December 2003

Supporting Database Designers in Entity-Relationship Modeling: An Ontology- Based Approach

Vijayan Sugumaran
Oakland University

Veda Storey
Georgia State University

SUPPORTING DATABASE DESIGNERS IN ENTITY-RELATIONSHIP MODELING: AN ONTOLOGY-BASED APPROACH

Vijayan Sugumaran
School of Business Administration
Oakland University
Rochester, MI USA
sugumara@oakland.edu

Veda C. Storey
J. Mack Robinson College of Business
Georgia State University
Atlanta, GA USA
vstorey@gsu.edu

Abstract

Database design has long been recognized as a difficult problem, requiring a great deal of skill on the part of the designer. Research has been carried out that provides methodologies and rules for creating good designs. There have even been attempts to automate the design process. However, before these can be truly successful, methodologies and tools are needed that can incorporate and use domain knowledge. In this research, a methodology for supporting database design is proposed that makes use of domain-specific knowledge about an application, which is stored in the form of ontologies. The ontologies provide information that is useful in both the creation of new designs and the verification of existing ones. They also capture the constraints of an application domain. A methodology for assisting database design that takes advantage of the ontologies has been implemented in a prototype system. Initial testing of the prototype illustrates that the incorporation and use of ontologies are effective in creating database design.

Keywords: Ontology, entity relationship modeling, database design, domain knowledge, database design assistant

Introduction

Database design is a difficult process because it relies heavily on a database designer understanding the users' database requirements and representing them so that they accurately capture the real-world application being modeled. A database designer, however, may not have knowledge of the domain being modeled and, thus, relies upon the user to articulate the design requirements. Much progress has been made in creating methodologies and guidelines for assisting the designer in conceptual database design, especially in understanding how to apply design constructs (e.g., Bodart et al. 2002; Parsons and Wand 2000; Shanks et al. 2002; Siau et al. 1997; Weber 1996). It would also be useful, however, if the designer would have at his or her disposal knowledge about the application domain in the form of a repository of application-specific knowledge (Lloyd-Williams 1997).

Ontologies have been proposed as an important means of representing real-world knowledge (Swartout and Tate 1999). They are believed to provide a natural means for representing information for database management systems because databases have inconsistencies in naming conventions, synonyms, and so forth. This knowledge could be particularly useful in two respects. First, domain knowledge, as stored in an ontology, can help in 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, to suggest further entities and their relationships. The constraints that are found in an ontology map to business rules in an application domain which has implications for semantic integrity constraints. Second, given a (partial) design, comparing the constructs in a design with those in an ontology can highlight missing constructs.

The objectives of this research, therefore, are to

- develop an approach for creating conceptual (E-R) models using ontologies for database design
- implement the approach in a prototype
- demonstrate the viability of the approach by generating conceptual (E-R) models from ontologies using the prototype

The contribution of the research is to provide a means for creating better database designs that reflect the semantics of the application while minimizing the amount of work on the part of the designer. This should result in a more comprehensive and consistent conceptual model.

Related Research

Progress has been made in developing methodologies for conceptual modeling. 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 (Alter 1999; Dennis and Wixom 2000). This has resulted in the development of case tools that have sophisticated diagramming capabilities. These tools have proven very useful for consistency checking of the syntactic aspects of the design. They could be even more useful if they had access to knowledge about the application domain. For example, they could use domain specific knowledge to assess whether an entity-relationship model is a complete and consistent representation of a user's application. A tool that incorporates domain knowledge for conceptual modeling would, thus, be useful for database design (Noah and Williams 2002).

Ontologies

Considerable research focuses on creating ontologies and using them to represent knowledge of an application domain. An ontology should be a way of describing one's world, although there are many different definitions, descriptions, types, and approaches to ontology development (Welty and Guarino 2001). For this research, the most relevant are *domain ontologies* that specify conceptualizations specific to a domain (Weber 2002).

An ontology generally consists of terms, their definitions, and axioms relating them (Gruber 1993). The use of ontologies is found in many areas. The semantic web relies heavily on ontologies to support machine understanding (Berners-Lee et al. 2001). Dahlgren (1995) proposes a linguistic-based ontology for natural language processing and uses a common-sense ontology for interpretations of text. In the database area, ontologies have been created for understanding Web-based obituaries (Embley et al. 1999), for managing semantic heterogeneity by use of a linguistic dictionary (Kedad and Metais 1999), and for classifying relationships in conceptual database design (Bergholtz and Johannesson 2001).

