ELECTRONIC **WORKSHOPS IN COMPUTING**

Series edited by Professor C.J. van Rijsbergen

Jessie Kennedy and Peter Barclay (Eds)

Interfaces to Databases (IDS-3)

Proceedings of the 3rd International Workshop on Interfaces to Databases, Napier University, Edinburgh, 8-10 July 1996

Paper:

TOTEM: An Interactive Tool for Creative Data Modelling

Alison Crerar, Peter J. Barclay and Richard Watt

Published in collaboration with the British Computer Society

BCS

Springer

TOTEM: an Interactive Tool for Creative Data Modelling.

Alison Crerar, Peter J. Barclay and Richard Watt

Department of Computer Studies, Napier University
Edinburgh, Scotland.
e-mail: {mac, pete}@dcs.napier.ac.uk
http://www.dcs.napier.ac.uk

Abstract

Data modelling is a crucial and early step in the design of all but the most trivial database systems. Although a fundamental skill for information systems designers, data modelling is not easy to do well and novices are known to have considerable difficulty a) in generating a satisfactory model from a given scenario and b) in appreciating the semantic nuances of a given model from its diagrammatic representation. Given a candidate model, novices typically find it difficult to envisage whether a set of specified query types can be met and what the implications of the model are for storage and access. One of the main differences between novice and expert performance is held to be this capacity to visualise the implications of a model from its diagrammatic representation. These two skills, generation and evaluation, are closely inter-linked: clearly, some competence in the second is necessary for self-monitoring during the model generation process; however, a high degree of facility is required if data modelling is to be truly a design process and not just a mechanistic activity whereby the model is derived from the scenario by simple transformation. The concept of data modelling as design, as a creative process, has been under-emphasised thus far in the pedagogic literature. We aim to broaden the prevalent convergent style of introductory texts (which tend to concentrate students towards the specimen 'correct' solution), by promoting a more divergent approach, namely, that there is likely to be a set of feasible solutions, candidate solutions will have advantages and disadvantages and creative solutions may involve a re-conceptualisation of the scenario. To promote these ideas in teaching, supporting tools are needed. While there are a number of software packages existing to help with the graphical aspects of model creation (modelling tools), there is a dearth of software offering modellers an environment in which they can create a model or partial model and then interact with it better to understand its 'behaviour'. This paper explains the rationale behind and work-in-progress on a concept-demonstrator TOTEM (TOol for TEaching Modelling) which owes much to the new generation of programming environments such as Visual Basic and Delphi. TOTEM is conceived as a Windows-based rapid 'design and test' environment for extended Entity-Relationship modelling that supports interaction with multiple candidate solutions, any of which can be explored with test data. We report a video study to assess the usability of design mode in single user, single model conditions and present the main ideas behind the next planned phase of work.

Keywords

creative modelling, data modelling, Entity-Relationship modelling, expert, modelling tool, novice, teaching, usability evaluation, video.

1 Background

The work reported here was undertaken in the context of an academic Computing department where the teaching of various modelling techniques is a central part of training software engineers. It represents ongoing collaboration between the HCI Group (first author) and Object Systems Group (second author) aimed at achieving a better understanding of the *process* of data modelling and ultimately embedding that understanding in the design of more effective software tools. In a previous paper we reported the use of split-screen video techniques to study the on-line behaviour of three competent data modellers working, two with a Windows-based tool and the other with pen and flip-chart [1]. There were a number of interesting insights arising from that work, particularly concerning the flexibility of pen and paper as a design medium; by contrast, the software tool imposed a considerable cognitive load. An unexpected observation was the reliance of the computer-users on pen and paper for drafting their initial models. From this earlier pilot study we were alerted both to usability issues likely to reduce the effectiveness of interactive modelling tools and to desirable features (such as ‘multi-page working’, maintenance of model histories and the need for a semantics-free designers’ scratch-pad) which if incorporated in support tools would begin to redress the usability gulf evident between the two media.

The work described below builds upon the previous study by targeting the problems of novices engaged in deriving a conceptual data model from a real-world scenario: a typical undergraduate exercise. However, it differs a little in emphasis from the previous study, focusing on two areas of current concern for us, namely:

a) that modelling as taught through standard texts tends to reinforce the erroneous idea that there is one ‘correct’ model for a given set of requirements, or to state it less strongly, that once a model that satisfies the requirements has been found, the job is done [3], and
b) that student modellers are generally not good at assessing quickly the implications of a diagrammatic model: either their own or somebody else’s.

Let us amplify a little on these two points. Text books that include questions and model solutions are inevitably subject to space constraints. This, combined with their expository nature, tends to favour uncomplicated examples for which the model solution can often appear to be the only sensible one. Likewise, in the teaching situation, time constraints, together with the need to ensure that all members of the group attain basic competence, conspire against introducing possibly confusing ‘counter-models’. Reflecting on the differences between expert (staff) and novice (student) modelling performance one striking distinction seems to be the expert’s ability to envisage the implications of a diagrammatic representation ‘at a glance’. In this respect expertise in modelling seems akin to expertise in other domains such as chess [4] and architecture [5], where well known patterns comprising a number of sub-components are recognised and/or generated in meaningful ‘chunks’. It has been suggested that expert modellers often use one or a small number of standard models (e.g., for banking or manufacturing) which they modify as the situation demands [3]. Simsion [18] refers to these re-usable models as ‘generic models’ and makes the useful distinction between *standard structures,* which are re-usable components of a data model and *generic models* which are whole models of commonly occurring business functions which readily transfer between applications. The identification of such re-usable patterns is a topic of current interest in software engineering more generally [6]. The abilities to visualise the effects of a model and to spot quickly the opportunity for generic model re-use were hallmarks of an expert performance observed in our own video study (see Section 5.2).

Encouraging students to generate and evaluate alternative models seem self-evidently valuable, if not essential, in transforming novices into experts. Moreover, we know that real world tasks are not nearly so neat as the text book examples:

> The conceptual modelling problem, in most realistic cases, involves a more open-ended, semantically rich problem space in which the modeller is forced to engage in both broader conceptual thinking, as well as focused problem-solving activities. It is more realistic to think of the solutions (conceptual models developed) in terms of degree of correctness than a unique solution. [7, p.87]

However, the weakness is, as we have noted, that novices are not yet in a position to appreciate the semantic nuances and must rely on a tutor to point out whether their model is in the set of feasible solutions or is flawed. This situation is not helped by current database software. The burgeoning choice of PC DBMSs such as dBase, Paradox and Access (very widely used in workplaces by both trained computing professionals and end-users) have no support for conceptual modelling as such; users are expected to dive straight in with table layouts. On first glance the illusion is of the rapid production of superficially impressive databases with GUI front-ends, but experimental evidence suggest that poor systems are often the result [8].

