A Visual Interface to a Conceptual Data Modelling Tool

Tim Dudley
Cognos Inc.
Ottawa, CANADA

ABSTRACT

Two of the major problems with currently available software design tools are that they require too much detail too early in the design process, and that they do little to provide sufficient context for the software designer as the design is taking place. This paper presents a visual interface that addresses these two issues, and that in turn leads to a new user interface metaphor. The interface is to a set of cooperating expert systems that acts as a non-intrusive, intelligent assistant in the conceptual data modelling phase of database design. The user interface allows the designer to simultaneously work at the abstract level and at the detail level while building the enterprise model from which the database will be generated. The key features of the interface are graphical gestural input, and the spatial maintenance of logical context through a technique of information filtering, using a structure of hierarchical windows. Planned extensions to the interface include user manipulation of a 3-D representation of extended entity-relationship diagrams, interactive control of behavior modelling animations, and the ability to deal with several models at the same time.

Introduction

For several years, Cognos Inc. has been involved in research in the areas of visual programming and knowledge representation [1, 2, 3]. In 1987, we began to design and build a prototype system, called the Modeller, which combines results of our research in these two disciplines. The intention of the Modeller is to provide a non-intrusive, intelligent design assistant to database designers, which will enable them to generate clear conceptual models of enterprises from which optimal database schemas can be automatically generated. These conceptual models can then be used to produce logical designs (hierarchical, relational, network), that in turn can be used to produce physical designs for specific DBMS implementations, all by using the same intelligent design assistant.

The Modeller consists of a set of cooperating expert systems, which support these three levels of database design (conceptual, logical, and physical). It is based on an internally-developed taxonomy of concepts for modelling the behavior of an enterprise, using an extended entity-relationship (EE-R) approach to data modeling.

The Modeller itself has four important attributes. First, it's knowledgeable — that is, it has encapsulated in it a large amount of theoretical knowledge about enterprise modelling and database design. Second, it's intelligent — that is, it can reason about the enterprise that it is modelling, even though it has no real-world knowledge about that enterprise. Third, it's non-intrusive — which means that, as an intelligent assistant, it doesn't initiate any dialogues with the designer unless it has been requested to perform a function for which insufficient data has been supplied, or until it makes no sense for the designer to proceed further in the current context. And fourth, it can explain its reasoning. The intention of the user interface is to provide the means for a designer to manipulate the design as directly as possible, and to benefit from the intelligence of the tool, without getting preoccupied with the workings of the tool itself.

How It's Built:

The architecture of the Modeller consists of three layers — the user interface layer, the message-passing protocol layer, and the expert systems layer.

![Figure 1](image-url)
without disturbing the expert systems. In addition, it receives all responses from the expert systems that are being passed to the Interactive User Interface, and determines what effect, if any, those responses will have on the graphical layout. It then updates the database and the display.

The message-passing layer handles the message passing protocols that enable the Interactive User Interface and the expert systems to communicate with each other (in addition to other functions which are beyond the scope of this paper).

The expert system layer is the heart of the Modeller, and is described in detail in [4].

There are several advantages to this architecture. The most prominent of these is that it allows the Modeller to be distributed. The message-passing layer allows the Interactive User Interface to run as a separate task, potentially on a different workstation than the expert systems, and effectively insulates the user interface and the expert systems from any changes to each other. In addition, it allows other future applications to interact with the Modeller, at the same level as the Interactive User Interface, without affecting that interface.

This paper deals only with the user interface for the conceptual data modelling expert, referred to below as the "CDM Expert".

How the Interface Works:
Because the user interface to the CDM Expert so closely reflects the application, it is necessary to briefly describe the way the system deals with the modelling process in order to intelligently discuss the interface.

A conceptual model is created by a set of initial statements, called assertions, that are input by the designer, and that describe the enterprise being modelled. This set of assertions describes the model, using an extended entity-relationship approach. The CDM Expert processes these assertions against the taxonomy of enterprise modelling concepts, inferring what it can about the enterprise, and builds up a conceptual model that can be extended, edited, browsed, queried, and verified, until the designer is satisfied with the model.