Most ontology creation is carried out on a manual basis. To be truly useful, a repository of domain ontologies is needed along with methodologies for creating them. Several frameworks for creating ontologies have been proposed, with the most well-known being SHOE (Heflin and Hendler 2000) and Ontolingua (Farquhar et al. 2002). These provide formalisms for representing basic information pertaining to a domain. For example, SHOE offers constructs such as categories, relationships, inferences, renames, and descriptions for creating an ontology (Heflin and Hendler 2000). The SHOE ontology is used to annotate Web pages to improve searching on the Web. It is possible to describe and categorize basic concepts and relationships that may exist within a domain using SHOE constructs for retrieval purposes. However, it is not sufficient for capturing the level of domain semantics needed for conceptual modeling. For example, there is no mechanism for representing semantic integrity constraints that need to be reflected in a database design.

Ontolingua uses a Lisp-like notation to express ontologies (Farquhar et al. 2002). It is claimed to be language neutral and can be translated into other ontology schemes. However, it does not support an inference engine, so it simply stores all the artifacts that describe a domain without inferring anything about that domain. Ontolingua ontologies are not suitable for database design because they do not lend themselves to inferencing.

Proposed Ontology Representation and Use

Prior research on ontologies has focused on methodologies for developing ontologies as well as the structural aspects of ontologies (Fensel et al. 2001; McGuiness et al. 2000). Much of the development of ontologies, however, has focused on identifying terms and specifying how they are related to other terms. Recent research has focused on the use of ontologies in information systems design (Sugumaran and Storey 2002).

This research proposes an extension to typical ontology structures used in various application domains. In addition to the traditional structural primitives such as the common terms and various kinds of relationships (*synonym*, *is_a*, and *related_to*), we model the notion of *constraints* between terms. Constraints provide a mechanism for incorporating additional domain knowledge into the ontology. This is very helpful in database design because constraints can help in defining and enforcing business rules in conceptual modeling and database design. The proposed ontology structure includes the following four types of constraint primitives: (1) prerequisite, (2)mutually-inclusive, (3) mutually-exclusive, and (4) temporal. These ontology primitives are schematically shown in Figure 1.

Prerequisite constraint: One term orrelationship depends upon another. For example, in the online auction domain, a bid requires an item. However, information about the item can exist on its own without requiring a bid. Consider two terms A and B. If A requires B, then, we define the prerequiste constraint $A \rightarrow B$. Term B may occur by itself; however, for term A to occur, term B is required.

Temporal constraint: One term or relationship must occur before another. For terms A and B, if A must precede B, we define the temporal constraint as $A \mapsto B$. For example, to bid on an item, one must register as a bidder.

Mutually inclusive constraint: One term may require another for its existence. For terms A and B, if A requires B and B also requires A, then we have a mutually inclusive relationship between A and B, i.e., these two terms have to occur simultaneously. Then, we define the mutually inclusive constraint as $A \leftrightarrow B$. For example, bid and bidder require each other. To be a bidder, one must have made a bid. Likewise, a bid cannot be made without a bidder.

Mutual exclusive constraint: 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 constraint for availability and the constraint for closed bidding. If A and B are mutually exclusive, then only one of them can occur in a given context. We define the mutually exclusive constraint as $A \not\leftrightarrow B$.