More comprehensive CASE tools do include data modelling environments: however, as Shanks and Simsion [9] observe, “... the support which CASE tools provide to database designers is analogous to the support a word processor gives to a writer.... A real danger is the false sense of security which a CASE tool may give to a database designer.” (p.294).

Drawing these threads together we observe:

- a tendency to train student modellers towards ‘the correct data model’ or ‘a correct data model’ for a given scenario
- that novice modellers are not adept at assimilating swiftly the implications of a model
- that the ubiquitous ‘user-friendly’ DBMSs provide no support for the process of modelling

In the following sections of this paper we describe the development of a prototype rapid modelling tool, TOTEM (TOol for TEaching Modelling), that has been designed to help novices to generate and evaluate alternative data models. Mindful of the importance of feedback in learning to evaluate solutions [10] and of the expectations of learners, now accustomed to responsive graphical user interfaces [11], our approach has been strongly influenced by the highly effective ‘design and run’ programming environments typified by products such as Microsoft’s Visual Basic [12] and Borland’s Delphi [13]. The challenge was to design an interactive environment that would a) be so simple to operate that it would obviate the need for preliminary paper sketches, b) allow students to interact with their models to check their validity and detect problems at the earliest possible stage, c) provide a basis for our ongoing experimental research into cognitive aspects of the data modelling process and d) fill an identified niche by exporting schemata in a format compatible with most of the proprietary PC Windows-based database packages.

The paper is organised as follows: Section 2 outlines previous research on the nature of modelling problems, Section 3 explains the notion of *creative modelling* which our tool aims to foster, Section 4 describes and illustrates the main features of TOTEM giving an overview both of its functionality and its operation, Section 5 presents the results of a small video evaluation study replicating the method used in Barclay et al. [1], and finally Section 6 summarises plans for further work.

2 Understanding Modelling Problems

Conceptual data modelling is the process of analysing the information requirements of a client and constructing an abstract model of the data that needs to be held and of the interrelationships between the data elements. The form of representation used most commonly is Chen’s entity-relationship (E-R) notation [14] or a variant of it. The quality of the initial data model is crucial to a successful outcome, since the logical structures identified in the data model feed forward, often automatically, into the generation of a conceptual schema and ultimately into the realisation of a physical database implementation.

Previous studies on data modelling have shown that a wide variety of errors may occur, although some are more frequent than others; for example, binary relationships are more likely to be modelled correctly than either unary or ternary relationships [10]. Such differences may be attributable either to the greater conceptual simplicity of binary relationships compared to those of other degrees, or perhaps to teaching approaches which never progress beyond initial modelling exercises, chosen for their manageability, towards more realistic examples. For example, students may not realise that they are ‘allowed’ to have two or more distinct relationships between the same pair of entities, even though such a possibility is not forbidden by any modelling guidelines they have been given, simply because they have never before encountered such a case (see Figure 1 for an example). Batra and Anthony [15] have identified degree of nesting (the number of levels of nesting of data) and possibly derivation span (the number or intermediate relations in a derived relationship) as whole-model measures of complexity.

It would be useful if our modelling tools could assist the user to avoid such errors. Moreover, the problems are even harder than these earlier studies suggest. Most of them have concentrated on the individual components of the modelling process, such as choice of entities or determining the degree of relationships, and considered the difficulties the user may experience with each of these facets in isolation. However, we must not neglect also to consider the model in a more holistic way; often, a particular design decision may be ‘right’ or ‘wrong’ only in the context of the model as a whole, or in the light of other decisions already made. A simple example is shown in Figure 1 and discussed below.

Norman [16] has established that errors may be corrected either by self-monitoring, or by deleterious outcomes. Self-monitoring is achieved by the individual’s own, ongoing assessment of his actions; a deleterious outcome occurs when an uncorrected error gives rise to an outcome not in accordance with the individual’s intention. In terms of data modelling, self-monitoring refers to the modeller’s own, question-driven critique of his model as it is developed; a deleterious outcome is where the model is realised as an implemented database which does not meet its requirements. Clearly, error-correction through self-monitoring is to be preferred, and therefore our modelling tools should allow, if not encourage, such self-monitoring.

Figure 1 is one way of showing that students are registered for particular courses, each of which consists of a number of non-shared modules. Each student may take several different modules, but is enrolled for only one course. Furthermore, there is a class representative for each course. As it stands, this fragment may appear simple and uncontentious: but a number of unresolved issues lurk.

For example, whether the relationship *takes* needs to be explicitly represented depends on whether a student necessarily takes all modules which constitute a course, or whether some may be optional. If all the modules are mandatory, then *takes* may be omitted since it may be inferred from the composition of *enrolls_for* and *is_in*; on the other hand if some modules are optional, then *takes* must be included to show which modules each student actually takes. Modellers must be encouraged to explore their models in detail so that such issues are not overlooked.

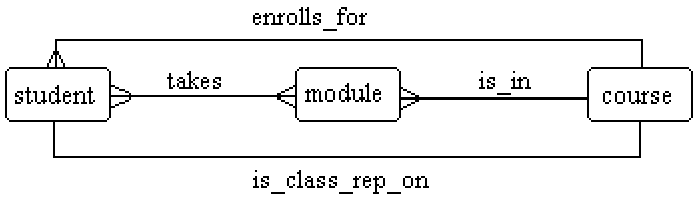


Figure 1. Part of an E-R Model Ripe for Probing.

On the other hand, if modules are optional, and *takes* is therefore required, the relationship *enrolls_for* could, in theory, be omitted: since a module is not shared between courses, the course for which a student is enrolled may be derived from the course association of any module(s) taken (providing an assumption is made that students are not permitted to take modules belonging to more than one course). Here we see how a decision in one part of the model may affect other parts, emphasising the need to review the model as a whole as it evolves.

We should also note that real life may be more complicated than our model suggests; even if *enrolls_for* may be omitted in theory, operational needs may require a record of which course each student is enrolled on before any modules can be chosen, and so for a real information system to be based on the model, the inclusion of *enrolls_for* might nonetheless be required. So modellers should be encouraged to consider pragmatic issues too and to anticipate the practical impact of design decisions, thinking through matters such as the temporal sequence of data acquisition and how this affects query handling.

Another small point may be gleaned from this example. A constraint which is implicit in the scenario is not explicitly represented in the model: the class representative of any given course must actually be enrolled on that course. Although this kind of constraint cannot be represented directly in the E-R diagram, the modeller should certainly document it so that it can be enforced by appropriate mechanisms later in the design process. However, nothing in the diagram encourages the modeller to consider this possibility, which means that the constraint could remain unnoticed and unrepresented until the population of a database with contravening data results in a deleterious outcome.