Conceptual data modelling is a very difficult phase of the design process. It requires the designer to think at a high level of abstraction, while at the same time, to supply sufficient detail so that an emerging conceptual model of an enterprise makes sense. The designer must be able to readily switch back and forth between these two levels of thinking. In addition, s/he must always know where s/he is in the overall context of the modelling process, and be able to switch from one context to another. The intention of the Interactive User Interface is to support both of those levels at the same time, and to maintain as much context for the designer as possible. This is done using a combination of techniques, including making available several different views of a single model (with changes made in the context of one view being reflected in all other relevant views); gestural input to generate commands to the CDM Expert; and context-dependent filtering of information using a structure of hierarchical windows.

Views of the Model:
The main idea underlying the Interactive User Interface is that, for any one conceptual model, the designer can have several distinct Views, each of which is displayed in its own window. Views are initially selected from a menu. The windows in which they appear are generally movable, scrollable, and resizable, to let the designer arrange his/her workspace as s/he desires. Actions on the model are dependent on which View is designated as the current "listener window" [5]. The listener window can be readily selected, which enables the designer to work at either the high level or the detail level. The analogy is to an ionizing radiation box, that has several windows and a pair of gloved hand-holes associated with each window, by which the user manipulates whatever is in the box. The concept that we are trying to reinforce with this metaphor is that there is only one model, but there are several different ways of looking at it. Several Views may be visible at any one time, but only one can act as the listener window at a given time. Anything highlighted in one View will have its corresponding representation highlighted in all other Views, and any changes made to the model will be reflected in all Views.

The Interactive User Interface is menu driven, and utilizes direct manipulation techniques. Figure 2 shows the current menu bar, with the Operations menu open:

![Menu Bar](image)

**Figure 2**

In the case of multi-tasking, each model has its own set of views. This problem is addressed later in the paper.
The File and Edit menus provide the standard functions, and need not be addressed here. The Status menu gives the designer the facility to set conditions on the Modeller, such as turning on/off the checking for graph connectivity, applying or not applying default rules, and several other conditions. Figure 3 shows the Views menu:

Figure 3

When the designer opens a new design from the File menu, the EE-R Graph View is opened by default. The designer can then enter the design graphically, using the gestural input described below. Alternatively, she can open the Views menu and select the Assertions View, and then key in assertions that describe the enterprise. The CDM Expert uses these assertions to reason about the enterprise and to build up the conceptual model. Once the enterprise has been described to the point where the designer wants the model verified by the CDM Expert, she can select the “Process Model” command from the Operations menu. This causes the CDM Expert to do global checking on the model (based on the switch settings from the Status menu), post any errors, and open and display the Results View. Figure 4 shows an extended entity-relationship diagram displayed in the EE-R Graph View:

Figure 4

Figure 5 shows the same display after a “Process Model” command has been selected. It shows the Results View displayed in front of the EE-R Graph View, with the Assertions View open.

Figure 5

The designer can now work at either the detail level (in the Assertions View), or at the conceptual high level (in the EE-R Graph View), by clicking in the desired window to designate it as the listener window, and bringing it to the front. Any of the windows can be resized, so that both the detail view and the high level view of the model are visible at the same time, along with the results of the last “Process Model” command.

We believe that this Views metaphor provides a very powerful means of providing and maintaining context for the designer. In a single interface, it combines two approaches to the design task — the visual, spatial approach and the linguistic, command language approach. Both approaches are suitable to the design task, but for different reasons at different stages of the process. The metaphor gives the designer the capability to work at both the high level of conceptual design (in the EE-R Graph View, using the gestural input techniques described below), and at the bottom end, detail-supplying level (text editing the assertions in the Assertions View), and to readily switch back and forth between them. It also frees the designer from the constraints of having to work in a specific, pre-defined design methodology. The technique of highlighting all representations of selected objects in their own (separate) Views preserves the context in which the designer is working, and reinforces the designer’s mental image of the conceptual model that she is designing.