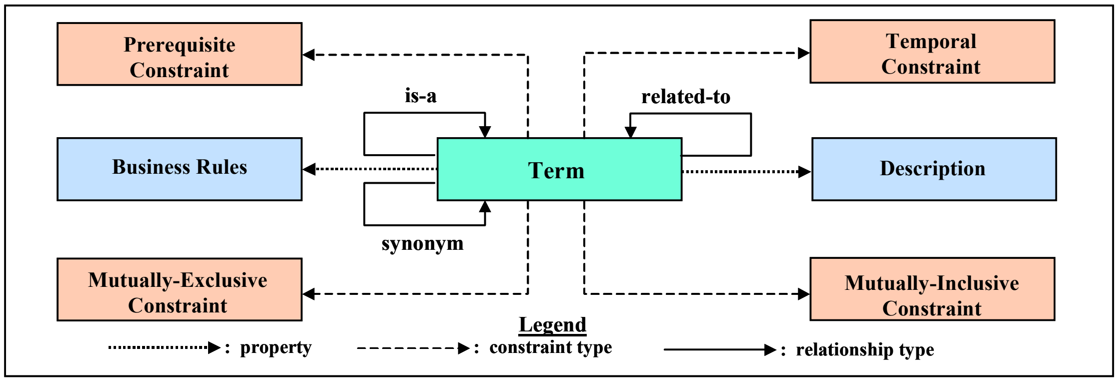


Figure 1. Meta Schema of Domain Ontology for Database Design

The ontology is represented as a semantic network, where the nodes correspond to the terms and the arcs correspond to relationships and constraints, as illustrated in Figure 2. The semantic network representation of an ontology, which is primarily a graph, can be easily traversed and used effectively to improve database design. The database design process usually begins with specifying user requirements. From the requirements, key terms are identified. The role of an ontology is to provide a comprehensive set of terms, definitions, relationships, and constraints for that domain. From these, the designer can select appropriate terms to incorporate in the design. Consider the following requirement fragment: "I want to design a database to support an auction Website. It needs to keep track of the items that are seller and buyers and the prices sold at." This fragment can be parsed using simple natural language processing techniques (parts of speech tagged parsing [Mason and Grove-Stephensen 2002]) and key terms such as item, seller, buyer, and price are identified. Examination of the ontology reveals that the term item suggests the need for *category*, *customer*, and *shipper*. The ontology also suggests that relationships exist between (1) *item* and *category*, *customer*, and *shipper*; (2) *item* and *seller*; and (3) *item* and *buyer*. The ontology can also make further suggestions based on 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.

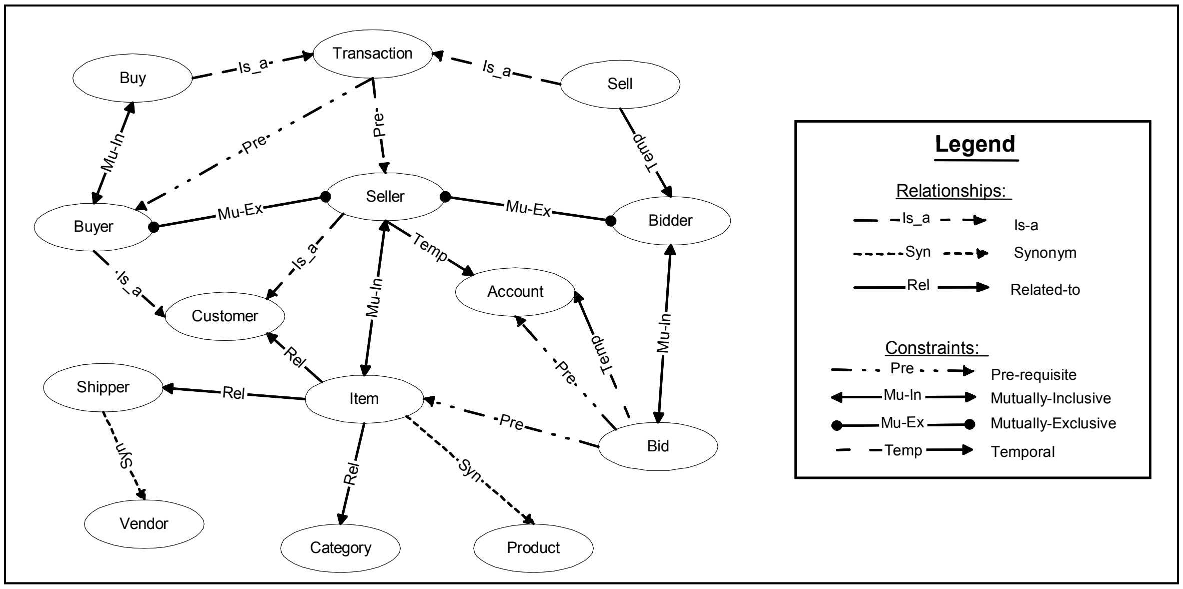


Figure 2. Partial Auction Domain Ontology

A designer may also 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 in the ontology helps to identify the most appropriate label for an entity or relationship. This assumes that the term used in the ontology is the best one for the application domain. For example, the relationship "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.

Ontology-Based Approach for Database Design

There are two main ways in which ontologies can serve as a knowledge base of domain semantics that can be used to improve a database design. The two primary functions for which an ontology could be used are design generation and design verification. The first task is to generate a design from scratch, using the terms and relationships in the ontology as a representative model of the domain. The second involves using the ontology to check for missing terms or inconsistencies in an existing, or partial, design.

