things.* Basic Books, 1988.

[PaD94] N. W. Paton, G. al-Qaimari, and D. K. Doan. On Interface Objects In Object-Oriented Database. In *Directions in Databases, Proc. of the 12th British National Conference On Databases (BNCOD 12)*, pages 153–169. Springer-Verlag (Lecture Notes in Computer Science), 1994. D. S. Bowers (Ed).

[PaK92] N. W. Paton, G. al-Qaimari, and A. C. Kilgour. An Extensible Interface To An Extensible Object-Oriented Database System. In *The 1st International Workshop On Interfaces to Database Systems (IDS92), Glasgow*, pages 265–281. Springer-Verlag, 1992. R. Cooper (Ed).

[PCE$^+$94] N. W. Paton, R. Cooper, D. England, G. al-Qaimari, and A. C. Kilgour. Integrated Architecture For Database Interface Development. In *IEE Proceedings - E, Computers and Digital Techniques, Special issue on HCI, 141(2)*, pages 73–78, March 1994.

[PDB93] N. W. Paton, O. Diaz, and M. L. Barja. Combining Active Rules and Metaclasses for Enhanced Extensibility. *Data and Knowledge Engineering, (10)*, pages 45–63, 1993.

[Ric87] S. Richardson. Operationalizing Usability and Acceptability: a Methodological Review. In *New Methods In Applied Ergonomics*, pages 125–134, 1987. J. R. Wilson and E. N. Corlett (Eds).

[RJ89] S. Ravden and G. Johnson. *Evaluating Usability Of Human-Computer Interfaces: a practical method.* Ellis Horwood Limited, 1989.

[Shn92] B. Shneiderman. *Designing The User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, second edition, 1992.

[Vea88] Y. Vassilion and et al. Iris - A Mapping Assistant for Generating Designs from Requirements. In *Advances in Object-Oriented Database Systems, 2nd International Workshop on Object-Oriented Database Systems*, pages 307–337. Springer-Verlag, 1988. K. R. Dittrich (Ed).

[Wu90] C. T. Wu. Benefits of Object-Oriented Programming in Implementing Visual Database Interface. *JOOP*, pages 8–16, March/April 1990.