2The problem of suiting an interface to a task is addressed in [6].
In order to readily support the conceptual level of the design process, it is necessary to provide the facility for the designer to very quickly sketch in a design without forcing him/her to provide excruciating detail. Our experience with practicing database designers is that the first thing they do in the conceptual phase of the design process is identify the entities involved in the enterprise, then arrange them into an ER diagram, then go through the diagram and determine the cardinalities of the relationships. At any point in this process, the designer may realize that she has not made the right decisions with respect to the entities that constitute the enterprise, and will go back and change the diagram, introducing new entities, deleting existing ones, expanding certain ones, combining others into groups, changing relationships, etc. The Modeller supports this activity by providing gestural input, and by using defaults to fill in the necessary detail. (We are assuming a two-button mouse, with the following general conventions: The right button is used to select an object, and to bring up a menu, or initiate the HELP facility, relating to the selected object. The left button is used for everything else — selecting objects, opening objects, dragging, etc.)

By recognizing a limited number of mouse actions, the Graphics Manager is able to determine the designer's intentions from mouse activity, and to translate that activity into assertions when it is appropriate to do so. These assertions are then sent to the CDM Expert through the messaging layer. The conceptual model is built up as if the assertions had been entered through the keyboard or read in from a file — in fact, the CDM Expert cannot distinguish the input source.

In addition to the normal graphical manipulation capabilities, mouse actions can be used to create and delete entities, relationships, and annotations. In addition, the designer can use the mouse to "open" entities in order to browse through the information associated with them, and to use a variety of button and menu selections to cause assertions to be created and sent to the CDM Expert.

The functions that constitute the gestural input facility are the following:

1. **Creation of an entity:**
   
   ![Figure 6](image)
   
   (In an empty area of the screen: mouse button down, drag cursor, mouse button up.)

   This causes the Graphics Manager to draw the entity, and to format and send an assertion to the CDM Expert that creates an entity, with a default name, and default properties.

2. **Creation of a relationship between two existing entities:**
   
   ![Figure 7](image)
   
   (With the cursor inside the boundary of an entity: mouse button down, drag cursor to inside the boundary of the second entity, mouse button up.)

   This causes the Graphics Manager to draw the relationship, and to format and send a set of assertions that create the relationship (with a default name) and cause a default set of cardinalities to be inferred.

3. **Creation of a "dangling" relationship:**
   
   ![Figure 8](image)
   
   (With the cursor inside the boundary of an entity: mouse button down, drag cursor outside the boundary onto blank area, mouse button up.)

   This causes the Graphics Manager to draw the relationship, and to format and send a set of assertions that create the relationship, and to create a "ghost" entity to act as the second entity in the relationship. The CDM Expert can then assign a default name, and cardinalities to both ends of the relationship. The Graphics Manager does not show the ghost entity, and as far as the designer is concerned, the relationship she has just created is incomplete. The point is that the system will temporarily resolve the relationship, in order to let the designer get on with what she is doing, and not notify him/her until she asks the system to process the graph. At that point, the system will notify the designer that there is an unresolved relationship in the enterprise. The designer may leave the relationship unresolved as long as she wishes — the system doesn't force the issue. (After all, it is only a design assistant...) However, the designer may explicitly resolve the relationship by dragging the desired entity icon onto the end of the relationship icon, or by dragging the end of the relationship icon onto the entity icon.)
• Creation of an annotation or comment:

![Figure 9](image)

(With the cursor positioned in an empty area of the screen, mouse button down, no cursor movement, mouse button up. "Single-clicking").

This action enables the designer to key in annotations that can be placed on the extended entity-relationship graph. Currently, we do not intend to send these comments to the CDM Expert, and instead will store them in the graphics database, as part of the EE-R graph, because they tend to be positionally dependent. (We are considering including them as comments in the Assertions list, if we can figure out where to put them.)

The normal graphical editing functions include the following:

• Selection and highlighting of an existing entity/relationship:

![Figure 10](image)

(With the cursor inside the boundary of an entity or relationship: mouse button down, no cursor movement, mouse button up.)

This causes the Graphics Manager to highlight the entity, and to enable/disable menu items as appropriate. It does not disturb the CDM Expert. Subsequent entities/relationships may be added to the existing selection by positioning the cursor on the desired object, and holding down the Shift key while clicking the mouse button.

Note that graphical objects may be deleted while they are selected by hitting the backspace key, or by selecting Clear from the Edit menu.

• Selection, highlighting, and moving of an entity/relationship:

![Figure 11](image)

(With the cursor on the "hot spot" of the object to be moved, mouse button down, drag outline of object to desired position, mouse button up.)