Applying ontologies in this way can be useful, first, because it will reduce the amount of work for any designer, in particular, a novice designer. When a design is incomplete, terms from an ontology can be shown to the user to check for missing terms and identify how they are related to other terms. Ontologies can also be used to analyze a design for missing constructs. They can assist in identifying appropriate terms for an application and comparing heterogeneous databases to create a synthesized database or map to a higher-level common database.

Conceptual Model Generation Framework

This section proposes a framework for conceptual modeling generation. The framework can be used to generate a conceptual model from scratch or augment an existing one. It is assumed that the domain ontology is expressed using the primitives outlined above.

Creating an E-R Model from Scratch

The proposed framework for generating an E-R model from scratch is shown in Figure 3. It is comprised of the following steps: (1) identify initial user terms, (2) map the initial terms to domain terms in the ontology and expand the model, (3) check for consistency, (4) generate the complete E-R model, and (5) map to a relational model. These steps are described below.

(1) *Identification of Initial Terms*: Identify the key terms and concepts relevant to an application based upon the user's input. The database designer can provide specific application requirements or a textual description of the problem domain and the desired functionalities from which appropriate terms are identified. Simple natural language processing techniques can be used to identify relevant terms from the user descriptions. Alternatively, if the user is familiar with the application domain, he or she can provide appropriate terms directly.

(2) *Map to domain terms and expand list of terms*: Once the initial set of terms is identified, the terms are compared to those in the ontology for that domain. The "synonym" and "is_a" relationships from the ontology are used to verify the appropriate use of the terms in the design. The initial set of terms is also expanded using the "related-to" relationship. This identifies additional terms in which the designer may be interested but has not articulated.

(3) *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 constraints, (1) prerequiste, (2) temporal, (3) mutually-inclusive, and (4) mutually-exclusive, are used to ensure that all of the relevant terms and concepts have been included.

(4) *Generate E-R model*: Based on the terms identified in the previous step, entities are created. Relationships between entities are generated by taking into account the application domain semantics incorporated within the ontology. Previously generated E-R models and the functionalities supported by these models are stored in a repository. This repository is also used in identifying commonly occurring relationships within a particular application domain, and this information is also used in generating the relationships between entities. Thus, at the end of this step, an overall E-R model is generated and presented to the user for feedback.

(5) *Map to Relational Model*: Once the E-R model is created, the corresponding relational data model is generated using the well established rules of mapping from E-R to relational databases (Teorey et al. 1986).