Data modelling requires a blend of high- and low-level skills. Batra and Sein [10] have shown that expert modellers expend most effort at what they call the *enterprise* and *representation* levels, focusing alternately on holistic issues and the details of recording their solution; compared to novice modellers, they spend comparatively little time at the *recognition* level, trying to relate some feature of the scenario modelled to known modelling concepts. This suggests a need to increase novice modellers' awareness of whole-model considerations.

3 Creative Data Modelling

As we have noted above, most expository texts on modelling, for reasons of space and clarity, tend to represent the process of entity- or object-modelling as a straightforward one of *identification* of the things of interest, of their attributes and of the relationships between them. The implication is that there is an objective world to be modelled and that the skill is in discovering and describing it (rather than in constructing and interpreting it). Tutorials typically include guidelines such as "Nouns are good candidates for entity types. To determine whether a noun should be designated as an entity, the following rules may be applied...." [17, p. 30]. Such rules of thumb are extremely valuable if advocated by tutors as helpful heuristics; the danger is in promoting them as a fool-proof method for deriving a ‘correct’ model. We acknowledge that deriving a model in this way can be a useful first step in capturing and formalising a client’s conceptual model of a system, but fear that practitioners trained in this mechanistic way will tend to conduct analyses which merely document and perpetuate existing systems, since they are unlikely to think beyond current practice as described to them.

By contrast, Simsion [3,18] and Shanks and Simsion [9] have promulgated a ‘creative’ approach to data modelling, stressing it as a *design* activity rather than a descriptive activity, which can generate a number of potential solutions each satisfying stated requirements in subtly different ways. They advocate the deliberate exploration of novel solutions as a necessary part of the modelling process. The following quotation captures the essence of their position:

> The view of object selection as a straight forward and deterministic process is neither theoretically sustainable nor supported by practical experience. Entities attributes and relationships are classifications of “real world” data, and it is possible to propose alternative classifications, subtly different category predicates, or entirely different views of the problem. Experienced data modellers will propose different workable solutions to even quite simple problems. [9, p. 298].

While the value of procedural guidelines as a training aid is not at issue, it seems curious that the ‘creative’ approach has not taken root in academic circles and has been largely overlooked in the literature.

Simsion [3] suggests that a good way of generating alternative models is to apply De Bono’s advice about lateral thinking [19], by refining the entities in an existing model into successively finer-grained candidates and then re-combining them in different ways and evaluating the resulting configurations for utility. Although the effort involved in doing this is clearly greater than in the more traditional rule-following paradigm, it is justified if the result is a more effective and flexible representation. The following example gives a flavour of creative modelling in practice. The important thing is the consideration, not of different versions of the same model, but of different interpretations of the same problem domain as shown in Figures 2a and 2b.

Suppose the client is a company, part of a large group, that has evolved over time from a traditional, hierarchical organisation with autonomous budget-holding departments each employing full-time staff, to one where the departments still exist in name, but staff are now almost entirely self-employed, hired from agencies on short-term contracts. An analyst eliciting information from management is likely to hear the organisation and its activities described in terms of the ingrained descriptors such as ‘employee’ and ‘department’. If trained in the rule-following school of modelling, he will translate these nouns into entities, producing a model like Figure 2a. Whatever the final version of such a model, it will retain the historical perspective. On the other hand, brain-storming by a more creative analyst might produce a range of alternative views, such as Figure 2b, which abandons the notion of a ‘department’ and captures the idea of ‘project’ as the thing of central concern. It may be that such a view, while radical, would simplify internal processing a good deal, since in practice project teams may be cross-departmental, each department having to set up its own ‘mini project folio’ to charge its services. The essential point is that the two models are quite different and Figure 2b might not occur to a modeller of more rigid training.

Whereas alternative models may exhibit different degrees of correctness and we would like to see evaluation of multiple solutions more widely stressed in teaching, experience has shown that certain components which are clearly erroneous regularly occur in students’ modelling exercises. We have attempted to pass on the benefits of this experience by warning students of common traps; such information is typically presented as a set of ‘tips’ rather than as part of a coherent design method. Nonetheless, errors often remain hidden in models until the model is ‘explored’, which might be viewed as a form of ad hoc testing. However, despite exhortations that the model should be reviewed at each stage, the actual exploration which reveals residual errors often does not happen before implementation. Whilst this might sometimes be inevitable, it would be useful to narrow the breach between design and implementation so that such errors may be discovered sooner and remedied more cheaply. This is one of the aims of TOTEM, the data modelling tool we proceed to describe.

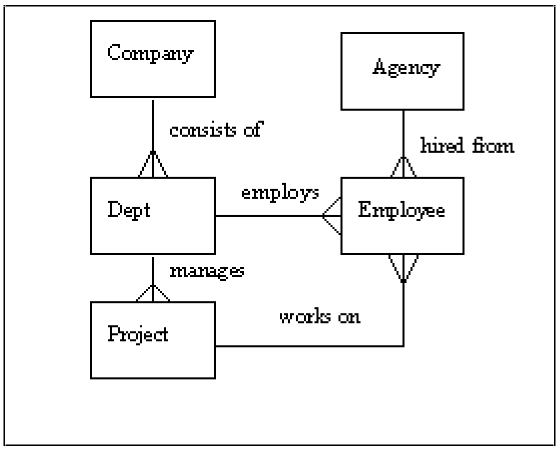


Figure 2a. A Traditional Interpretation of a Business Scenario.

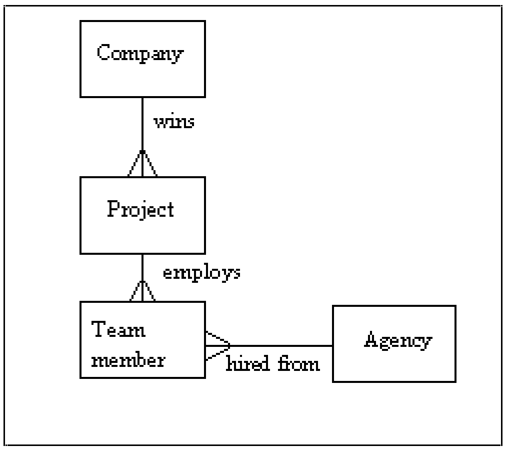


Figure 2b. An Alternative Interpretation of the 'same' Scenario.