The Graphics Manager considers the boundaries of entities to be "hot spots". When the cursor is dragged across an entity boundary, either from inside the entity or from outside, it changes shape to indicate that it is over the entity’s hot spot. When the cursor is over the hot spot, the entity is temporarily selected and highlighted, and may be repositioned. Relationships have three hot spots, one at each end and one in the middle, in order to enable them to be completely repositioned, or for one end or the other to be moved.

In the case of moving an entity, the Graphics Manager does not disturb the CDM Expert; however, moving a relationship can cause changes in the conceptual model. In that case, the Graphics Manager determines what assertions are necessary to send to the CDM Expert in order to have the model accurately represented in the EE-R diagram, presents the designer with appropriate dialog boxes, if required, then formats and sends those assertions.

• Selection of a set of entities and relationships:

![Figure 12](image)

(With the cursor positioned in an empty area of the screen, mouse button down, drag cursor to diagonally-opposite corner of the bounding rectangle for the desired objects to be selected, mouse button up.)

This action does not currently send anything to the CDM Expert. However, a future possibility is that a selected set of entities and relationships would be formally defined (via the menu) as a Group, in which case their graphic representation would be collapsed and replaced with an entity icon that has visual attributes that are different from that of "low-level", or basic entities. The Graphics Manager would then format and send the requisite assertions to the CDM Expert in order to introduce the hierarchy into the model. In this way, it would be very straightforward to build up hierarchies of EE-R graphs.
Opening of an object:

(Figure 13)

(With the cursor positioned over the object, mouse button down, no cursor movement, mouse button up, repeat. "Double-clicking".)

This action highlights the selected object, and brings up a window that contains the next level of detail associated with that object. In the case of a grouped object, the window contains the structure of the group. In the case of a basic object, the window contains the properties of the object, with buttons that, when activated, bring up further detail. This is the basis of the information filtering technique explained in the next section.

Context Dependent Information Filtering:
The intention with the information filtering technique is to focus the level of abstraction on a subset of the enterprise model, based on the context that exists when the information is requested. The technique is particularly applicable in two different situations — one, when the designer has asked the CDM Expert to process the model, and the processing results in an inconsistent or unworkable model; and the other, when the designer is browsing the model to get at information that is related to particular objects, or parts of the model.

For example, Figure 14 shows the results of a "Process Model" command: (This is the same situation as shown in Figure 5, taken one step further.)

(Looking at the Results window, the designer has chosen to resolve the initial contradiction by modifying Assertion a6. Rather than having to go to the Assertions View and locate a6, the designer can click on the assertion number in the Results View, which brings up the assertion, and offers all of the reasonable alternative actions that can be taken with it — i.e.: delete it, or change it. As the designer resolves contradictions in the Results View, the statements describing the contradictions in that view are grayed out, since it is no longer current. The designer can continue to modify the model by accessing subsets of it via buttons on the Results View, or by going to any of the other views.

In the other situation, in which the designer is browsing the model in the EE-R Graph View, layers of successively finer detail can be accessed through the use of the property sheet. In that View, all entities, groups, and relationships have property sheets associated with them. Double-clicking on an object selects and opens it, and displays its property sheet. (In the case of a grouped object, the property sheet contains the name and a structure diagram of the grouped object.) The property sheet also makes available sets of operations that are appropriate for the part of the model with which the property sheet is associated, and filters the available information such that information in the model that is directly associated with the highlighted object(s) is directly accessible. Figure 15 shows the property sheet for an entity named "Client":

The property sheet contains buttons that allow the designer to go further into detail, and to do specific operations that make sense in the refined context that is being created by the resultant cascading windows. Figures 16 and 17 show an example of this:
In this case, the designer has hit the "Related to" button on the property sheet, to see the list of entities that are related to Client. A window has come up that shows the list (which is a filtered subset of the Entities View), and that contains buttons that enable the designer to add or delete relationships. In Figure 17, the designer has hit the "Add Relationship" button, which brings up a window containing a list of entities to which a relationship may be established.

At this point, the designer may select an entity from the list, or key in the name of a new entity. The CDM Expert will create the relationship, with default names and cardinalities, and will also determine the type of relationship, if the designer has not specified one through the other button in the window. This same technique of using buttons to open windows containing context-sensitive information and a filtered command set is used in several of the other Views.