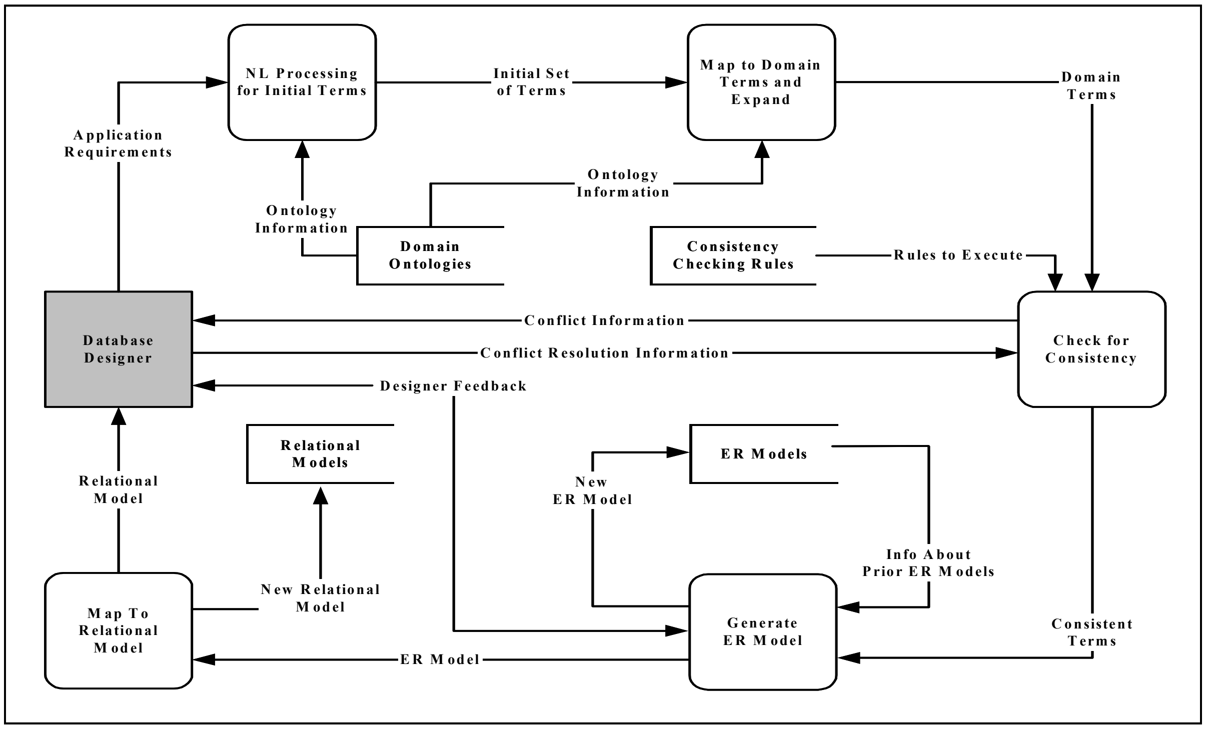


Figure 3. Framework for Creating E-R Model from Scratch

Validation of Existing E-R Model

The framework supports validation of an existing E-R model by checking whether it includes some of the generic entities that are typically part of similar designs in that application domain. The model can also be checked to assess if the entities and relationships are internally consistent. The E-R validation framework is shown in Figure 4 and consists of the following: (1) check appropriateness of terms, (2) identify missing terms, and (3) generate augmented E-R model.

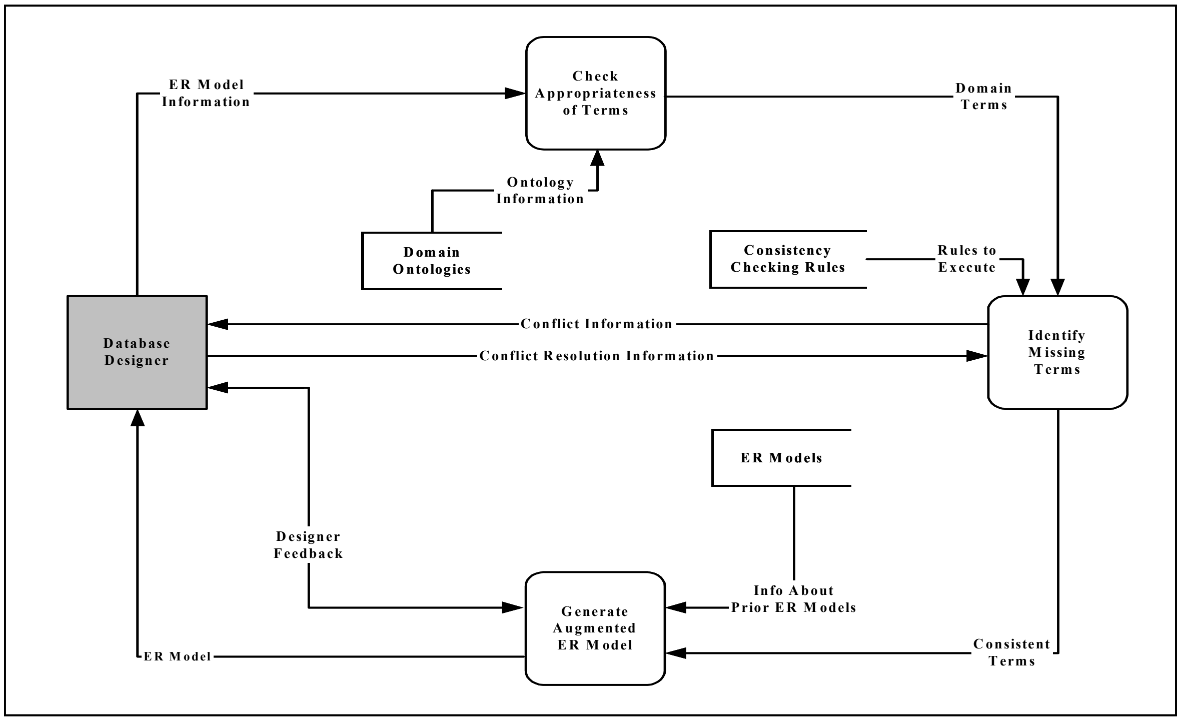


Figure 4. Framework for Validating an Existing E-R Model

(1) *Appropriateness of Terms*: The database designer provides information related to the E-R model. Based on this information, the terms used to name the entities are checked for appropriateness using the domain knowledge contained in the ontology.