4 TOTEM: an Overview

TOTEM is still in the early stages of an evolutionary development; the implementation is the work of the third author. At the time of writing, the facilities described in Section 4.1 have been completed in a multiple document interface (MDI) version, and have been subjected to regular formative evaluation during development by four expert reviewers. The first two authors initially specified the system and their testing has been a) for conformance with the functional requirements and b) with a view to addressing the human-computer interaction issues raised in their previous research [1]. Independent usability testing has been done by two former students both of whom have had experience of implementing interactive modelling environments. TOTEM has been implemented in Microsoft's Visual Basic v. 3.0 and runs under Windows 95 on IBM compatible PCs. In parallel, a second prototype is underway for the same platform using Borland's Delphi. Visual Basic happened to be the third author's development tool of choice and has proved highly effective for rapid prototyping. Delphi, which is an object Pascal with an interactive interface-design front-end much like Visual Basic's, is a significantly more powerful development environment. Delphi is a compiled language with a very fast execution speed, whereas Visual Basic is interpreted and inevitably slower. It is too early to say whether issues such as scalability and database support for our 'explore and test mode' will in the end favour one development environment over the other. At present both versions aim towards the same underlying functionality though in the Delphi version we are exploring different interface designs and gaining more experience with the product.

TOTEM has been designed primarily as a vehicle for teaching and research, drawing on the insights of previous studies of modelling, notably those referred to in this paper, and our own collective experience of teaching data modelling, but driven most strongly by the desire to test out some recent insights regarding the usability problems computer-based data modelling environments can introduce [1]. The latter study was a small-scale experiment comparing the modelling behaviour of three competent modellers, two using a Windows-based modelling environment (developed in-house, though not by ourselves) and the third using pen and flip-chart. Briefly, the flip-chart was found to be a much more flexible design medium, allowing non-standard annotations, freedom in order of working, directness of interaction, viewing of entire model history (by laying previous sheets on the floor!) and so forth. By comparison, operation of the computer-based environment was found to place a significant cognitive load on the modellers. It was clearly superior for manipulation of a diagram, but during model creation, operating the interface proved a hindrance to the process of modelling. Both computer users elected to work entirely with pen and paper for sketching out their initial designs (much to the researchers' surprise!).

Barclay et al. [1] made a number of recommendations for features that would begin to bridge the usability gap for data modellers between pen and paper and computer-based GUI environments; the main ones were: freedom in the order of working in design mode, provision of a scratch-pad for jotting ideas, ability to make ad hoc annotations, support for multi-page working and, related to the last point, the ability to freeze previous versions of a model. In designing TOTEM we have addressed all of these suggestions except the ability to make ad hoc annotations on the diagram (i.e. to make free-form markings which are not part of standard notation, such as crossing out a relationship line). In addition, turning to the needs of novices learning modelling, our previous ideas have been extended to include the specification of an explore-and-test mode. The need for multi-page working identified earlier fits well with inculcating a creative approach to modelling in students. Taking advantage of the Windows MDI (multiple document interface) facility multiple windows can be used to view the history of a single model, assess alternative solutions to the same problem or compare solutions to different problems. Section 5 describes a repeat of our previous video study [1] using a single document interface (SDI) version of TOTEM. We plan to introduce TOTEM in its present form into the classroom in the coming academic session. Once the functionality outlined in Section 4.2 has been completed, the effectiveness of the explore-and-test mode as a teaching aid will be investigated in detail.

To summarise, TOTEM has two modes, *design (and validate)* and *explore (and test)*, corresponding to *design (and code)* and *run (and test)* in a familiar software development environment such as Visual Basic or Delphi. However, TOTEM will allow the design-mode model to be viewed simultaneously with its 'run time' version. So it is envisaged that the stack of windows open on a modeller's desktop may include several design windows and several explore windows, and they may be arranged (tiled, cascaded, positioned, minimised, maximised) and flipped between as required.

Below we visit each of the two modes, describing the main functionality and some of the key operational characteristics.

4.1 Design (and Validate) Mode

Data models are created, edited, saved and retrieved in design mode. All the usual functions of creating, amending and deleting entities, relationships and attributes are supported, together with the facility to rearrange the placement of entities (relationship lines follow them automatically) on the screen. A guiding principle in design mode is that as far as possible all interactions (for example labelling of entities and relationships) should take place *on the diagram* itself rather than in some pop-up window that obscures the canvas. Table 1 gives an overview of the functions design mode currently supports. The left column shows the objects that can be manipulated, and listed in the same row as each are the operations they respond to. Thus the colour of all entities may be changed globally, or the colour of an individual entity may be changed separately. We are keen to test out this small set of core functions fully, achieve a high degree of user satisfaction and consult widely with our student modellers before proceeding to expand the capabilities (for example perhaps adding the ability to act upon subsets of models).

A flavour of the interactional characteristics of TOTEM will be given with reference to Figure 3 which shows a model in the early stages of development. All mouse clicks described are single clicks. The three entities were created by clicking three times on the '+' icon and dragging the resulting entity symbols into place. Entities may be connected into relationships before or after they are named; relationships may be named any time after creation; attributes may be added before an entity is named: there is no enforcement of ordering. The right mouse button is used to invoke context-sensitive pop-up menus. If the cursor is over an entity the pop-up menu offers Name/Attributes/Colour/Delete. In Figure 3 the entity 'Employee' has been selected in this way, the 'Attributes' menu item has been chosen and attributes are being entered. If the cursor is over the background (and depending on the number of entities present) the pop-up menu offers Add entity/Unary relationship/Binary relationship/Ternary relationship/Add post-it. Creating a binary relationship is done by selecting from the menu and then single clicking the two entities involved. Clicking on a relationship line invokes a pop-up menu offering Name/Delete. Naming of entities and relationships is done in-line on the diagram to create, as far as is possible, a feeling of working directly with the objects. The crow's feet are toggled on and off with a single click. If entities are dragged their relationship lines follow. Post-its may be moved or re-sized simply by dragging.

TOTEM respects the belief that model creation should be allowed to happen without the imposition of strict semantics: the modeller should be allowed to record ideas as they occur, even if out of sequence; parts of the model may be left unfinished, labels omitted, etc. Ideas are not generated according to logical frameworks, though such a framework must be later imposed on them. Therefore semantic checking does not intrude during construction of the model, but can be invoked as required by a proposed validate function. Without an understanding of real-world semantics, the validate function cannot prove a model correct, but it can at least alert the user to certain inconsistencies.