The combination of the gestural input, the Views metaphor, and the information filtering techniques provide a very strong basis from which a designer can work. Logical context of the design is maintained through spatial and temporal structuring of the windows. Through the cascading windows, the designer always knows the context in which s/he is working, and specifically, what effect the commands that are being entered will have. The techniques enable the designer to work at the detail level, while the system maintains and displays the conceptual level or overview of the current state of the model. The system also allows the designer to hide detail as necessary, by giving control of the windows to the him/her. These techniques, in combination with the underlying intelligent CDM Expert, provide a very powerful tool for use in enterprise modelling.

Current state of affairs:
The Modeller is being implemented as a series of prototypes. The first prototype runs under MS-DOS, and consists of the CDM Expert, and a windowed text interface (written in C). The initial interface supports tiled and pop-up windows, but has no graphics, and no pick function. The expert systems deal with only one model, because of the lack of multi-tasking capability in MS-DOS. The modelling process is restricted to static structural modelling. This prototype included one logical-level expert (relational), and four physical-level experts. The physical experts produce schemas for a generic relational database, PDL, Generic SQL, and PowerHouse Starbase, with communication from expert system to expert system via file transfer. The purpose of this release is to get user feedback on the validity of the underlying concepts of the expert systems, to verify the taxonomy of the enterprise modelling concepts, and to evaluate the overall functionality and usefulness of the system. An attempt was made to maintain some of the information filtering concepts from the Interactive User Interface described in this paper; however, the user interface for this release is a throw-away.

The second prototype consists of the first prototype, plus the Interactive User Interface as described here. It also includes a limited automatic diagram layout facility, which produces an EER diagram from a set of textual assertions. The diagram is initially laid out randomly. The user can then either clean up the diagram manually, or request the system to gather entities together, based on their relationships to each other. This results in a diagram of clusters of related entities, which then must be manually cleaned up. (Our experience is that it takes about thirty minutes to produce a reasonable diagram, working with a model containing about 200 entities.) The facility to save and restore graphical layouts is provided. Behavior modelling is not included, except for a View of structure-implied behavior associated with a relationship and a pair of entities. The prototype runs on an i80386-class machine. The expert systems are implemented in Arity Prolog, and the Interactive User Interface...
Interface using the Windows/886 Presentation Manager. Again, this prototype will deal only with one model at a time. We intend to introduce a hypertext-style context-sensitive help facility with this prototype.

The third prototype will extend the second one to include multitasking and behavior modelling, and is intended to include more third level expert systems to produce physical DBMS schemas. The prototype is currently targeted to run under OS/2 with the OS/2 Presentation Manager, and is designed to work with multiple, simultaneously active models. The user interface will be extended to include a set of Views for each model, and a mechanism for controlling those sets. It will also include a visual, direct-manipulation means of moving information among models. Each model will have its own “canvas” within which its Views will be contained, each of which will be color coded in order to help maintain context. We expect this to present some extremely challenging problems in the areas of information management and spatial presentation.

Future directions:
Once the third prototype is produced, several innovative research avenues can be explored, based on that work. Three (of several) extensions to the Modeller capability that are currently under consideration are: the implementation of a 3-D View of the EE-R diagrams, with attendant manipulation facilities; animation of the behavior modelling activity, including surface generation over a time-dependent set of EE-R diagrams; and the reverse engineering activity — that is, taking an existing database schema as input, and extracting a conceptual model from it, that can then be verified by the Modeller.

It has also become apparent from our work on diagramming, that we should be able to formally define patterns as being semantically equivalent to other patterns. We are investigating the possibility of representing these patterns in the form of regular expressions, with the intention of eventually being able to mathematically minimize an extended EE-R diagram of a conceptual model.

Summary:
This paper has presented a visual user interface to a unique expert-system-based tool that provides intelligent assistance in the conceptual modelling phase of database design. The tool is being implemented as a series of prototypes, running on the i80386 class of machines under commercially available operating systems. The interface combines several techniques, notably gesture input and information filtering, that support the design activity at both the conceptual level and the detail level simultaneously. A new metaphor has been introduced, called the Views metaphor, that is extensible, and that allows the mixing of several interface styles within one application interface.

References:


