The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment

VIJAYAN SUGUMARAN
Oakland University
and
VEDA C. STOREY
Georgia State University

Database design is difficult because it involves a database designer understanding an application and translating the design requirements into a conceptual model. However, the designer may have little or no knowledge about the application or task for which the database is being designed. This research presents a methodology for supporting database design creation and evaluation that makes use of domain-specific knowledge about an application stored in the form of domain ontologies. The methodology is implemented in a prototype system, the Ontology Management and Database Design Environment. Initial testing of the prototype illustrates that the incorporation and use of ontologies is effective in creating entity-relationship models.

Categories and Subject Descriptors: H.2.1 [Database Management]: Logical Design—Data models

General Terms: Design, Experimentation

Additional Key Words and Phrases: Ontology, conceptual modeling, entity-relationship modeling, database design, integrity constraints, ontology management and database design environment

1. INTRODUCTION

Database design is difficult because it relies heavily on a database designer understanding the users’ database requirements and representing them so...
they accurately capture the real world application that is being modeled. A database designer, however, may not have knowledge of the domain in that is being modeled and, thus must rely on the user to articulate the design requirements. The conceptual modeling phase of database design focuses on building a high-quality representation of selected phenomena in some domain. Database designers generate conceptual models (scripts) using (a) conceptual modeling grammars (e.g., the entity-relationship modeling grammar) and (b) conceptual modeling methods and work within an organizational context [Weber 2002a]. Much progress has been made in developing methodologies and guidelines to assist the designer in conceptual database design, especially in understanding how to apply design constructs [Bodart et al. 2002; Weber 1996; Shanks et al. 2002; Siau et al. 1997; Parsons and Wand, 2000]. However, it would also be useful if a designer could have at his or her disposal, knowledge about an application domain in the form of a repository of application-specific knowledge [Lloyd-Williams 1997].

Ontologies have been proposed as an important way to represent real world knowledge and, at some level, to support interoperability [Swartout 1999; Gualtieri 2005; Knight et al. 2006]. A domain ontology essentially provides knowledge of the relevant terms of a domain. If a knowledge base of such terms were available and could be processed automatically, it could both speed up the design process and make the results more relevant.

Ontologies help to capture the semantics of a domain by acting as a surrogate for the meaning of terms. Semantics, for the purpose of this research, is defined as the meaning, or essential message, of the terms used in a conceptual model, that is, of words and phrases representing entities and relationships. First, domain knowledge, as stored in an ontology, can help when creating a database design because it can suggest what terms (e.g., entities) might appear in an application domain and how they are related to other terms. The relationships in an ontology map to business rules and have implications for semantic integrity constraints. Second, the comparison of constructs in a partial design with those in an ontology could highlight missing constructs.

The objectives of this research are to

—develop a methodology for creating or evaluating entity-relationship models using domain ontologies that capture and represent some of the semantics of an application domain;
—implement the methodology in a prototype;
—empirically assess the effectiveness of the methodology.

The contribution of the research is to provide a way to create database designs that capture and represent the semantics of an application. It also demonstrates how domain ontologies can be used to facilitate database design and provides a prototype for assessment of ontology use. This should minimize the amount of work on the part of the designer and result in the development of more comprehensive and consistent conceptual models.
2. RELATED RESEARCH

2.1 Database Design

Much progress has been made on developing methodologies for database design. Methodologies that support distinct conceptual and logical design have been proposed and research carried out on automating various aspects of these design phases [Denis and Wixom 2000; Alter 1999]. This has resulted in the development of CASE tools with sophisticated diagramming capabilities. These tools have proven useful for consistency checking of the syntactic aspects of a design [Stewart and Arora 2003]. However, they could be even more useful if they had access to knowledge about the semantics of an application domain [Noah and Williams 2002].

2.2 Conceptual Modeling

Conceptual modeling deals with understanding the real world and representing it in such a way that it can be translated into a design that captures the main aspects of an application. Using the entity-relationship model, this means applying the entity, relationship, and attribute construct. A relationship is of the form A verb phrase B where A and B are entities. In linguistics, the semantic counterparts of verb phrases are events. Events are phenomena that occur in time, either statically (one sits in front of a computer) or dynamically (one walks to lunch) [Guarino, 2005]. Bunge [1977] and others consider events that include some notion of time to be only dynamic. Understanding relationships, therefore, includes understanding and classifying events, although not all relationships correspond to events. Prior research on conceptual modeling has distinguished entities from attributes and relationships and focused on defining the nature of relationships [Dey et al. 1999; Wand et al. 1999; Weber 1996; Shanks et al. 2002, Weber 2002a].

2.3 Ontologies

An ontology defines a set of constructs used to represent real-world phenomena [Allen and March 2006]. An ontology should be a way of describing one's world, although there are many different definitions, descriptions, types, and approaches to ontology development [Weber 2002b; Welty and Guarino 2001]. Ontologies generally consist of terms, their definitions, and axioms that relate them [Gruber 1993]. They can also be thought of as the consequence of capturing, representing, and using surrogates for the meanings of terms [Purao and Storey 2005]. This research focuses on domain ontologies that specify conceptualizations specific to a domain [Weber 2002b]. For example, a domain ontology for auctions could consist of terms such as item, bid price, bidder, and seller and the relationships between them.

Several research projects have attempted to use some form of ontologies for conceptual design. Bergholtz and Johannesson [2001] propose an ontology to classify relationships based on data abstractions and speech acts. Storey [2005] presents an ontology for classifying the verb phrases of relationships based on research in linguistics and semantic data models. Purao and Storey
Role of Domain Ontologies in Database Design

[2005] develop a multilayered ontology for classifying relationships by using data abstractions and other data modeling primitives and by separating domain-dependent and domain-independent aspects of the relationship constructs. Storey et al. [1998] classify entities into various categories. All of these are interactive classification schemes that attempt to capture semantics by mapping terms or phrases to predefined categories. Other research on the classification of verb phrases includes Dahchour’s [2001] proposal to understand relationships using meta-classes. Another project is found in Siau [2004]. These research efforts, however, concentrate on the classification schemes and most require user interaction to make the mapping from the term name to the classification category.

Database design is a good problem for the application of ontologies because having a surrogate for the meaning of constructs could help a designer with little knowledge of an application domain. Furthermore, conceptual modeling using the entity-relationship model has only three main constructs to consider. However, most ontologies are not sufficient for capturing the level of semantics needed for conceptual modeling. For example, there is no mechanism for representing semantic integrity constraints that need to be reflected in a design.

2.4 Ontology Libraries

Ontology development is fundamentally a difficult problem. Research on creating ontologies and using ontologies has been motivated by the Semantic Web and knowledge reuse [Ding and Foo 2002; Pinto and Martins 2004; Lammari and Metais 2004]. The Semantic Web is intended to extend the World Wide Web by capturing and representing the semantics of an application domain in domain ontologies [Hendler 2001]. Research on the Semantic Web instigated the need for libraries of ontologies, the most well-known of which is the DAML ontologies that were developed specifically for the Semantic Web (www.daml.org).

Another motivation for the development of ontology libraries is for knowledge reuse, which is important because of the overwhelming amount of information that is constantly being generated. It is intended that such ontology libraries will be publicly available [McGuiness 2002].

The creation of ontologies requires a great deal of effort to identify domain terms and how they are related, to organize the terms into hierarchies, it separate instances (e.g., Charlottetown) from type (city) and so forth. Much ontology development is carried out on a manual basis, although tools and techniques are being developed for automatic or semiautomatic generation of ontologies. Ding and Foo [2002] suggest that small ontologies are useful because they enable semantic agreement among the people using them with little time lag. Although research on semiautomatic or automatic ontology evolution and generation has been carried out [Embley et al. 2005; Al-Muhammed et al. 2005; Storey et al. 2005; Oberle et al. 2004], current efforts are at an early stage of development and still require user intervention. Embley’s [2004] methodology for generating ontologies, for example, considers the information stored in tables on the Web for a specific domain. Ontology integration issues must also be addressed.
with some resemblance to research on heterogeneous databases [Ding and Foo 2002].

3. ONTOLOGY FOR DATABASE DESIGN

This section presents an ontology that incorporates domain knowledge for database design. The method used to develop the ontology is outlined, and the components and representation issues discussed.

The research is intended to use libraries of ontologies as they are developed. As mentioned previously, there has been much call for the development of ontologies for different application domains that can be shared among different applications and interest groups [Embley 2004; Heflin and Hendler 2000; Fensel et al. 2001; Brewster and O’Hara 2004; Noy et al. 2004]. This research on database design is one such application. As research progresses on automated and semiautomated ontology development [Ding and Foo 2002], the availability of useful ontologies will increase. Although the actual development of ontologies is not the focus of this research, the following outlines the procedure followed for the manual, but systematic, development of the ontologies used in this research.

3.1 Components and Development

The main components of an ontology are terms, relationships, and axioms. The relationships that typically appear in an ontology are subclass-of (is_a), instance-of, synonym, and related-to [Gomez-Perez 1999; McGuinness 2002]. This research extends these main concepts to include domain relationships that capture additional constraints needed for database design and which represent, to some extent, semantic integrity constraints. There additional constraints capture various aspects of a business application.

The ontology for the auction ontology was developed using the following steps.

Step 1. Identify category of Web site. For this research, it was the retail category.

Step 2. Specify target domain and initial domain knowledge. The target domain was auction. For the auction domain, the keyword “auction” was used as input to a search on Google.

Step 3. Crawl and scan Web pages. From the Web sites for auctions, the standard format of the auction domain was extracted. This enabled the researchers to identify the main terms of the application domain. It also enabled the researchers to infer which terms were somehow related to each other. Then, the domain relationships were identified.

Step 4. Extract concepts. The terms extracted were in the form of nouns and verbs which translate loosely into entities and relationships.

Step 5. Analyze and cluster extracted features. A content analysis of the extracted concepts from the Web pages identified which terms might belong together based on the frequency of their appearances together. For example, item and bid occur together. From these, inferences were made based on common sense knowledge of the application. For example, an item is a prerequisite for a bid.
The relationships between the terms were generated by examining how close the terms appeared to each other on the Web site and by adding constraints based on knowledge of the application domain. For example, in order for a customer to buy an item, a customer needs to bid on it, to be a seller, one must have an item to sell, and so forth. These translate into constraints, for example, prerequisite constraints.

3.2 Ontology Components and Representation
The auction domain ontology used in this research is a lightweight ontology when compared, for example, to upper level ontologies such as CYC [Lenat et al. 1995], OWL ontologies (http://protege.stanford.edu/plugins/owl/owl-library/index.html), or SUMO [Pease et al. 2002].

In ontology development, concepts or terms are often organized into taxonomies with binary relationships. Most ontologies focus heavily on terms, often organized as a class and subclasses (e.g., the DAML ontologies). To a much lesser extent, instances of terms are included. Representing and reasoning with these basic components is not enough to create a heavyweight ontology capable of supporting complex reasoning [Corcho et al. 2003a]. Therefore, before implementing an ontology, one should decide on the application needs in terms of expressiveness and inference.

This research extends the relationship component of an ontology [Sugumaran and Storey 2002]. For conceptual modeling, the basic relationships (synonym, is a, and related to) employed in an ontology are limiting because of their inability to capture domain specific knowledge (business rules). We model other types of relationships, which we call domain relationships, the purpose of which is to define and enforce business rules. The following four types of domain relationships, a) prerequisite, b) mutually inclusive, c) mutually exclusive, and d) temporal, are defined as follows.

**Prerequisite relationship.** One term/relationship depends upon another. For example, in the online auction, a bid requires an item. However, information about the item can exist without requiring a bid. Consider two terms A and B. If A requires B, then, the prerequisite constraint is: A → B. Term B may occur by itself; however, for term A to occur, term B is required.

**Temporal relationship.** One term or relationship must occur before another. If A must precede B, the temporal constraint is: A ↞ B. For example, to bid on an item, one must register as a bidder.

Temporal relationships can be difficult to establish. Considerable research examines the dimensions of time and how the notion of time can be incorporated into information system [Allen 1983]. There has been some ontology development work done to capture time. For example, the DAML ontology includes the terms time point, time unit, month, day-of-the week, time lapse (between one time and another), hour, year, calendar, start and end time, duration, interval sequence, upper and lower bounds, etc. This ontology, thus, captures basic time measurement and other notions such as duration, overlap, meets, etc. However, it does not provide adequate representation for temporal dependence.
Mutually inclusive relationship. One term may require another for its existence. If A requires B and B also requires A, a mutually inclusive relationship exists between A and B, $A \leftrightarrow B$, signifying that these two terms occur simultaneously. To be a bidder, one must make a bid. Likewise, a bid cannot be made without a bidder.

Mutually exclusive relationship. One term or relationship cannot occur at the same time as another. For example, an individual cannot be the buyer and seller of the same item. This operates on the relationship for availability and the relationship for closed bidding. If A and B are mutually exclusive, then only one of them can occur in a given context. The mutually exclusive relationship is: $A \leftrightarrow B$.

An ontology is conceptually represented as a semantic network where the nodes correspond to the ontology's concepts or terms, and the arcs correspond to various relationships as illustrated in Figure 1. This semantic network is converted into a set of facts, using standard templates. These templates are designed to be application-domain independent so that they can be processed by a number of applications and promote interoperability. The internal representation of the ontology components stored in the domain ontology repository is given in the following.