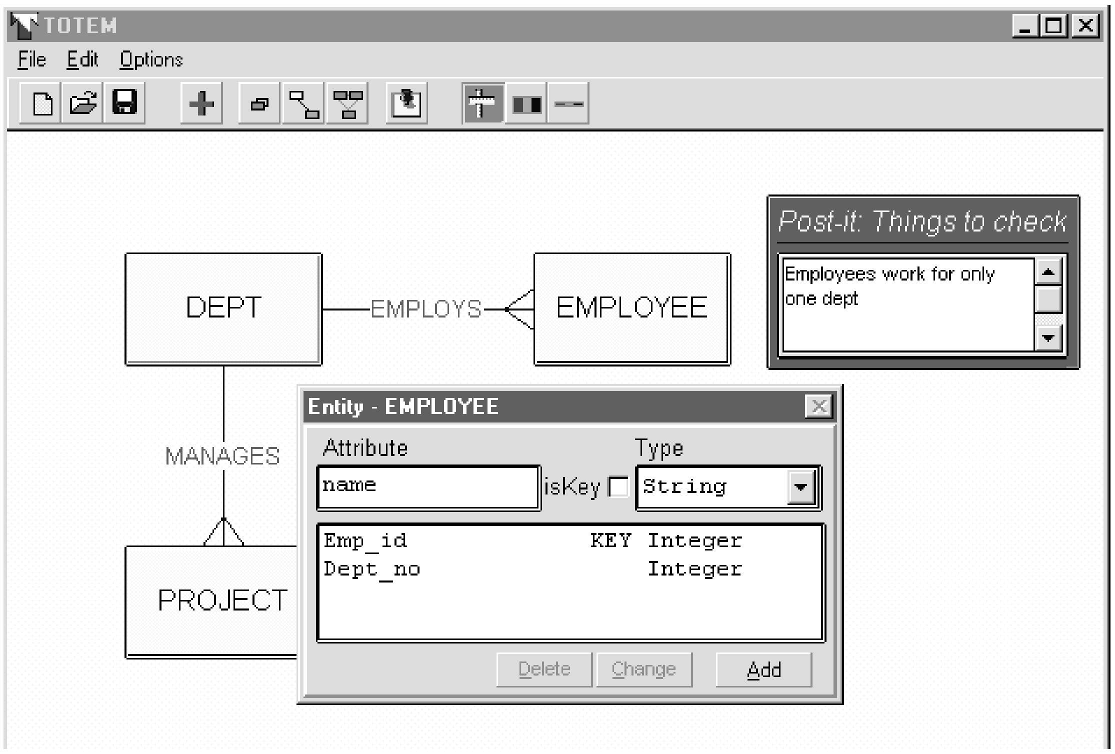
Object	**Operations supported**						
Whole model	Open	Save	Clear				
All entities	Change colour						
All relationships	Change colour						
All post-its	Hide	Show					
Entity	Create	Delete	Move	Change colour	Add, Edit, Delete name	Add, Edit, Delete attribute name(s) and type(s)	Mark, unmark attribute(s) as key
Binary relationship	Create	Delete	Change colour	Add, Edit, Delete name	Add, delete crow's feet		
Post-it	Hide	Show	Create	Delete	Move	Re-size	Add, Edit, Delete label and text

Table 1. Overview of the Functions Currently Available in TOTEM's Design Mode.

The (planned) validate function will offer incremental semantic checking of a model. In keeping with our premise that to encourage creativity modellers should be allowed to work in the way that suits them best, we propose that a model may be checked in its entirety or by selected features only. For example, the modeller may wish to select entities only, checking that they are all named, or to check attributes only (to check for occurrences of similar but not identical names which may indicate a typing error). As well as checks applicable to entities, attributes and relationships individually, we envisage a collection of global checks which will address the overall integrity of the model, such as detecting entities which are related to no others, or possible fan traps. A 'check all' option is planned, which applies all available consistency checks to the model; this will typically be of use when the modeller considers that the model is nearing completion. Thus, we propose a stratified regimen of consistency checking, which does not constrain the user but provides informative feedback appropriate to the state of the model.

Some of the potential problems identified during validation may be categorised as warnings, and some as errors. If two attributes have similar names, this is a warning only, since it may be what the user intends; however, if an entity is unnamed, this is an error, since the model cannot be considered complete while this condition obtains. In order to run the model in the explore mode, it must contain no errors.

TOTEM provides a simple versioning mechanism for models. At any time, the current state of a particular model may be stored as a named and date-stamped version, to which the user may later revert; this allows backtracking in the case of the erroneous development of a model. However, this feature is also intended to allow the concurrent development of alternative versions of the same model (which may be evaluated relative to each other in the explore mode). Consideration of alternatives is essential to the process of creative modelling; we have argued earlier that it is necessary to compare not only the choices available for individual design decisions, but entire models encompassing different configurations of these choices.



Employees work for only one dept

name

String

```
Emp_id              KEY  Integer
Dept_no                  Integer
```

Figure 3. TOTEM's Design Mode.

4.2 Explore (and Test) Mode

This mode has not been implemented at the time of writing, so the description that follows presents the characteristics as we intend them in operation. At any time when the model under construction satisfies validation criteria (as described above), the modeller may choose to explore it. Selecting the explore function opens a new window in which the active model is presented in a 'live' version. This means the user can interact with the model to assess its effectiveness.

Initially the explore-mode window shows a replica of the design-mode model. The user may wish to interact with a part of the model, and certainly, if multiple relationships exist (as they do in Figure 1) it will be necessary to indicate which of these is of interest. Clicking on relationship lines will toggle between selected and de-selected allowing relationships to be either included or excluded from the activation. Each entity has a drop down list which can either display its attribute list, or be populated by data instances (retrieved from a database set up for the purpose). Exploration is done by interacting with the relationships and the lists. For example, suppose that the fragment shown in Figure 4 is to be explored and that specimen data files are available. Enabling both relationships (*takes* and *contains*) will result in drop down lists appearing beneath each entity symbol, showing lists of students, courses and modules respectively. Selecting (by single mouse click) a tuple from a populated list makes *that* item the focus, so selecting a particular student will cause his course and its modules to be highlighted automatically, whereas selecting a course causes its constituent modules and all students taking it to be highlighted. This allows simple queries on the test data to be performed navigationally.

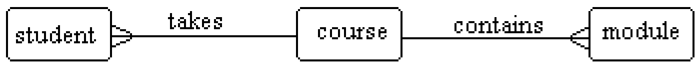


Figure 4. Model Fragment to Illustrate Explore-Mode Concepts.

While a model is being explored, it may be presented in differing ways. For example, attribute names attached to entities may be either shown or hidden; similarly, the user may choose to see either the entire model, or some subset thereof (by selecting and de-selecting relationships). These mechanisms provide *graphical abridgement* as described by Radermacher [20]. Of course, different presentations of the same model may be used concurrently in different windows.

