Entity-relationship and object-oriented data modeling—an experimental comparison of design quality

Peretz Shoval, Sagit Shiran

Information Systems Program, Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Received 7 February 1996; revised manuscript received 31 May 1996; accepted 31 July 1996

Abstract

We compare EER and OO data models from the point of view of design quality. Quality is measured in terms of (a) correctness of the conceptual schemas being designed, (b) time to complete the design task, and (c) designers' preferences of the models. Result of an experimental comparison of the two models reveal that the EER model surpasses the OO model for designing unary and ternary relationships, it takes less time to design EER schemas, and the EER model is preferred by designers. We conclude that even if the objective is to implement an OO database schema, the recommended procedure is to: (1) create an EER conceptual scheme, (2) map it to an OO schema, and augment the target schema with behavioral constructs that are unique to the OO approach.

Keywords: Data modeling; Entity relationship model; Object oriented model; Database design; Conceptual schema; Experimentation; Schema transformation; Quality of design

1. Introduction

The object oriented (OO) approach has spread into various areas of computing that include not only programming, but also systems analysis and design, database management, among others. There is no doubts about the advantages of the OO approach for programming, in that it supports software reuse, information hiding (encapsulation), inheritance, and other features of good software. However, the superiority of OO approach in earlier stages of software development, i.e., system analysis (specification) and database conceptual design, has not as yet been proven. In any case, before abandoning other well established and commonly used methods, the new methods should be examined for various types of software development activities.

This study concentrates on one aspect of this issue: database conceptual design (also termed data modeling). This activity is carried out at an early stage of system development, and is aimed at creating a conceptual schema, i.e., a data model of reality. The conceptual schema is
usually represented in a diagrammatic form, as it serves as a communication tool between developers and users. Once approved by users (as a proper representation of reality), the conceptual schema is converted into a specific database schema, depending on the data model and DBMS that is used for implementation. (This conversion is usually a simple matter, being an algorithmic, automatic process.) The major problem, however, is to create a good conceptual schema that is semantically correct, complete, easy to use, and comprehensible.

For many years, Entity-Relationship (ER), with its many extensions (generally termed EER) has been the most widely used model for conceptual design. The basic constructs of this model consist of entities, attributes, and relationships. Recently, the OO model, which also has many variations, has been threatening to replace the EER model. In addition to static (structural) constructs of the model whose representation of data structure is somewhat equivalent to EER representation (e.g., object classes considered equivalent to entity and relationship types) – the OO approach models system behavior through ‘methods’ (procedures) that are attached to the object classes, with messages passing among them. While this extra capability, unlike EER, provides more than just a data structure model, it does not necessarily mean that EER should be replaced by OO, since EER may indeed be ‘better’ than OO for modeling the data structure. Taking into consideration that an EER schema can easily be mapped to an OO schema (see, for example, Gogola et al. [9], Kornatzky and Shoval [12], Nachouki et al. [14], and Narasimhan et al. [15]), it may be worthwhile to begin the process of data modeling with the design of an EER conceptual schema. This would then be converted to an OO schema by an appropriate mapping algorithm (if the objective is to implement an OO DBMS), and as a final step added the behavioral constructs.

Such an ‘indirect’ strategy for creating an OO schema could be considered when the EER model is regarded a ‘better’ than the OO model. This aspect of data structure modeling is the concern of our present study. ‘Better’ can be judged from various perspectives: (a) ease of use, i.e., how easy is it to learn and use a model or method; (b) quality of the design, i.e., how accurate and complete is the conceptual schema that is being designed, as compared to the reality being modeled; (c) comprehension, i.e., how easy is it for users to understand the schema.

We focused on design quality comparative experiment involving the two models. We measured the correctness of schemas, the time it takes to complete the design tasks, and the designers’ preferences of the models. Our results revealed that while there are no significant differences between the two models for correctness of modeling entities/objects, attributes, supertype-subtype (inheritance) relationships and binary relationships, EER is better when modeling more complex constructs like ternary and unary relationships. In addition, the experiment showed that it takes less time to complete the task with EER, and that designers prefer working with the EER model.

2. Related studies

Various studies on the evaluation and comparison of data models and methods have been conducted in the past one and a half decades. The earlier studies (from the late seventies and early eighties) mainly compared record-based models (i.e., hierarchical, network and relation-
Later on, EER became the most frequently studied model. EER has been compared with relational and other record-based models, as well as with other semantic models (see, for example, Batra et al. [4] and Juhn and Naumann [10]). Some studies compared the data models from a designer perspective in an attempt to find out which yields better, more accurate and precise schemas, which model designers prefer to work with, and which requires less time to complete the design task. Other studies compared the query languages of the data models in order to determine the languages by which the designers compose queries with greater accuracy and speed. Yet, other studies took an end-user perspective, attempting to determine ease of comprehension. A survey of earlier studies involving controlled experimentation and comparison of data models can be found in Batra and Shrinivasan [5]. Although the results of these studies are not always clear-cut or consistent, there is a tendency to agree that EER is superior to other, record-based and conceptual models. One of their general conclusions was that usability of data models should be evaluated by their ability to model relationships, a point that is stressed in our study.

Recently we have begun to find studies which evaluate the OO model on the basis of controlled experiments. For example, Agrawal and Sinha [2], who examined the influence of designers’ experience in functional analysis and task characteristics on quality of OO schema design, found out that the OO model does not always lead to good results; the quality of the result depends on task characteristics. Palvya et al. [16] compared OO, ER and DSD models from an end-user perspective, to see which schema is more comprehensible, and concluded that OO schemas are superior in this respect. However, they measured comprehension on overall terms only, without examining specific constructs of the models (e.g. specific relationship types), and used small (perhaps even trivial) schemas. Shoval and Frumermann [17] also compared EER and OO models with respect to user comprehension, but with more complex schemas. They examined comprehension of various constructs of the models, including different types of relationships. While they found no significant differences in comprehension of entities/objects, attributes and binary relationships, they found that EER schemas are more comprehensible for ternary relationships because EER represents relationships with a specific (diamond) symbol that connects the involved entities. In contrast, all object classes in OO – including those that represent ternary relationships – appear the same (rectangles), thus perhaps ‘hiding’ semantic information from users.

Bock and Ryan [6] have reported on a comparison of OO and EER models from a designer perspective. They examined correctness of design for eight types of constructs: objects/entities, attribute identifiers, inheritance relationships, unary 1:1 relationships, binary 1:n and m:n relationships, and ternary m:n:1 and m:n:p relationships. Their experiment involved two groups of students who studied and then experimented with one of the two models. Correctness of schemas was measured according to a grading scheme developed by Batra et al. [4]. Their results indicated that EER is better when representing attribute identifiers, unary 1:1 and binary m:n relationships, while there are no significant differences for the other dimensions. Also, they found no difference in time to complete the tasks. Although these results are interesting, in our opinion they have limited external validity. The OO model they used (as described in Kroenke [13]) does not show reference links between related object classes, and represents relationships only by reference attributes. This limits the ability to represent and understand relationships. For example, it is not possible to distinguish
between a class that represents a binary relationship that is related to another class, from a class that represents a ternary relationship. Similarly, their model does not correctly represent a class that is an \( m:n:1 \) relationship. Also, this model limits the designer to representing different relationship types in only one way, e.g., a binary \( m:n \) relationship with attributes must be represented as a separate class.

We also compared OO and EER models for quality of design, but in our OO model relationships are represented both with reference attributes and links between the involved classes. Moreover, we specify reference attributes so that different types of classes are represented correctly. Thus binary and ternary relationships are distinguished. In the next section we briefly describe and exemplify the specific EER and OO models we compared.

3. The EER and OO models

The objective of this section is not to provide a tutorial on the two models, but to overview and exemplify them. Since each of the models has many variants, we confine ourselves to two specific types, that are sufficiently general to validate the results of the comparison.

3.1. The EER model

As EER is more 'standard' than OO, we provide only a brief overview. Fig. 1 shows an example of an EER diagram which includes all our constructs of the model. The diagrammatic notation is similar to that in Teorey et al. [19] and Elmasri and Navathe [8]. We distinguish between entity (rectangle), attribute (circle) and relationship (diamond) types. (For brevity, we will omit 'type' from the construct names.) An identifying attribute is underlined, and a multi-valued attribute has a dual circle. Cardinalities of relationships are signified by both the \( m:n \) notation (e.g., \( 1:n \)) and by coloring the 'n' edges of the diamond. Supertype-Subtype hierarchies between entities are signified by arrows linking the sub-entities to the super-entity. We do not show 'weak entities' and constraints, e.g., total-role, as they have no equivalents in the OO model.

3.2. The OO model

The OO model is still evolving and as yet no standard has been defined. Moreover, even within a specific OO model there are variations; the same reality can be represented in many different forms (see the discussion, for example, in Spaccapietra et al. [18]). The model that we use is based on O2 [7] and ODE [1]. By extending the model we were able to show all the details of the OO schema on diagram, as would be expected at this conceptual design stage. Special attention was given to the representation of relationships between object classes. Since the focus of this study is on modeling data and not system behavior, methods are not included in the OO model.

An example of an OO schema is shown in Fig. 2. It represents the same reality as the EER diagram in Fig. 1. An object class (rectangle) has attributes. (For brevity, we will omit 'class' and simply say 'object'.) An attribute may have an atomic value (e.g., attribute state of object
An attribute may refer to another object. This is signified by writing the referenced object name in brackets next to the respective attribute name. For example, mayor of City is a reference-attribute to Person. This actually signifies a relationship between the objects. A relationship is bi-directional, signified in two complementary ways: (a) reference-attributes are included in each of the involved objects (therefore, in the above example, in addition to the attribute mayor in object City, there is the attribute mayor-of [City] in Person); (b) a link connects the two objects, with an indication of the relationship cardinality (so the above relationship is marked 1:1). Note that in the case of an 1:n relationship between two objects, the object in the '1' side has a 'set' reference attribute to the other object (e.g., set inhabitants [Person] in City), and in case of an m:n relationship both objects have set reference attributes. An attribute may be a tuple consisting of multiple (value or reference) attributes, and signified by { } brackets. For example, City has a set attribute producers which is a tuple of two: a reference attribute to Producer and a valued attribute number_of_plants. (More examples of tuple attributes can be seen on the diagram.)

Relationships are represented as follows: a binary-relationship is represented by two objects
with mutual reference attributes, and a link indicating the cardinality. (An alternative representation could omit reference attributes in both objects). An \( m:n \) relationship can also be represented as a separate object, with reference attributes to the related objects. In the example (Fig. 2) there is an \( m:n \) relationship between City and Producer. Each of the involved objects has set-valued reference attributes, where each attribute is a tuple. For example, in Producer we have: set where_producing\{[City], number_of_plants\}.

When there is a ternary- or higher order relationship, it is represented as a separate object and linked to the other objects involved in the relationship. The appropriate reference attributes in all objects are included. In our example, we had to store facts about agents selling vehicles to persons. Therefore, we used the objects Person, Agent and Vehicle, and for the relationship we created the object Sales. Each of the three original objects has an appropriate set reference attribute to Sales, while Sales has a tuple consisting three reference attributes to those objects. In addition, Sales has atomic valued attributes price ($) and date_of_sale. The use of tuple of reference attributes in an object that represents a relationship can determine the degree of the relationship represented by that object (e.g., whether it is binary or ternary).

Finally, object hierarchies were signified by links from the subtypes to the supertypes, similar to the EER notation. In both models, the subtype objects (entities) inherited the attributes and relationships of the supertype, and included only their own specific attributes and relationships. Hence, there is no visible difference between the two models with respect to the representation of object/entity hierarchies.
4. Objectives and hypotheses

We compared EER and OO data models or design methods from a designer perspective, in order to find out which of the two models provides better schemas. Quality was examined mainly in terms of correctness of the various constructs of the two models. Additional criteria included time to complete design tasks, and designer preferences of the models. The comparison is important both from theoretical and practical perspectives: If the OO should prove to be better, perhaps OO should replace EER design, at least when the objective is to implement an OO DBMS. If, however, the EER design is better, we would decide to use the EER for conceptual data modeling even if the target is to implement an OO DBMS. The reasoning behind this is that an EER schema can easily be mapped to an OO schema, and augmented with behavioral constructs (e.g. methods) later on.

We defined nine null hypotheses with respect to correctness of modeling nine different constructs of the two models, as follows:

1. There is no difference in correctness of modeling entity types or object classes.
2. There is no difference in correctness of modeling attributes of entity types or object classes.
3. There is no difference in correctness of modeling inheritance relationships.
4. There is no difference in correctness of modeling unary 1:1 relationships.
5. There is no difference in correctness of modeling binary 1:1 relationships.
6. There is no difference in correctness of modeling binary 1:n relationships.
7. There is no difference in correctness of modeling binary m:n relationships.
8. There is no difference in correctness of modeling ternary m:n:1 relationships.
9. There is no difference in correctness of modeling ternary m:n:p relationships.

In addition, we defined the following hypotheses with respect to time and preference:

10. There is no difference in time to complete design tasks.
11. There is no difference in designer preferences of the models.

For each of the above null hypotheses there is an opposite hypotheses saying that there is a difference.

5. The experiment

The experimental design was as follows: the dependent variables were, according to the above hypotheses: (a) correctness of design, (b) time to complete task, and (c) designer preference. The independent variables were the EER and OO models, and the controlled variables were the design tasks and subjects (designers).

We defined two design tasks, both similar in size and complexity, i.e., involving the same types and numbers of various constructs. We created a narrative description for each task, on which basis each subject was asked to design a conceptual schema diagram using one of the two models. The two tasks are presented in Appendix A. Figs. 3 and 4 show the solutions for Task 1 in the form of EER and OO diagrams. Figures 5 and 6 show the solutions for Task 2. (Note that these are not the only possible solutions. For example, in the case of an OO model,
relationships are represented as two objects. Another possibility could be to create a separate object for the relationship.)

The subjects included 44 students, all majoring in Information Systems, with a similar background and course of studies. They were at an advanced stage in the program, after having participated in both Systems Analysis and Design, and Database Management courses. All subjects were trained to use the two models and the same amount of time – six hours – was devoted to study and practice for each of the models. To avoid bias, the same instructor taught all the students. In order to motivate the subjects, they were told in advance that their performance in the experiment would be part of their final grade.

Subjects were randomly divided into two groups, such that subjects in one group designed the first task with OO and the second task with EER. Subjects in the other group switched models and tasks. In addition, we changed the order of model usage, such that in each group, half of the subjects started with EER and continued with OO, the others working in the opposite order. They were allowed to use class notes and examples during the experiment. Each task was distributed separately, and the starting and ending times of each task were recorded. At the end, each participant was given a short questionnaire to express his preference. The students were asked to grade the ability of each model to design a conceptual schema according to a 7-point scale. In addition, according to another 7-point scale, they were asked to grade their preference, ranging from absolute preference for OO to absolute preference for EER. Three open-ended questions concerned the models' strengths and weaknesses, and their preferences.

6. Results

Correctness of each schema was measured according to the evaluation scheme described in Batra et al. [4]. Each construct type (in each model) was scored separately by subtracting a certain number of points for each error type, and distinguishing between minor, medium and major errors. After scoring each type of construct in each schema, we averaged the scores of all subjects (tasks) on each construct type, for each of the two models separately. The results are summarized in Table 1 and discussed herein.

6.1. Correctness of design (hypotheses 1–9)

We used the $T$-test to measure the difference of mean scores for each of the nine types of constructs, as shown in lines 1–9 of Table 1. ($T$-test is a parametric test that is used to test hypotheses about the difference between two samples that are taken from a normally distributed population.) The last columns of the table indicate the significance of the results. The “which?” column indicates if the results are significant (at $\alpha = 0.05$) and which model is favored. To verify these results we also conducted the Wilcoxon signed rank test, a non-parametric test which does not assume normal distribution of the population. (It only assumes that the populations being sampled have identical continuous distributions.) The results of these tests fully support the results of the $T$-tests for all hypotheses (data not shown).

As shown in the table, we found no significant difference for six of the constructs, namely
entity types or object classes (1), attributes (2), inheritance, or supertype-subtype hierarchies (3), and the three types of binary relationships (5–7). These results are explained by the similarity of the models when dealing with these constructs. The slight differences in the symbols that represent entity types or object classes, as well as inheritance links, turned out to be insignificant for design quality. Although binary relationships are represented differently in the two models, the difference in correctness of the designs is insignificant. Because it is simple to understand and identify these relationships, the choice of symbols is irrelevant. This is evidenced by the high scores on those constructs – above or close to 90 (with the exception of m:n relationships). In other words, there were relatively few design errors. Interestingly, in both models the scores for 1:1 relationships were higher than those for 1:n relationships, which, in turn, were higher than those for m:n relationships. These consistent results strengthen the validity of the results.

Contrary to the above, we found significant differences for unary (4) and ternary (8 and 9) relationships; all favored the EER model. In order to explain why EER modeling turned out to be superior for these more complex types of relationships, we refer to Batra [3], who proposed a framework for classification of errors in data modeling. His classification consists of six error types:

(a) Syntax errors: For example, a key is not specified.
(b) Abstraction: inappropriate mapping from reality to representation; for example, a binary relationship in the real world is mapped to a unary relationship in the model.
(c) Simplification: a complex situation is described simplistically but erroneously; for example, a ternary-relationship is represented as two or more binary relationships.

dentity types or object classes (1), attributes (2), inheritance, or supertype-subtype hierarchies (3), and the three types of binary relationships (5–7). These results are explained by the similarity of the models when dealing with these constructs. The slight differences in the symbols that represent entity types or object classes, as well as inheritance links, turned out to be insignificant for design quality. Although binary relationships are represented differently in the two models, the difference in correctness of the designs is insignificant. Because it is simple to understand and identify these relationships, the choice of symbols is irrelevant. This is evidenced by the high scores on those constructs – above or close to 90 (with the exception of m:n relationships). In other words, there were relatively few design errors. Interestingly, in both models the scores for 1:1 relationships were higher than those for 1:n relationships, which, in turn, were higher than those for m:n relationships. These consistent results strengthen the validity of the results.

Contrary to the above, we found significant differences for unary (4) and ternary (8 and 9) relationships; all favored the EER model. In order to explain why EER modeling turned out to be superior for these more complex types of relationships, we refer to Batra [3], who proposed a framework for classification of errors in data modeling. His classification consists of six error types:

(a) Syntax errors: For example, a key is not specified.
(b) Abstraction: inappropriate mapping from reality to representation; for example, a binary relationship in the real world is mapped to a unary relationship in the model.
(c) Simplification: a complex situation is described simplistically but erroneously; for example, a ternary-relationship is represented as two or more binary relationships.
(d) Overload: similar to simplification, as both involve a complex situation. However, here the designer does recognize the situation as a whole, and does not use a simplifying approach. Since many elements are involved, the analysis process places a heavy burden on the memory of the designer, who may have to infer some elements. Failure to consider all elements may lead to incomplete or erroneous representation; for example, incorrect modeling of the connectivity of a ternary relationship.

(e) Convergence: inconsistency between the real world and model representation caused by the possibility that similar situations in the real world lead to dissimilar representations in the model. The confusion may cause errors. For example, a binary-relationship can be represented as two related entities/objects, but an $m:n$ binary-relationship can be represented by three.

(f) Divergence: another type of inconsistency, occurring when dissimilar situations have similar representations in the model. For example, in OO both an entity and a relationship may be represented as an object.

Based on the above framework, we analyze the difference in results between the two models for unary and ternary relationships:

**Unary $1:1$ relationships**

Representation of unary relationships may be complicated, especially in the OO model, because there are different options by which to represent them. For example, the $1:1$ unary ‘marriage’ relationship, which appeared in the experiment, can be represented in OO model as a reference attribute in Employee, or as a separate object (Marriage) related to Employee. As it turned out, in both models many errors occurred: an average score of 88.07% in EER and 70.45% in OO. Both models evidenced errors of simplification (i.e., the relationship was represented as a simple attribute of Employee), but in OO we found more errors in cases where the designers chose to represent the relationship as a separate object (Marriage). Many designers who adopted this solution made an error in specifying a wrong cardinality, denoting $1:1$ rather than $1:2$. We classify this as an overload error.

**Ternary $m:n:p$ relationships**

There were only a few errors in modeling $m:n:p$ relationships with the EER model – an average score of 94.32%. But many errors occurred with OO – actually averaging 76.14%! Typical errors in OO occurred when instead of a separate object, the relationship was represented as (complex) attributes of other objects. The experiment used a ternary relationship between Worker, Project and Skill. Some designers defined a tuple of reference attributes \{[Skill], [Project], work_hours\} in the object Worker. This error type may be classified as convergence (representing the ternary relationship as if it is a binary relationship).

**Ternary $m:n:1$ relationships**

Here too, the EER model was, indeed, better: 85.23% vs. 67.61% to OO! These results indicate that in both models $m:n:1$ relationships are more difficult to model than $m:n:p$ relationships. Designers made several errors of simplification (they represented the situation as two binary relationships, e.g., the relationship between Worker, Project and City was represented by some designers as two binary relationships: one between Worker and Project,
another between Project and City. But in OO more errors occurred for correctly defining the cardinality of the relationship. In EER the \( m:n:p \) and \( m:n:1 \) relationships are represented similarly, and the entity in the ‘1’ side is visible (the other entities are at the ‘n’ sides). However, in OO, \( m:n:p \) and \( m:n:1 \) must be represented differently, because all three objects linked to the relationship object are \( 1:n \) and have set reference attributes. Therefore, in order to distinguish the object in the ‘1’ side of the \( m:n:1 \) relationship, the reference attributes in the ternary-relationship object must be properly specified. In the experiment we had a situation where a worker may work in many cities, but only on one project for each city. To represent this correctly we defined a ternary-relationship object as \textit{Workers in Cities} with two separate reference attributes: (a) tuple assignment \{[Worker], [City]\}, (b) project \{[Project]\}. Another possibility is to attach a 'method' to that object, that will enforce the rule that a worker works on no more than one project in a city. (The method will fire whenever a new object occurrence is added to the database.) At any rate, the tuple of reference attributes \{[Worker], [City], [Project]\} is not a correct representation of this situation. This was a typical error of OO designers, which may have been caused by the convergence problem interacting the overload problem.

Other errors in OO modeling of ternary relationships (not occurring in EER modeling) were caused by creating an incorrect relationship between the two ternary-relationships. For example, the ternary relationship \textit{Workers-Skills-On-Projects} was connected to the object City. We classify this as a divergence error (because a relationship modeled as an object can be related to other objects).

In summary, we concluded that it is easier to model complex relationships with EER because in it uses the same symbol (diamond) in all cases, and cardinalities can be clearly and simply specified on the links. This is opposed to OO, where different options turned out to be misleading, causing convergence and overload errors. We also found more syntax errors in OO, e.g., missing links between related objects, or missing reference attributes in one of the related objects. Of course, these minor errors are irrelevant when compared to the inherent problem in OO to correctly represent the relationship types.

When we note the similarities and differences between our results and the results obtained by Bock and Ryan [6], we see that the OO model is not superior for any dimension. In both studies EER was better than OO for unary 1:1 relationships; but in [6] EER was better for binary \( m:n \) relationships, and in our study EER was better for ternary relationships. The reason our study showed no difference for \( m:n \) relationships is because our OO model represents an \( m:n \) relationship in two alternative ways (two objects with set reference attributes in each, or three objects, the third being a relationship object). Bock and Ryan’s OO model provides only one option (the second). It is possible that our OO designers achieved better results (not different from the results obtained by the EER designers) because of the extra possibility. We have no way of explaining why [6] found no differences when modeling ternary relationships, except to refer the reader to our earlier discussion (Section 2) on how their OO model represents such relationships. At any rate, the results of both experiments are not contradictory.

In addition, it should be noted that our results on correctness of schema design are consistent with earlier results of Shoval and Frumermann [17] for schema comprehension—they found no differences between the same two models for understanding binary relation-
ships, but there was a significant difference that supports the EER model in that ternary relationships are better understood.

6.2. Time for completing the design task (hypotheses 10)

As seen in Table 1, line 10, it takes significantly less time to design EER schemas. In terms of practically, the time factor is of minor importance. What matters is that this result is consistent with and supports the findings on correctness of design, thus strengthening the validity of the experimental results.

6.3. Preference of models by designers (hypotheses 11)

For this dimension as well, EER turned out to be significantly better. This is evidenced both by the scores given to each model independently (row 11a in Table 1), and the relative preference of the two models (row 11b). The average of 2.82 on a 7-point scale, where 1 means absolute preference of EER and 7 - absolute preference of OO, indicated a significant preference of the EER model. Significance was verified with the Wilcoxon signed rank test, the scores being \( Z = -3.907 \), and \( P < 0.0001 \). It should be noted that both models scored relatively high on the scales (row 11a). We do not claim that the OO model is disliked or rejected by designers; but it is clear that designers prefer to model with EER. Again, even if we assume that preference is not as important as performance, the consistent results further validate the experiment.

6.4. Content analysis of designers' remarks

In addition to answering the questions on model preference (hypothesis 11), the designers answered three open-ended questions in respect to the strengths and weaknesses of each model, in addition to an explanation of their preferences. We analyzed the answers using the content analysis technique (Kerlinger [11]). This was done by identifying statements (i.e. opinions) that appeared many times in the answers, and grouping them into main categories. We distinguished between positive (strength) and negative (weakness) statements for each model, and counted the number of statements (i.e. the number of subjects who expressed them) in each category. (Obviously, not all subjects expressed opinions in all categories; some expressed more opinions than others; infrequent opinions (i.e., statements on issues that appeared only a few times) are not shown because of their limited validity). The findings are summarized in Table 2.

An analysis of these statements led us to the following conclusions:

(1) *Ease of use*: More designers felt that the EER model is easier to use than the OO model (17 compared to 8), and many designers (14) said that OO is complicated (but no designer said so about EER).

(2) *Understandability*: Although this experiment did not measure understandability, some designers did relate to it. Their answers reflected no significant preference.

(3) *Relationships*: The number of positive and negative opinions on clarity and understandability of relationships indicated that designers handle relationships better with the EER
Table 2
Analysis of open ended questions

<table>
<thead>
<tr>
<th>Category</th>
<th>Statement (Opinion)</th>
<th>EER</th>
<th>OO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive statements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Ease of use</td>
<td>Model is easy to use</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>(2) Understandability</td>
<td>Schema is easy to understand</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>(3) Relationships</td>
<td>Relationships are clear and understandable</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>(4) Attributes</td>
<td>Marking the attributes is easy</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>(5) Mapping to database schema</td>
<td>Simple to map the diagram to a database schema</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td><strong>Negative statements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Ease of use</td>
<td>The model is complicated to use</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>(2) Understandability</td>
<td>Schema is difficult to understand</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>(3) Relationships</td>
<td>The relationships are not clear</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>(4) Attributes</td>
<td>Marking the attributes is not easy</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>(5) Mapping to database schema</td>
<td>Complicated to map the diagram to a database schema</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

model. This conclusion is in line with the experimental results, which showed that designers represent ternary and unary relationships more correctly with EER (with no difference for binary relationships).

(4) **Attributes**: There were not many opinions on the ease of marking attributes, and from the expressed opinions we cannot detect any preference. (Note that in the experiment we found no difference in quality of designing attributes.)

(5) **Mapping to database schema**: This is another aspect which was not tested in the experiment, but some designers expressed an opinion. Interestingly, 7 designers felt that it is simpler to map an OO diagram to a database schema, and 5 designers claimed that the EER diagram is complicated in this respect. This issue deserves separate investigation (particularly if we take into consideration that mapping from one schema to another is an algorithmic process that can be carried out automatically).

7. **Summary and conclusions**

The results of this comparative experiment reveal that the EER model is superior to the OO model for the following reasons: (a) the target EER schema is more correct when dealing with complex (unary and ternary) relationships, with no significant differences when dealing with other modeling constructs; (b) it takes less time to design an EER schema; and (c) designers prefer to use the EER model. Because our results support the EER model across all these dimensions strengthens the validity of the experiment. Furthermore, these results (on quality of design) are consistent with earlier results on user comprehension of EER and OO schemas, and do not contradict other results on correctness of design.

Therefore, even if the objective is to design and implement an OO schema, within an OO DBMS or any other OO programming environment, we would suggest the following ‘indirect’
strategy: (a) design an EER schema; (b) map it to an equivalent OO schema – using an appropriate mapping algorithm or tool; and (c) augment the OO schema with the necessary behavioral constructs (i.e., ‘methods’ and ‘messages’).

Appendix A: Experimental tasks

Experimental task 1

This task is based on Batra et al. [4], and Bock and Ryan [6], with extensions

Projects Inc., an engineering firm, requires a database to keep track of all employees; their skills, projects and departments. Each employee’s SSN, name and date of birth has to be stored. If an employee is currently married to another employee of Projects Inc., it is required to store the date of marriage and who is married to whom. However, no record of marriage need be maintained if the spouse of an employee is not an employee of the firm. Each employee is given a job title. We are interested in collecting more data which is specific to the

Fig. 3. EER solution to experimental task no. 1.
following types: engineer, worker, secretary and manager. The relevant data to be recorded for engineers is their type of degree (e.g., electrical, mechanical, etc.) and their rank; for workers – their worker ID (a unique number assigned by the form) and number of years of experience; and for secretaries – their typing speed (in lpm) and the languages each secretary speaks. It is required to know the club names each manager belongs to.

An employee belongs to only one department. Each department has a unique name, a budget (in $), one manager (a manager can only head one department) and a number of telephone numbers.

To procure various kinds of equipment, each department deals with many suppliers. A supplier typically supplies equipment to many departments. It is required to store the name and address of each supplier and the date of last meeting between a department and a supplier. (You may assume that each supplier has a unique name.) Each department issues many orders. An order is issued by one department and has an identifying number and a date of issue. An order may include many items (and an item may be ordered in many orders). It is required to store the quantity of item in each order. An item is distinguished by a number and it also has a name and a cost (in $). An order is issued to one supplier (but a supplier supplies many orders).

Fig. 4. OO solution to experimental task no. 1.
Many workers can work on a project. A worker can work on many projects, but can be assigned to only one project in a given city. It is necessary to track the date on which a worker began working on a project in a given city. We are interested in the city name and population for each city. A worker can have many skills (e.g., preparing material requisitions, checking drawings, etc.), but he/she may use only a given set of skills on a particular project. A worker uses each skill that he/she possesses in at least one project. It is necessary to keep track of the number of hours that a worker uses each skill in a project. Each skill is assigned a number. A short description is required to be stored for each skill. Projects are distinguished by project numbers. It is required to store the estimated cost of each project (in $).

Experimental task 2

We want to design a database schema for a hospital. The hospital staff consists of doctors, nurses, secretaries and maintenance workers. For each employee it is required to store the SSN, name and address. If an employee is currently married to another employee, then it is required to store the date of marriage and who is married to whom. However, no record of marriage need be maintained if the spouse of an employee is not an employee of the hospital.
Each doctor in the hospital has a unique number and field of specialty (one or more). It is required to store the professional training (e.g., practical nurse, registered nurse, etc.) for each nurse. Maintenance workers may have several occupations (e.g., cleaning worker, porter, etc.). A secretary can be familiar with different word processors.

Every hospital employee belongs to one department. A department is identified by name. A department is headed by one doctor (who can only head one department). We must know the number of beds (for patients) existing in each department.

For each patient hospitalized it is required to store his/her SSN, name, sex, date of birth and date of hospitalization. A patient is hospitalized in one department. During hospitalization, a patient may be diagnosed for several diseases. Each disease is characterized by a disease code and name. For every diagnosis of disease, the diagnosis date and the diagnosing doctor should be recorded. A diagnosis is made by one doctor.

During the hospitalization, each patient undergoes several tests. (The same test can be administered to a large number of patients), and he/she can receive different drugs. The results of every test a patient takes should be kept. A test is identified by a test code. A test also has a description and cost (in $). Every drug has a drug code, type (e.g., pills, solution,
ointment, etc.) and quantity in each package. A patient may receive more than one drug. It is important to store the number of units taken per day by the patient for each drug.

Drugs are delivered to the hospital in shipments. A shipment arrives from one supplier (a supplier can supply a large number of shipments). Each shipment has an identity number and date of shipment. A shipment may contain many drugs, designated to different departments in the hospital. One drug may be designated to several departments. The quantity of each drug in the shipment designated to each department has to be recorded. The name and address of each supplier should be stored. (You may assume that a supplier is identified by name).

References


Peretz Shoval is Professor of Information systems at Ben-Gurion University, Israel. He earned his Ph.D. in Information systems (1981) from the University of Pittsburgh, where he specialized in expert systems for information retrieval. In 1984 he joined the Dept. of Industrial Engineering & Management at Ben-Gurion University, where he started and is head of the Information systems Engineering Program. Prior to moving to academia he held professional and managerial positions in computer companies and in the IDF. Shoval's research interests include, among others, database design, information systems, analysis and design methods, and information retrieval. He has published numerous papers in such journals as Data & Knowledge Engineering, Information systems, IEEE Trans. on Knowledge and Data Engineering, IEEE Trans. on Software Engineering, Int. J. of Man–Machine Studies, Information & Management, Information Processing & Management, Information & Software Technology, Database Management, Decision Support systems, J. of Intelligent Information Systems, and more. Shoval is the developer of ADISSA methodology and tools for systems analysis and design, and of ADDS system for conceptual and logical database design.

Sagit Shiran received her B.Sc. and M.Sc. degrees in Industrial Engineering, specializing in Information Systems Engineering, from the Ben-Gurion University of the Negev in 1992 and 1995, respectively. She is currently working as a systems analyst in the Computers and Systems Division of Bezeq, the Israeli Telecommunication Corp. Her research interests include system analysis and design, database design and object oriented modeling.