Fig. 1. Partial auction domain ontology.
Role of Domain Ontologies in Database Design

The term component is expressed as a fact using the following template: (term: id name), where term: is the keyword indicating the type of fact, namely, the fact representing terms. id is an integer that uniquely identifies the term, and name is a string of characters that represents the term itself. For example, the concept of “Bid” in the auction domain is expressed as (term: 1 Bid). The relationship component is represented using the following format:


In this fact, the keyword relationship: indicates that it is a relationship fact. It also contains other keywords, namely, tm1:, tid1:, rtype:, tm2:, tid2:, which help separate the specific values of the ids and the terms within the fact. In addition, rel-id is an integer that uniquely identifies the relationship fact, term1 is a string of characters that corresponds to the first term, id1 is the identifier of the first term, relationship-type represents the type of the relationship, namely, is-a, synonym, related_to, prerequisite, mutually-inclusive, mutually-exclusive, and temporal. The second term in the relationship is represented by term2 with id2 as its identifier. An example of a relationship fact is:

(relationship: 5 tm1: Bid tid1: 1 rtype: prerequisite tm2: Item tid2: 3)

The relationship fact (id 5) indicates that the term “Bid” (id 1) has a prerequisite relationship with the term “Item” (id 3). That is, the Item must exist in order to place a bid on it.

In terms of the entity-relationship model, the ontology currently captures entities and relationships. Part of the difficulty with capturing attributes is that existing ontologies do not distinguish them. Therefore, rules need to be written to do so. Consider, for example, the computer science department ontology from the DAML library. It contains the classes (which roughly map to entities) of student, subject, faculty, workshop paper, journal, student, etc. The ontology is very good at identifying subclasses (e.g., full-time or part-time professors, journal or conference papers). However, it does not contain attributes such as name, address, date-of-birth for professor, or date of submission for research papers).

4. ONTOLOGY-BASED APPROACH FOR DATABASE DESIGN

The two primary ways in which an ontology could be used to improve a design are 1) conceptual model generation and 2) conceptual model validation. The role of an ontology is to provide a comprehensive set of terms, definitions, and relationships for the domain for which a conceptual model is to be created. Therefore, the first task is to generate a design from scratch, using the terms and relationships in the ontology as representative of the domain. The second task is to use the ontology to check for missing terms or inconsistencies in an existing or partial design.
4.1 Conceptual Model Generation from Scratch

The steps involved in creating an entity-relationship (E-R) model from scratch using an ontology are shown in Figure 2. The steps are a) identify initial user terms, b) map the initial terms to domain terms in the ontology and expand the model, c) check for consistency, d) generate the E-R model, and e) map the E-R model to a relational model. These steps are accomplished based on a set of rules which are stored in a knowledge base and summarized in Figure 3.

Step (a) Identification of Initial Terms. The user provides his or her application requirements or a textual description of the problem domain and desired functionalities. Consider the following user requirement.

“I want to design a database to support an auction Web site. It needs to keep track of products, their seller and buyers, and prices at which they are sold.”

This requirements fragment can be parsed using simple natural language processing techniques (parts of speech tagged parsing [Mason and Grove-Stephensen 2002]) and key terms.

For the purposes of this research, simple natural language processing techniques are used. The parts of speech tagger identifies the nouns and noun phrases (potential entities) as well as verbs and verb phrases (potential relationships). Here, the potential entities are Product, Seller, and Buyer. The ontology further suggests transitive inferences. For example, an item is related to a category. Since a seller is related to an item, it is possible that a seller is related to a category, and corresponding relationships are needed.
Rule #1 – Finding related terms

If: Term X is selected and
   Term X has a "related_to" relationship with Term Y and
   Term Y is currently not selected

Then: display a message with this information to the user and
       get user feedback and
       select Term Y if desired

Rule #2 – Rule for checking synonyms

If: Term X specified by user
   There exists a Term Y in the ontology
   Term X is a synonym for Term Y

Then: Term Y is more appropriate
       Display a message with this information to the user
       Get user feedback and
       Replace Term X with Term Y

Rule #3 – Rule for checking subclass

If: Term X specified by user
   There exists a Term Y in the ontology
   Term X has "is_a" relationship with Term Y

Then: Term Y is a subclass of Term Y
       Display a message with this information to the user
       Get user feedback and

Rule #4 – Rule for mutually inclusive relationship

If: Term X is to be selected and
   Term X has a “mutually-inclusive“ relationship with Term Y and
   Term Y is currently not selected

Then: display a message with this information to the user and
       get user feedback and
       select both Term X and Term Y

Rule #5 – Rule for identifying relationships for an entity

If: Entity X is included in the model and
    Entity Y is included in the model and
    There exists a relationship RI between Entity X and Entity Y in the
    repository

Then: RI is a candidate relationship
       display a message with this information to the user

Rule #6 – Rule for prerequisite relationship

If: Term X is to be selected and
    Term X has a “prerequisite“ relationship with Term Y and
    Term Y is currently not selected

Then: Term Y should be automatically selected
       Display a message with this information to the user and
       Get user feedback and
       Select Term X as well as Term Y if desired

Fig. 3. Sample rules (implemented as JESS rules in the system).

A designer might need to refine a fragment of an existing entity-relationship model. An ontology could check that the names of the entities are the best ones for the application domain. The inclusion of synonyms helps to identify the most appropriate label for an entity or relationship, assuming that the terms used in the ontology are the most appropriate. For example, Customer has Account could be refined to Customer owns Account. The ontology can also suggest possible missing entities and relationships. The simplest way to do so is to trace through the possible inferences that can be made from one term to another. For example, the ontology can expand the E-R diagram by identifying Bidder and Seller as subclasses of Customer. Relationships between Bidder and Seller and Product are then added to the E-R diagram. The user can select which terms to include in the model or provide appropriate terms directly.

Step (b) Map to domain terms and expand list of terms. After the initial set of terms is identified, the terms are compared to those in the ontology for that
role of domain ontologies in database design

domain in order to expand the terms. There are two ways this is done. First, the synonym and is-a relationships from the ontology are used to verify the appropriate use of the terms in the design (e.g., the user requirement mentions the term Product.) However, in the auction domain, the term Item is used to refer to things being auctioned. Hence, the appropriate term to use is Item. From the auction domain ontology, Item is a synonym of Product and, hence, the term Item will be used. Similarly, suppose that in the domain ontology there is a relationship Electronic-item is-an item. Then, this might suggest that Electronic-item is needed as an additional entity. Since Item is an entity in the original model, then an inference is made that this is an entity.

The initial set of terms is expanded using the related-to relationship to identify additional terms. For example, if Customer buys Item and Item related-to Bid, then a relationship might be a needed between Customer and Bid (Customer places Bid). Rule #1 (see Figure 3, which lists all the rules) is applied. Several rules are used to check whether the term appears in the ontology. These include 1) whether the specified term is a synonym of a typically used term (Rule #2) and 2) whether the term is a subclass of another term (Rule #3). There are also rules to transitively compute other terms that are related to the original term by following the related-to relationships. At the end of this step, an expanded list of terms is presented to the user for feedback.

Step (c) Consistency checking. After gathering the set of potential terms, additional domain knowledge can be used to check whether the terms and concepts selected are complete and consistent. The four types of domain relationships, prerequisite, temporal, mutually-inclusive, and mutually-exclusive, ensure that all the relevant terms and concepts have been included or excluded. For each term, the domain relationships defined in the ontology are first identified. Then, for each of those relationships, the associated terms are gathered and the corresponding rules enforced to ensure that the terms are consistent. For example, if a particular term has a prerequisite relationship with another term, then the rule would check whether the related term is also included. If not, it will be selected automatically. Similarly, if two terms are mutually inclusive and if only one of them is included in the model, then the consistency checking rule would add the other term to the model and inform the user. In the auction domain, Item and Seller are mutually inclusive. Rule #4 (Figure 3) checks for mutually inclusive relationships.

Step (d) Generate E-R model. Entities are created based on the terms identified in the previous step. Relationships between entities are generated by taking into account the application domain semantics inherent in the ontology. The verb phrases gathered from natural language processing of the problem description are used as the starting point for identifying relationships between entities. Previously generated E-R models and typical data model fragments are stored in the E-R model repository. This repository is used to identify commonly occurring relationships within an application domain and to generate the relationships between entities. For example, a typical entity-relationship model in the auction domain would have entities such as Buyer, Seller, Item, Bid, and Transaction and relationships that correspond to Seller sells Item,
Buyer places Bids, Item receives Bids and possibly others. Binary relationships in a given application domain are stored as facts within the repository using a predefined template. Rules pattern match against these facts to identify potential relationships for the terms/entities identified in the previous steps. Rule #5 (Figure 3) identifies one or more relationships for two given entities that are to be included in the model. For example, if the list of entities under consideration contains Buyer, Item, and Bid, they can be matched against the auction E-R model fragments and appropriate relationships identified. At the end of this step, an overall E-R model is generated and presented to the user for feedback. The user can add or delete any relationship.

Step (e) Map to relational model. Once the E-R model is created, the corresponding relational data model is generated using well-established rules of mapping from the entity-relationship to relational model [Teorey et al. 1986]. The E-R model is stored in the E-R model repository.

4.2 Validation of an Existing E-R Model

An existing E-R model can be validated by checking whether it includes some of the entities that are typically part of similar designs in that application domain. The model can also be checked to assess whether the entities and relationships are internally consistent. The E-R validation is shown in Figure 4 and consists of the steps a) check the appropriateness of terms, b) identify missing terms, and c) generate an augmented E-R model.
Step (a) Appropriateness of terms. The names of the entities are checked for appropriateness using the domain knowledge contained in the ontology. The terms in the ontology are the commonly used terms in a domain. The ontology also contains synonyms. If the model uses a synonym, the original term is suggested because one of the purposes of an ontology is to contain the most commonly used terms in a given domain. For example, the designer might have used *Consumer buys Item* with the ontology identifying synonyms (*Consumer*, *Customer*). Then, the designer would need to decide which label to use. The default is the one in the ontology, for instance, *Customer*. Of course, this could be modified to refine user verification.

Step (b) Identify missing terms. If entities appear in *is a* or *related to* relationships, they are used to gather related terms. The domain relationships also ensure that they are internally consistent. For example, Rule #6 (Figure 3) checks if the prerequisites for a given term are also included in the model. If not, they are added with the user’s approval.

The terms in the ontology may correspond to either an entity or an attribute in an entity-relationship model. Currently, it is not possible to easily distinguish which term should map to an entity and which to an attribute. Since ontologies usually contain classes (represented as nouns), the implementation assumes that a missing term (one in the ontology, but not in the entity-relationship model) is a potential missing entity. For example, the computer science ontology contains the term “Lecturer” which could suggest a missing entity. If there is a relationship in the ontology that does not appear in the E-R model, it could be a missing construct. For example, if “Publication” and “Professor” appear together in a computer science ontology, this suggests a relationship.

Step (c) Generate an augmented E-R Model. For terms identified in the previous step, appropriate relationships are suggested based on the relationships gleaned from the E-R model fragments stored in the E-R model repository. The repository will build up over time. For the entities included in the model, pattern matching is done against the entities in the repository, and candidate relationships are identified and suggested as potential new relationships to be considered. Rule #7 (Figure 3) shows a rule for identifying additional relationships. With designer’s feedback, an augmented E-R model is generated that meets the requirements specified by the designer.

4.3 Consistency Checking

The domain relationships allow one to express the application domain semantics and business rules. For example, in an online auction domain, a *Bid* needs (prerequisite) *Item* and *Bidder*. The temporal constraint for *Bid* shows that an *Account* must be created before a bid can be placed. The mutually exclusive constraint for the term *Seller* indicates that the seller of an item cannot also be its buyer. Thus, while adding a term to or deleting a term from the model, the domain relationships ensure that the selection or deletion action results in a consistent set of terms. A set of heuristics perform the consistency checking, two of which are described in the following sections.
4.3.1 **Heuristic for Term Selection.** For a given term, this heuristic identifies other terms that must be included. For example, if there are prerequisite relationships for a term, those terms must also be included. If the term has a mutually inclusive relationship, the mutually inclusive term must be included. For example, in the auction domain, the mutually inclusive relationship for *Bid* implies that, whenever *Bid* is included in a design, the associated terms *Item*, *Buyer*, and *Seller* should also be included. On the other hand, if the term has a mutually exclusive relationship, those mutually exclusive terms should not be part of the design. Enforcing this relationship involves determining if two or more terms in a model are mutually exclusive. If so, this conflict must be resolved with user feedback.

The term selection heuristic begins by gathering the domain relationships that exist in the ontology for the term selected. For each relationship type, it creates a set to store the associated terms, that is, different sets are created for prerequisite, mutually inclusive, and temporal relationships. For each term added to these sets, their domain relationships and, in turn, corresponding terms are gathered. Thus, for the initial term, associated terms are gathered recursively and included in the model with user feedback. For each term added, its mutually exclusive terms are also gathered and checked to determine whether they are already included. If they are present, the current term cannot be added and user feedback is sought to resolve conflicts. Rule #8 (Figure 3) shows an example of how to check for mutually exclusive relationships.

4.3.2 **Heuristic for Term Deletion.** This heuristic ensures consistency when deleting a term from the solution. When a term is deleted, there cannot be any other terms that depend upon it. Therefore, if the term to be deleted is a prerequisite or a mutually inclusive term for another term that is currently included in the model, that term should not be deleted. On the other hand, when a term is deleted, its prerequisite and mutually inclusive terms can also be deleted, provided no other terms require them.

The term deletion heuristic also starts by gathering the prerequisite, mutually inclusive, and temporal relationships for the term to be deleted. The terms in these relationships are checked against the terms that are currently selected. If at least one term is selected, then the deletion of the initial term would violate the relationship. Hence, the term cannot be deleted and the user is informed of the violation. Rule #9 (Figure 3) checks for dependent terms still selected when deleting a term. If there is another term that depends upon the term to be deleted, then this action is aborted.

5. **OMDDE Architecture and Implementation**

The architecture of Ontology Management and Database Design Environment (OMDDE) is shown in Figure 5. It has an HTML client, a Web server that supports Java technologies, and a relational database. The client is a basic Web browser for browsing or modifying existing ontologies or creating new ones. The server side consists of 1) an ontology management component, 2) a database design component, and 3) repositories.
Role of Domain Ontologies in Database Design

A prototype of OMDDE has been developed using Java technologies, Servlet/JSP (Java Server Pages), and JESS [Friedman-Hill 2006]. The Model-View-Controller separates data access functionality from the presentation and control logic. For example, the ontology creation and management, E-R model generation, and E-R model validation modules are implemented as servlets that use JESS rules and facts stored in the knowledge base. The visualization module is implemented using JSP technology. The E-R model generation and validation activities use java servlets for gathering user input; identifying key terms from the user's description of the requirements; accessing and retrieving domain ontology information; identifying missing elements from the model; and passing the results to the JSP pages. The consistency checking and E-R model generation tasks are accomplished through rules written in JESS that use domain specific knowledge in the ontology and the E-R model repositories. The components of OMDDE are briefly described in the following.

5.1 Ontology Management Component

The ontology management component manages the domain ontologies. It consists of a) a browse ontology module and b) an ontology creation and management module. This component facilitates the creation, storage, and use of ontologies. The ontologies are stored in a simple format, expressed as unordered facts in JESS as shown in Figure 6. It is very difficult to automatically, or semiautomatically, use someone else's ontology. However, our prior research attempted to assess the quality of ontologies which will also impact how effectively they can be used [Burton-Jones et al. 2005].

**Browse Module.** The browse module provides mechanisms for navigating the ontologies stored in the repository. The user can search the repository through key words or select specific ontologies to view by name or id.

V. Sugumaran and V. C. Storey

Fig. 6. Example ontology facts.

```
(defun find-related-terms ()
  (cond
    (selected ?tid1 ?X)
    (related-to ?tm2 ?Y ?tid2)
    (not-selected ?tid2 ?Y)
    (=
      (bind ?str1 (str-cat "Since the term " (implode $Y) " is related to " "" (implode $X) "", " "")
           (bind ?str2 (str-cat "it may be of interest to the user.")
                    (bind ?str3 "Click on 'OK' to include it to the list, or 'Cancel' to abort.")
                    (bind ?str (str-cat ?str1 ?str2 ?str3))
                    (bind *d* (new YNDialog ?*current-pnl* "Related Term" ?str "OK" Cancel ")))))
```

Fig. 7. JESS rule for finding related terms.

**Ontology Creation and Management Module.** This module assists the user in creating or modifying an ontology. It facilitates the basic steps for creating ontologies and managing and evolving existing ontologies.

5.2 Database Design Component

**E-R Model Generation Module.** This module implements the steps described in Section 4 for creating an entity-relationship model from the users' requirements. The module interfaces with a natural language parser to obtain the problem description tagged with parts of speech. The nouns and noun phrases are the starting point for identifying terms and checking them against the ontology. This module implements various rules for identifying entities and relationships. An example rule in JESS is shown in Figure 7. This rule is used to find related terms. For example, if a particular term ($?X$) is initially specified by the user and if it is related to another term ($?Y$), then it may be of interest and should be included in the list of potential terms for user feedback.

After the initial terms are identified, they are refined using the relationships embedded within the ontology and the entity-relationship models from the repository. The system suggests relationships for the E-R model, generates the overall entity-relationship model, and passes it to the visualization module for display. These rules rely on both the ontologies and the E-R models stored in the repositories and include capabilities for incrementally creating the E-R model fragments, obtaining user feedback, and building the overall model.

**E-R Model Validation Module.** This module checks the consistency of an E-R model, or portion of one, created by the user. It identifies terms that are in the

---

^1This corresponds to a multifield variable in JESS.

user's model but not in the repositories so they can be considered for updating the ontology.

Browse/Visualization Module. The user can view the E-R diagram that has been created up to a given point.

5.3 Repositories
There are two repositories: 1) the domain ontology repository and 2) the E-R model repository.

Domain Ontology Repository. The domain ontology repository stores the ontologies by application domains. The user can browse the ontologies to learn more about a domain. The ontologies can be created by different domain analysts and evolve over time. The contents of the ontology are represented as facts and standard templates with the format for some of the facts shown in Figure 6.

E-R Model Repository. This repository contains E-R models that have been created over time. By examining the E-R models in a domain and their corresponding requirements, generic relationships that exist between entities can be identified. The E-R models are stored as fragments within the repository. Each E-R model is broken down into fragments with each fragment corresponding to a relationship that exists within the model. For each fragment, its identifier, the E-R model of which it is a part, the entities involved, the name of the relationship, and its purpose, are stored. This information is stored as JESS facts the format of which is shown in Figure 8. Rules are followed that search for potential relationships for entities based on user requirements. Relevant fragments from the E-R model are identified by pattern matching on entity names through the left-hand side conditions and corresponding relationship names are suggested to the user by the right-hand side of the rule.

5.4 Sample Session
This section illustrates the creation of an ontology and its use for generating and validating an E-R model. The user (database designer) selects either ontology management or conceptual modeling functions as shown in Figure 9. The system supports the ontology management activities a) creation of a new ontology, b) modification of an existing ontology, and c) browsing ontologies of a particular domain. The conceptual modeling functions supported by OMDDE enable the user to 1) create an initial version of the model using the knowledge contained in the ontology as a starting point or 2) validate a conceptual model the user has created against the ontology to check for the appropriate use of terms and for missing constructs.
5.4.1 Ontology Creation and Management Activities. The “Ontology Management Assistant” and “Database Design Assistant” subsystems implement the ontology management functions and the database design functions, respectively. When the user wants to create a new ontology, the system prompts the user for the domain and ontology names. The system allows the user to define terms, basic relationships, and domain relationships. The system presents the terms that have been added to the ontology so the user can select two terms in order to establish a relationship between them. The basic relationships currently supported are is-a, synonym, and related-to; the domain relationships supported are prerequisite, mutually inclusive, mutually exclusive, and temporal. When a relationship is created, consistency-checking rules are enforced to ensure that relationships and constraints are consistent.

In “Edit Ontology” (Figure 9), the system presents the names of ontologies that exist within each application domain and prompts the user to select one to be modified. Any aspect of the ontology can be updated, and the system ensures that the changes are consistent. For example, the user may update an ontology by adding or modifying terms and relationships. The system ensures that the changes are consistent by invoking a set of rules that check whether a term or relationship is already in the ontology (if it is identified as a new term or relationship), the basic and domain relationships defined for terms are consistent, and the descriptions of terms and relationships are complete. Similarly, when a term is being removed from the ontology, it checks which relationships the term appears in and deletes those relationships as well.
5.4.2 Conceptual Modeling Activities. The Database Design Assistant subsystem controls the user interactions during conceptual modeling activities. The user can create an entity-relationship model or augment and validate an existing model through the interface shown in Figure 9. The database designer can explicitly check whether the model uses appropriate terms and relationships or is missing any major constructs. This is accomplished by 1) mapping the entities and relationships of the current E-R model to the domain ontology and 2) enforcing the constraints and relationships through consistency-checking rules. The system checks for missing and superfluous entities and relationships and presents a report to the user.

The designer can start the E-R model creation process through the “Create Conceptual Model” link shown in Figure 9. The system then provides another screen where the user can input natural language text describing the problem. Through natural language processing and consulting the ontology, the initial entities and relationships are identified. The user may also choose to enter initial entities and relationships based on his or her requirements. These are augmented by applying the heuristics and rules described in the previous sections so a richer model can be developed.

The designer can validate a partially completed E-R model through “Validate Model” in Figure 9. The designer can input information about the entities and relationships using the interface shown in Figure 10. At the end of the process, the system generates a report that suggests new entities, new relationships, name changes for entities and relationships, and so forth. In Figure 10, the validation report suggests adding “Bidder” and “Seller” entities and three other relationships. The designer can request that the updated E-R model be
presented. If the system identifies terms that are not part of the ontology, this information is also reported so the ontology can be updated.

6. ASSESSMENT

Our goal is to provide a designer with an ontology that contains knowledge about an application domain that he or she is interested in. The resulting conceptual model, when supported with information from a domain ontology, would be of higher quality than when such domain knowledge is not available. To assess the usefulness of the system, three controlled experiments were conducted. The first experiment aimed at assessing the value of the system when used by novice designers. The second experiment focused on determining whether the system would also help experienced designers by comparing the performance of OMDDE versus a CASE tool. In order to determine whether an ontology would be more useful compared to another source of knowledge for E-R modeling, a third experiment was conducted in which one group used OMDDE and the other group used Wikipedia to obtain relevant domain knowledge. Each experiment and the results are described in the following sections.

6.1 Validation with Novice Designers

The objective of this experiment is to test whether designers who have access to the ontologies in the Ontology Management and Database Design System can generate better quality entity-relationship models than those who do not have access to it. The repository of E-R models is intended to build up over time. Thus, the testing was carried out without an empty E-R model repository. The quality of the model is not a single measure but a composite measure of different facets [Batra et al. 1990; Lindland et al. 1994; Moody and Shanks 1998; Bock and Yager 2002; Turk et al. 2003]. Bock and Yager [2002], for example, identify the following facets: entity, identifier attribute, various types of relationships (unary 1:1, binary 1:M, binary M:N, ternary 1:M, and ternary M:N), and generalization. We focus on the entity and the relationship facet with the quality of the E-R model generated by the subjects measured in terms of the scores for these two facets. We contend that, using the Ontology Management and Database Design System, the designer can gain an understanding of the domain that will help in selecting appropriate entities and relationships. Using the system during E-R modeling should help the user to identify and incorporate relevant (or correct) entity-relationship constructs.

Although the quality of an E-R model depends on various facets of the model, researchers have argued that it is incorrect to compute an overall quality score by adding the individual facet scores [Batra et al. 1990; Bock and Yager 2002]. This is because such a total score would lack construct validity. Hence, the analysis of model quality is conducted at the individual facet level. The following hypotheses were tested:

H1: Designers using OMDDE will produce E-R models with higher scores compared to the models generated by designers without the use of the system;
Table I. Scoring Scheme

<table>
<thead>
<tr>
<th>Facet</th>
<th>Error Classification and Points</th>
<th>Maximum Possible Points for both applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major Error (−1.0 Points)</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Missing</td>
<td>10</td>
</tr>
<tr>
<td>Relationship</td>
<td>Missing</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Medium Error (−0.5 Points)</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Superfluous Entity</td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>Superfluous Relationship</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minor Error (−0.25 Points)</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Incorrect Name</td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>Incorrect Name</td>
<td></td>
</tr>
</tbody>
</table>

H1a: Designers using OMDDE will perform better in identifying appropriate “Entities” for the E-R model than the designers without access to OMDDE;

H1b: Designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers without access to OMDDE.

6.1.1 Output Measurement. The correctness of the E-R model is evaluated by a scoring scheme which specifies how to score a solution on each facet (entities and relationships). A correct solution was developed for each case by the researchers to provide a benchmark for evaluating the subjects’ solutions. Each solution was evaluated by two independent evaluators. Our scoring scheme is a simplified, commonly-used scoring mechanism for judging the quality of E-R models [Batra et al. 1990; Bock and Yager 2002; Turk et al. 2003]. It focuses on the correct identification of appropriate entities and relationships based on a given problem description. For every correct entity or relationship included in the model, one point is awarded. If the subjects fail to identify an entity or a relationship that should be part of the model, one point is subtracted from the score. On the other hand, if the E-R model contains superfluous entities and relationships, a penalty of −0.5 points is assessed. If an incorrect name is used for an entity or a relationship, −0.25 point is assessed. Table I summarizes the scoring scheme.

6.1.2 The Experiment. Students in an introduction to database management system class were solicited to participate in the study. The subjects had no prior knowledge of the study and its objectives. Two E-R modeling cases of comparable complexity were developed by the researchers for the travel and online auction domains. The researchers had developed ontologies for these domains for a prior research project. The travel domain ontology was developed by integrating several existing ontologies a) DAML (http://www.daml.org/), b) Workshop on Evaluation of Ontology-Based Tools [Corcho et al. 2003b], c) SchemaWeb (http://taga.umbc.edu/ontologies/travel.owl), and d) Protégé Web site (http://protege.stanford.edu/plugins/owl/owl-library/travel.owl). Similarly, the auction ontology was developed from: the auction ontology available at the SchemaWeb Web site (http://taga.umbc.edu/ontologies/auction.owl) and by visiting various online auction Web sites. The information in the ontologies is used as a basis for improving E-R models. Neither of the evaluators of the subjects’ responses was familiar with the content of the ontologies.

The subjects were given a two-hour training session on the system. The first half of the training focused on the concept of an ontology, the major components
of an ontology and how ontologies could be used in conceptual modeling. The subjects also used the system on a test case. This allowed the subjects to become familiar with the system. It was also intended to reinforce the concepts covered in the first half of the training.

The subjects were randomly assigned to two groups, group A and group B. They filled out a questionnaire which asked for their experience in Management Information Systems (MIS)-related courses and/or experience. Each group analyzed the travel and the auction cases and developed the corresponding E-R models. One case was analyzed using OMDDE, and the other case was analyzed without the benefit of the system. Participants were trained on the format of the solution to be generated. The E-R model was represented by listing the names of the entities and the binary relationships that exist using the triple ‘<Entity1, Relationship Name, Entity2>’. The specific assignment of the groups and the cases is shown in Table II.

The solutions generated by the participants were transcribed into a common format so that the evaluators could not determine whether the solution was generated with the benefit of the system. The solutions were numbered for reference, randomized and sent to the two evaluators who were not involved in the experiment. They were provided the answer key for each case and the scoring scheme.

6.1.3 Data Analysis and Results from the Experiment. The subject pool consisted of 60 students (MIS major) enrolled in a database course. They had been taught the fundamentals of data modeling but were not considered experts in E-R modeling. Two separate evaluators graded the data models based on the answer key provided to them. The inter-rater reliability was high for both entity scores (0.95) and relationship scores (0.90), well above the recommended 85 percent [Kassarjian 1977].

Multivariate Analysis of Variance (MANOVA) was conducted to test the difference in performance between the various groups with the results presented in Table III. For the multivariate analysis, the entity and relationship scores were specified as the dependent variables, the fixed factor was the OMDDE system (with or without) and Case (Travel and Online Auction) was specified as the covariate. Order effect was not significant. The role of the Case used by the subjects was tested to see if it had any significant influence on the overall model quality. As seen in Table IIIa, only the presence of the system was shown to have a significant impact on the E-R model scores; the Case did not have any impact. Also, Table IIIb shows that the mean scores for entities and relationships were both higher when the students used the system. As the overall MANOVA is significant at 0.000 (Table IIIa), this supports H1 that designers using the system would produce E-R models with higher scores compared to the models generated without the use of the system.
Table III. MANOVA Results

Table IIIa. Overall MANOVA of E-R Model Scores

<table>
<thead>
<tr>
<th>Source</th>
<th>F (Pillai’s Trace)</th>
<th>Significance</th>
<th>Partial Eta Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE</td>
<td>0.877</td>
<td>0.421</td>
<td>0.030</td>
</tr>
<tr>
<td>OMDDE (System)</td>
<td>129.354</td>
<td>0.000</td>
<td>0.822</td>
</tr>
</tbody>
</table>

Table IIIb. MANOVA of Scores for Individual Facets of the E-R Model

<table>
<thead>
<tr>
<th>Source (System)</th>
<th>Dependent Variable</th>
<th>With OMDDE</th>
<th>Without OMDDE</th>
<th>df</th>
<th>F</th>
<th>Significance</th>
<th>Partial Eta Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMDDE</td>
<td>Entity Score</td>
<td>8.900</td>
<td>6.508</td>
<td>1</td>
<td>240.907</td>
<td>0.000</td>
<td>0.809</td>
</tr>
<tr>
<td></td>
<td>Relationship Score</td>
<td>6.225</td>
<td>5.100</td>
<td>1</td>
<td>81.811</td>
<td>0.000</td>
<td>0.589</td>
</tr>
</tbody>
</table>

The results shown in Table IIIb also indicate that there are significant differences by treatment on the measures of the individual facet scores. The group that used the OMDDE system outperformed the group that did not use the system in terms of both the entity score and the relationship score. The mean entity score for the group that had access to the system is 8.900, whereas the score for the group without access to the system is 6.508. As seen in Table IIIb, this difference is significant (at 0.000), thus substantiating the hypothesis (H1a) that the designers using OMDDE will perform better on identifying appropriate “Entities” for the E-R model than the designers without access to OMDDE. Similarly, the mean relationship score for the group using the system is 6.225, and 5.100 for the group without the system. This difference is also significant at 0.000 (Table IIIb) and supports the hypothesis (H1b) that the designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers without access to OMDDE.

6.2 Validation with Experienced Designers

The second experiment was conducted to investigate whether designers using the OMDDE system would create better quality E-R models than the designers using a CASE tool. Typically, CASE tools provide a graphical environment and support a number of modeling methodologies. Since UML is widely used for modeling, we selected the Poseidon for UML CASE tool (http://gentleware.com) because of its intuitive interface and ease of use. It facilitates interoperability and its HTML documentation generator allows exporting the models to an HTML format for sharing. It also supports code generation in Java, C++, XML, HTML, C#, VB.Net, and PHP. The following hypotheses were tested.

H2: Designers using OMDDE will produce E-R models with higher scores compared to the models generated by designers using a CASE tool.

H2a: Designers using OMDDE will perform better in identifying appropriate “Entities” for the E-R model than the designers using a CASE tool.

H2b: Designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers using a CASE tool.

To test these hypotheses, experienced IT professionals were used as subjects. Six systems analysts from industry with over ten years of experience in data
modeling and application development, as well as four MIS faculty, who have taught introduction to database management, participated in the experiment. The same experimental procedure was followed as the initial validation. The objective of this experiment was also to test the hypotheses listed. The subjects had considerable background in conceptual modeling using UML and database design, but were not familiar with ontologies or the role ontologies can play in conceptual modeling. Hence, the subjects were provided with some training on ontologies and how they are used in database design. They were familiarized with the OMDDE system by practicing with an example. The subjects also used the Poseidon CASE tool to generate E-R models when not using OMDDE. The subjects were trained on Poseidon for creating conceptual models using class diagrams.

The subjects were again randomly assigned to two groups. Each group analyzed the same travel and auction cases and developed E-R models. The first group started with the OMMDE system for the travel case and then used the CASE tool for the auction case. The second group started with the CASE tool for the travel case and then OMMDE for the auction case. The subjects provided written comments about the ease of use and usefulness of the OMDDE system. The solutions generated by the subjects were transcribed into a common format and randomized before being evaluated. The answer key and the scoring scheme from the previous experiment were used for evaluating the models. Two independent evaluators graded the data models. The inter-rater reliability was high for both the entity and relationships scores (.94 and .90, respectively). The results from the data analysis are given in Table IV.

The cases used and the order in which the subjects analyzed the cases did not have any effect on the scores. As shown in Table IVa, only the OMDDE system had a significant impact on the E-R model scores. As the overall MANOVA is significant at 0.000 (Table IVa), this supports the hypothesis (H2) that designers using the OMDDE system would produce E-R models with higher scores compared to the models generated using a CASE tool.

Table IVb shows that the group that used the OMDDE system outperformed the group that used the CASE tool in terms of both the entity score and the relationship score. The mean entity score for the group that used OMDDE is 9.050, whereas the score for the group that used the CASE tool is 7.100. This
“Ontologies definitely help with appropriate use of domain terminologies. The embedded knowledge in the ontology can help identify related entities as well as relationships. The tool is helpful for users not familiar with the domain. The system could be improved by making it a little bit more flexible.”

“The tool is intuitive and easy to use. It will be useful for creating E-R models quickly and also checking for consistency and completeness. The domain information can be used by a novice user in creating a new model or adding elements to an already existing model. The system suggests things that one may not think of normally.”

“OMDDE is helpful in creating E-R models and validating them. It enables the user to look at different aspects of an application domain. It would be a useful modeling tool. A graphical interface for inputting the E-R diagram or importing the information from a file for validation would be quite attractive.”

Fig. 11. Sample comments from the experiment.

The difference is significant (at 0.000), thus substantiating the hypothesis (H2a) that designers using OMDDE will perform better on identifying appropriate “Entities” for the E-R model than the designers using a CASE tool. Similarly, the mean relationship score for the group using the system is 6.625, and 5.275 for the group using the CASE tool. This difference is also significant at 0.000 (Table IVb) and supports the hypothesis (H2b) that the designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers using a CASE tool.

The subjects provided informal feedback on the system. Selected comments are shown in Figure 11. From this, it is evident that the system performed reasonably well in suggesting entities and relationships and validating the E-R model. The subjects remarked that the system is intuitive and easy to use and that ontologies could play an important role in conceptual modeling and quickly generate a consistent and complete E-R model. The comments were positive, but the subjects also suggested improvements, such as adding additional navigational and explanation facilities.

6.3 Comparing Knowledge Sources

The third experiment was conducted to test whether designers who have access to ontologies can generate better quality entity-relationship models than those who consulted a different source of knowledge. For this experiment, the Wikipedia Web site (www.wikipedia.org) which is a free encyclopedia built collaboratively using Wiki software, was used as the alternate knowledge source. Similar to the previous two experiments the following hypotheses were tested:

H3: Designers using OMDDE will produce E-R models with higher scores compared to the models generated by designers using Wikipedia.

H3a: Designers using OMDDE will perform better in identifying appropriate “Entities” for the E-R model than the designers using Wikipedia.

H3b: Designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers using Wikipedia.

Forty undergraduate students from a database management systems course participated in the experiment voluntarily. They did not have any prior
knowledge of the study and were given a two-hour training session. Subjects were introduced to the concept of ontology and its use in E-R modeling and interacted with the OMDDE system on a test case. They were also instructed how to navigate the Wikipedia Web site to search for information on various topics. The subjects filled out a questionnaire regarding their MIS coursework and experience.

The same experimental procedure was followed as the prior two experiments. The subjects were randomly assigned to two groups and the same travel and auction cases used to develop the E-R models. The first group used the OMDDE system for the travel case and Wikipedia for the auction case. The second group started with the Wikipedia for the travel case and then moved on to the auction case using the OMDDE system. At the end of the experiment, the subjects also provided feedback on their experience with the OMDDE system.

The solutions generated by the subjects were transcribed into a common format and randomized before being evaluated by two independent evaluators. The answer key and the scoring scheme from the previous experiments were used to evaluate the models. The inter-rater reliability was high for both the entity and relationship scores (0.97 and 0.93). The results from the data analysis are given in Table V.

As indicated in Table Va, only the OMDDE system had a significant impact on the E-R model scores. As the overall MANOVA is significant at 0.000 (Table Va), this supports the hypothesis (H3) that designers using the OMDDE system would produce E-R models with higher scores compared to the designers using the Wikipedia. The cases used and the order in which the subjects analyzed the cases did not have any effect on the scores.

Similar to the previous experiments, the group that used the OMDDE system outperformed the group that used the Wikipedia in both the entity and the relationship scores. As seen in Table Vb, the mean entity score for the group that used OMDDE is 9.313, whereas the score for the group that used the Wikipedia is 7.475. This difference is significant (at 0.000), thus substantiating the hypothesis (H3a) that designers using OMDDE will perform better on identifying appropriate “Entities” for the E-R model than the designers using the Wikipedia. Similarly, the mean relationship score for the group using the
OMDDE system is 6.788, and 5.687 for the group using Wikipedia. This difference is also significant at 0.000 (Table Vb) and supports the hypothesis (H3b) that the designers using OMDDE will perform better in identifying appropriate “Relationships” for the E-R model than the designers using Wikipedia.

The subjects provided positive feedback on the ease of use and usefulness of the OMDDE system. They indicated that it was useful in identifying entities and relationships but also suggested some modifications. With respect to Wikipedia, the subjects commented that it was time consuming to go through the information contained in the wikis and there was no support for identifying entities and relationships. Also, the quality of information contained in the wikis may not be consistent.

7. CONCLUSION

This research has demonstrated how domain knowledge stored in an ontology can be used to assist in the generation of complete and consistent database designs, both when the designs are generated from scratch and when a partial design exists. A framework for these two tasks has been presented which uses ontology primitives, such as relationships and constraints between terms, to generate a consistent design. A prototype has been implemented called the Ontology Management and Database Design Environment which supports entity-relationship modeling activities.

This research is an initial step to illustrate how domain ontologies, which capture knowledge about specific application domains, can be used for the creation and validation of entity-relationship models for conceptual modeling. The initial results demonstrate the feasibility and usefulness of the research. The prototype clearly helped both the naive and experienced modelers in creating better quality E-R models compared to either no tool or to a CASE tool or to Wikipedia. The results suggest that domain ontologies can indeed be an asset in supporting conceptual modeling activities. Our future work will refine the prototype, assess its effectiveness for a variety of design tasks, and incorporate organization-specific knowledge for data modeling into the methodology and prototype. It is believed that ontologies, eventually, will help to solve the “grand problem” of incorporating semantics into information processing [Embley 2004]. This research has shown one potential application of useful ontologies.

ACKNOWLEDGMENTS

The authors wish to thank the associate editor, Mary Fernandez, and the anonymous reviewers for their detailed and helpful comments on an earlier version of this article.

REFERENCES


Role of Domain Ontologies in Database Design


• V. Sugumaran and V. C. Storey


Received December 2004; revised September 2005, May 2006; accepted June 2006