This facility should allow the early detection of errors, as the modeller may quickly establish whether the model permits the correct retrieval of information for typical queries which he foresees. This approach somewhat blurs Norman's distinction between error-detection by self-monitoring and by deleterious outcomes. In our experience, trainee modellers often create models which, despite their attempts at self-monitoring, contain unnoticed errors; these errors often become apparent almost immediately when a database corresponding to the design is implemented, and it is discovered that some data cannot be retrieved (or even stored) as intended. At this stage, the user has a deleterious outcome and is faced with returning to his model to correct it, re-implementing and repopulating the database. Sometimes this cycle must be repeated several times, considerably delaying the final system. However, we expect that the user may make the same discovery of errors considerably sooner by investigating the animated test data available in TOTEM's explore mode, saving the expense of repeated implementations.

5 Video Evaluation Study

A video evaluation of TOTEM was undertaken using the same experimental method reported in Barclay et al. [1]. It was unnecessary to repeat the flip-chart condition on this occasion as the experiment had a different objective, namely to assess the usability of TOTEM against criteria derived from the previous study of pen-and-paper working.

5.1 Subjects and Method

Four volunteers were recruited: L1 and L2 were lecturers in computing (though not teachers of data modelling), S1 was a fourth year undergraduate in computing and S2 was on a conversion postgraduate diploma course in computing. All subjects were presumed to be reasonably competent modellers and familiar with Microsoft Windows conventions. None of the subjects had read our previous paper [1] before taking part (this was important, for example, because prior knowledge that preliminary drafting on paper was of interest would have spoiled spontaneous behaviour). None had taken part in a film study before. Prior to filming, all subjects had been given a) an information sheet outlining the purpose of the study and the limited set of operations they needed to master (see Appendix I) and b) access to the software for familiarisation purposes. The familiarisation was done just by interacting with TOTEM and typically occupied the subjects for between 30 minutes and one hour (there was no help from the researchers, from documentation or from on-line sources).

The subjects were filmed consecutively on the same afternoon, with no opportunity for collaboration. The conditions were the same in each case. As in the earlier study, a split-screen video technique was used, capturing each subject's upper body, face on, from one camera, and merging it with an image of the VDU screen from a second camera. The subjects were given a textual scenario to model (see Appendix II) and asked to give a speaking-aloud protocol, or verbal commentary, as they worked. Clip-on microphones were used to capture the speech. It was made clear to the subjects that we did not want an appraisal of the product they were using per se, but a commentary on the thought processes they were going through in formulating a model — usability issues might well emerge in context and users should explain them as they arose. On the table, within easy reach of the modellers, a pen and A4 pad were placed. We made no reference to these, but provided them to see what would happen. Each subject was filmed for between 20 and 30 minutes. During filming the second author sat beside the modeller. His rôle was to give each subject a standard introduction, provide someone for the modeller to aim his commentary at, and to intervene occasionally should the monologue dry up or an opportunity arise to elicit a little more detail than was forthcoming spontaneously. During filming the first author liaised with the film crew and took notes on the interactions; these formed the framework for subsequent video analysis.

5.2 Results

The most interesting aspect of the observations was not something we set out to investigate. Four modellers were recruited who were expected to be broadly similar in ability and comparable with subjects ST1 and FC of the previous study [1]. Aware that expert users are preferable to target users at the early stages of the usability evaluation of teaching tools [2], we sought volunteers among staff and senior students, expecting to obtain a reasonably homogeneous group of competent modellers whose interactions would allow us to concentrate on the operational effectiveness of the software. In fact, though the usability problems discovered were valuable and have led to immediate improvements, the unexpected bonus was that the individual differences between the modellers proved equally illuminating. By serendipity, we captured four archetypal profiles which are summarised in Table 2.

We had not set out to contrast novice and expert performance in this particular video study, but one of the rewards of behavioural research is that the subjects often present data that is unexpected yet relevant! In recruiting S2, we had unwittingly selected a rather weak student. S2's problems using TOTEM were severe, but arose a) because he did not share the Windows culture we had assumed and b) because his lack of general computer literacy caused him considerable confusion (for example, he thought he may have inadvertently sent an e-mail using the post-it facility during the familiarisation session). This subject did use paper, but wrote only three words on it, the names of three candidate entities. S2's problems have reminded us that the word 'intuitive' as applied to human-computer interfaces is seldom appropriate without qualification: much shared culture is a prerequisite to finding an interface easy to use. S2 was an individual who would have required training on the operation of Windows-based products in general and TOTEM in particular before he could have attempted the task with any degree of comfort. Despite some of his problems being idiosyncratic, S2 still provided us with useful usability feedback and several of the problems he experienced were also seen in others.

S1 had acquired the skills that S2 lacked. His difficulties were at the modelling level rather than the operational level. S1's interaction was characterised by long pauses as he focused deeply on individual elements and their naming. The model that he produced contained no key attributes, showing that answering the sample queries was not uppermost in his mind and that he was not adept at visualising navigation between 'tables'. He neglected to distinguish between item types (hedge trimmers) and instances of these items (multiple hedge trimmers in stock). He did establish a generic entity 'person' to cover trade and general public customers, making 'trade account' an entity (though incorrectly associating multiple trade accounts with any one 'person'). There was no concept in S1's model of long-term record-keeping, for example for accounting purposes, but the scenario did not explicitly mention this. S1 only encountered one minor usability problem with TOTEM during his session and he did not use paper for drafting. This is a typical profile of the novice modeller for whom TOTEM is intended.

Subject	**Interaction classification**	**Characteristics**
L1	Expert performance (data modelling & computer operation)	Pattern re-use, continual attention to system requirements, little time spent on recognition, good at identifying subsystems and boundaries. Very few usability problems and not interrupted by them. Thoroughly in control.
L2	Expert performance (interface evaluation & computer operation)	Set out to evaluate TOTEM more than to model. Prior experience as software tool builder and user informed expectations, comments and criticisms. Thoroughly in control.
S1	Novice performance (data modelling) Expert performance (computer operation)	Cognitively effortful, tended to get bogged down in trying to get one small part of model 'right'. Not displaying the 'surfacing and diving' behaviour characteristic of experts.
S2	Novice performance (computer operation & data modelling)	Operationally effortful, had such difficulties that he was unable to engage properly with the modelling task, though the modelling also looked weak.

Table 2. Summary of the Performances of the four Video'd Subjects.

L2, who was a highly skilled and innovative computer user chose, nonetheless, to draft his model on paper, spending about 8 minutes on this activity before beginning to reproduce his diagram on the screen. Most of the thinking was done at the paper stage. Once using TOTEM, L2 launched into a usability evaluation of the interface drawing on his considerable experience of other tools and interaction styles. It was clear from L2's E-R model that he was not an expert data modeller (for example 'warranty expiry date' and 'account limit' were modelled as entities); perhaps this contributed to his early reliance on paper. L2 provided us with a third very distinct profile, that of the expert user, experienced in software design and evaluation and able to draw on experience of other related tools and interface design standards. L2's performance showed none of the hallmarks of the expert data modeller, but his interaction yielded more usability defects and ideas for future development combined (14) than any of the other subjects.

L1's performance, from the outset, exhibited expert behaviour as described in sections 1 and 2 above. The DIY scenario given as the modelling problem was based on the library scenario used in our previous video study [1], though that was not apparent at surface level. Immediately after reading the scenario L1 said, "I'm trying to see if it's like things I've seen before and it does seem to be like a library". He then spent a couple of minutes deciding how he could split the problem into known chunks (pre-stored patterns) and others. As he began to model he continued, "I'm trying to make this look like a library system, which I'm quite familiar with, so if we map....... a (book) title on to a type of equipment, a particular model of sander for example...", mapping objects of current interest on to the entities from his generic lending/borrowing domain. As he modelled, and disregarding our instructions to ignore features such as colour (see Appendix I), L1 used colour to distinguish subsystems (accounts system versus hire system), specialisations ('equipment item' versus 'equipment model') and semantic category (people-related entities like 'customer' versus abstract entities like 'loan'). Thus the colour feature enabled him to make explicit to us a richness in his thinking as it unfolded, in a way that is not ordinarily available. L1 perceived the DIY problem to be made up of two generic domains, library and accounting, with some specific details such as three hire rates and the warranty details to be added. Apart from his use of generic domains, L1's performance was unique among the four we watched, in the extent to which he consulted and re-consulted the specimen queries (typical user requirements). His modelling was punctuated by returning to the queries to make sure that they could be satisfied, and if not to make changes. He was performing mentally and by quick inspection the process of model checking which TOTEM's explore mode will help novices through. S1 considered drafting his model on paper but decided to try TOTEM, remarking that he would probably return to paper, in fact he never did. He commented afterwards that he had been impressed by how easy it had been to work without paper.

Just as Batra and Sein had suggested [10] (see section 2 above), L1 (unlike S1) spent little time on recognition and alternated with facility between surfacing to check the requirements and visualise the model in operation and deep-diving to enter a Boolean flag or add a foreign key. Interestingly, S1 is not an expert data modeller, and has little occasion to use E-R modelling in his work, but he was expert in *the way he went about the task.* The aim of the TOTEM project is to build a tool that will help to transform modellers like S1 into modellers like L1.

To summarise, the video study was valuable in two ways: it gave us four case studies representing contrasting user types as shown in Table 2; it also yielded usability data on the core functions of TOTEM's design mode. The user profiles have given us a deeper understanding of the differences between novice and expert performance and these insights will inform future work. During the course of the video study we compiled a list of 8 changes to TOTEM to be made immediately: these were bugs or inconsistencies that demanded attention and would be quick to fix. A further 7 improvements were identified for attention thereafter, including a review of the operation of the attribute dialog box (see Figure 3) which was the interface object that caused most problems. The video study and subsequent feedback from the testers resulted in a further list of 11 ideas for future enhancements, which we will consider as work progresses. The video study has alerted us to most of TOTEM's current shortcomings and enabled us to attend to them before the system reaches the classroom. The study has also provided evidence that progress has been made towards a modelling environment that begins to approach the flexibility of pen and paper in its ease of use.

6 Summary and Outline of Further Work

Apart from its value to database designers, E-R modelling provides a useful graphical medium which can facilitate discussions between analysts and clients. Acquisition of data modelling skills is considered to be an important part of the training of any software designer, and essential to the development of all but the most trivial database systems. However, modelling expertise is hard to impart and current computer-based modelling environments tend to cater for the capture and documentation of models rather than their design and evaluation. We have also observed that, hitherto, the teaching of modelling, as represented by published guidelines and standard texts, has over-relied on deterministic methods, leading to the identification of a 'correct' model at the expense of more openness to the comparative merits of alternative feasible solutions. The creative modelling approach [3,9] has been under-emphasised. We would suggest that the rule-following and creative approaches to modelling outlined here tend to foster surface- and deep-level learning respectively during data modelling [22].

This paper reports progress that has been made towards realising the key recommendations of Barclay et al. [1]. In tackling those recommendations we have embraced the need for a good teaching tool and particularly for an environment in which creative modelling is encouraged. The resulting system, TOTEM, has been described. The video study suggests that it is a good step towards meeting our usability objectives, but of course we anticipate that forthcoming use in the classroom will result in further improvements both to the current interface and to the functionality. In terms of work on the software the medium-term goals include:

- Completing the validation function in design mode.
- Implementing of explore (and test) mode.
- Providing a means of automatically supplying test data.
- Exploring the scalability of the prototype system (number of windows and size of models).
- Investigating the utility and feasibility of being able to select and explore a subset of the current design-mode model.

An evaluation of TOTEM's design mode will take place next academic session, with three main objectives: to assess the usability of the software with novice modellers, to introduce students to co-operative problem-solving in an on-line environment and to try to identify ways in which explore mode might assist in the transition from novice to expert modeller. Once explore mode is available for evaluation, it too will be the subject of a usability analysis and the resulting system will be used to assess the effects of teaching with TOTEM on the quality of students' data models.

Our thesis remains that only with an understanding of the modelling process itself can a good modelling tool be achieved. Using TOTEM with students will increase our understanding of modelling-in-action and that understanding will in turn feed into future versions of the tool. In the coming academic session we plan to:

- Evaluate the usability of TOTEM compared with modelling using pen and paper.
- Assess the effect of TOTEM on the range and quality of students' solutions to data modelling problems.
- Devise teaching examples and strategies that take advantage of TOTEM's capabilities and promote a creative approach to data modelling.
- Test TOTEM as a lecturing aid.

Shanks and Simsion [9] observed the evident conflict between the creative modelling philosophy which stresses non-deterministic aspects of modelling and the interests of CASE tool vendors whose products are marketed on the strength of substantially automating large parts of the design and development process. They predicted that in the foreseeable future

> ...the value of developing a specification for Creative Data Modelling support is likely to lie in greater insight into the nature of the data modelling process, as performed by human modellers. (pp 302-304).

Understanding more about cognitive aspects of the data modelling process is our primary interest. From the small scale video study presented in Section 5 we gained a deeper understanding of the distinguishing hallmarks of novice and expert performance. Even in its present form TOTEM is beginning to illuminate modelling issues. Our initial objective in undertaking this work, achieving core functionality which approaches pen and paper in its ease of use, is well advanced. Phase two, adding explore mode, presents many interesting challenges both for design and implementation.

Acknowledgements

We would like to thank Julian Miller for his mathematical input to TOTEM, James Blair and Brian Smith for shooting and editing the video film, subjects L1, L2, S1 and S2 for taking part in the video study, Lindsay Ewing and Alan Francis for their evaluation of various earlier versions of TOTEM, Gillian Ferrier for her background work on the Delphi implementation and our anonymous referees for their helpful criticisms and encouraging comments.

7 References

1. Barclay, P., Crerar, A., & Davidson, K. Interfaces to data models: taking a step backwards. Proceedings of the 2nd International workshop on user interfaces to databases, 1994; 306–326. Springer-Verlag, London.
2. Crerar, A., & Davidson, K. Teaching and learning through CAL development: an HCI perspective. Paper presented at XXIX Annual Int. Conf. of the Association for Educational Training and Technology, Napier University, April 1994.
3. Simsion, G. C. Creative data modelling – encouraging innovation in data design. Proceedings of the 10th International Conference on the Entity-Relationship Approach, 1991; 111–128.
4. Chase, W. G., & Simon, H. A. Perception of chess. Cognitive Psychology, 1973; 4:55–81.
5. Lawson, B. R. How architects think. Architecture Press, London 1980.
6. Gamma, E., Helm, R., Johnson, R. & Vlissides, J. Design patterns: elements of reusable object-oriented software. Addison-Wesley, 1994.
7. Batra, D. & Davis, J. G. Conceptual data modelling in database design: similarities and differences between expert and novice designers. International Journal of Man-Machine Studies, 1992; 37: 83–110.
8. Alavi & Weiss. Managing the risks associated with end-user computing. Journal of Management Information Systems 1985; 2: 5–20.
9. Shanks, G. & Simsion, G. Automated support for creative database design. Proceedings of the 3rd Australian Database Conference, 1992; 293–305.
10. Batra, D., & Sein, K. Improving conceptual database design through feedback. International Journal of Human-Computer Studies, 1994; 40:653–676.
11. Sawyer, P. & Mariani, J. Database systems: challenges and opportunities for graphical HCI. Interacting with Computers, 1995; 7:273-303.
12. Microsoft. Visual Basic product documentation.
13. Borland. Delphi product documentation.
14. Chen, P. P. The entity-relationship model — toward a unified view of data. ACM Transactions on Database Systems, 1976; 1: 9–36.
15. Batra and Anthony. Effects of data model and task characteristics on designer performance: a laboratory study. International Journal of Human Computer Studies, 1994; 41:481–508.
16. Norman, D. A. Design rules based on analyses of human error. Communications of the ACM, 1983; 26:254–258.
17. Song, I., & Froehlich, K. Entity-relationship modeling: a practical how-to guide. IEE Potentials, Dec ’94/Jan ’95; 29–34.
18. Simsion, G. C. Data modeling essentials: analysis, design and innovation International Thomson Computer Press, Boston, MA, 1994.
19. De Bono, E. The use of lateral thinking. Pelican Books, Harmondsworth, Middlesex, 1971.
20. Radermacher, K. An extensible graphical programming environment for semantic modelling. In Richard Cooper (ed.), Proceedings of the 1st International Workshop on Interfaces to Database Systems. Springer-Verlag, London, 1992, pp 353–376. (Workshops in Computing Series).
21. Jarvenpaa, S. L., & Machesky, J. L. 1989. Data analysis and learning: an experimental study of data modeling tools. International Journal of Man-Machine Studies, 1989; 31:367–391.
22. Entwistle, N. A model of the teaching-learning process. In Richardson, J. T. E., Eysenck, M. W., & Piper, D. W. (Eds) Student learning: research in education and cognitive psychology. Open University Press, Milton Keynes, 1987, pp 13-28.

8 Appendix I

TOTEM video study — notes to participants

Thank you for agreeing to take part in a video study to evaluate the usability of TOTEM, a new data modelling tool. Please turn up at the TV studio, Room E23 Merchiston, at .... on Monday 17th June. We may be filming when you arrive, so please wait outside the studio door, we won't keep you waiting long.

Further briefing will be given on Monday, but the essential thing to know is that we are interested in the usability of TOTEM as a tool and not in the quality of the data model you produce. We will be giving you a short textual scenario to model (rather than an E-R diagram to copy) because we are interested in TOTEM's effectiveness during the *process* of modelling, i.e. when problem-solving is underway and the cognitive load is greater.

Please copy TOTEM on to your C: drive and familiarise yourself with performing the fundamental operations listed in Table 1. The left column shows the object to be created or operated on (entity and binary relationship), and the associated row shows the operations you need to be familiar with. You will find that TOTEM supports many of these operations in alternative ways (for example, entities may be created from the menu, icon bar or by context-sensitive pop-up menu on right mouse button). TOTEM supports other operations such as changing the colour of entities and relationship lines, but please ignore additional facilities for this exercise.

Entity	Create	Delete	Move	Add/Edit entity name	Add/Edit/Delete attribute name and type. Mark attribute(s) as primary key.
Binary Relationship	Create	Delete	Add/Edit relationship name	Add/Delete crow's feet	

Table 1. It is essential that you are familiar with these operations prior to filming.

In addition to entities and attributes, TOTEM also offers post-its. These allow users to make notes about a model, perhaps to state assumptions or leave ad hoc annotations to remind themselves about issues to resolve. These will probably be more useful in the MDI version which supports multiple views. You are testing the SDI version of the software (one E-R model at a time) so may not find the post-its useful. Use of post-its in this exercise is optional. Use them only if you find them helpful as a scratch-pad during the modelling process. Table 2 shows the operations associated with post-its.

Post-it	Create	Delete	Move	Re-size	Add/Edit text	Hide	Show

Table 2. Operations available on post-its. *Use of post-its is optional.*

9 Appendix II

The scenario used for the video study.

Produce an entity-relationship model to capture the following scenario.

U-Do, a DIY hire business would like a database to manage its operation more effectively. U-Do hires out a diverse range of equipment (such as floor-sanders, carpet-shampooers, hedge-trimmers, mountain bikes, barbecues) to tradesmen and members of the general public.

Tradesmen can open an account, with an agreed upper limit and do not need to leave a deposit for each item hired. Members of the public are required to leave a deposit and must pay for the hire in full on returning the equipment.

There are daily, 3-day and weekly rates for each item hired; these are the same for all customers.

It is crucial that all items of equipment are kept in good working order and that periods out of service are minimised. Much of the stock is still under manufacturer's warranty and will be fixed free of charge by a designated agent, so it is important to know if the warranty is still in force.

U-Do buys from a range of suppliers and does not always use the same supplier for a particular item type (so all hedge-trimmers may not be from the same source).

Typical on-line queries might be:

Is there a sander available now?
When will there be a hedge trimmer available?
What is the 3-day rate on a large ladder?
Is a particular bike still under warranty?
Who hired the cement mixer?