(2) *Identifying Missing Terms*: For the terms captured in the E-R model, if "is_a" or "related_to" relationships exist, use them to gather related terms. The constraints defined in the ontology for the terms used in the E-R model are also enforced to ensure that they are internally consistent.

(3) *Augmenting the E-R Model*: For the terms identified in the previous step, appropriate relationships are also suggested based on the knowledge culled from previous E-R models stored in the E-R model repository. With the designer's feedback, an augmented E-R model is generated that meets the requirements specified by the designer.

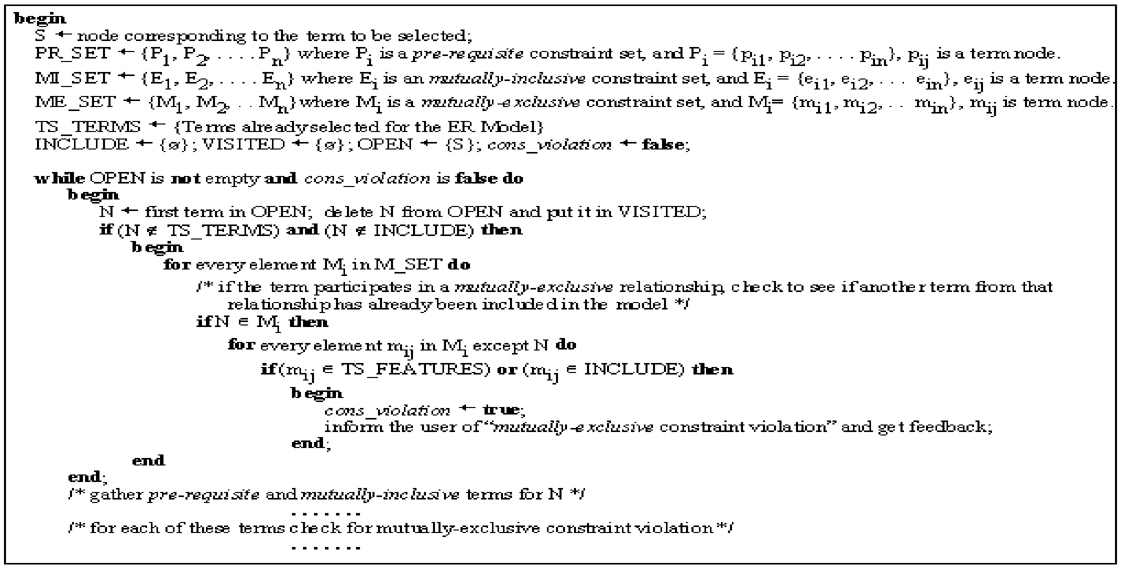
Consistency Checking

The structure of the ontology contains various primitives to express different types of relationships and constraints that exist between terms or concepts within an application domain. The constraints feature supported by the ontology allows one to express the application domain semantics and business rules. For example, in the online auction domain, a bid needs (prerequiste) item, account, and bidder. The temporal constraint for bid indicates 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.

When trying to identify or add new terms (entities) to an E-R model, or remove a particular term from an E-R model, the constraints are utilized to check for consistency. For example, when adding a new term, if there exists prerequisite, or mutually inclusive terms, those terms must also be included. In the auction domain, the mutually inclusive constraint for bid implies that, whenever bid is included in a design, the associated terms item, buyer, and seller should also be included. Similarly, when deleting a particular term, if other terms depend upon this term, it should not be deleted. The ontology can be represented as a graph, where the nodes correspond to terms and the arcs to relationships and constraints. While selecting or deleting a term, this graph can be utilized to ensure that the selection or deletion action results in a consistent set of terms. As part of the framework, heuristics perform consistency checking while selecting or removing a term. These heuristics have then been translated into production rules and stored in the knowledge base. They are briefly described below.

Heuristic for Term Selection

This heuristic primarily focuses on identifying other required terms that must be included as a result of selecting a particular term. In other words, when a particular term is to be selected, and there is a prerequiste constraint for that term, those terms must also be selected. If the same term has a mutually inclusive constraint, those mutually inclusive terms must also be selected. On the other hand, if the term has a mutually exclusive constraint, those mutually exclusive terms should not be part of the solution, and hence must be deleted. A portion of the constraint enforcement heuristic for term selection is shown in Figure 5.



```
begin
  S ← node corresponding to the term to be selected;
  PR_SET ← {$P_1$, $P_2$, . . . . $P_n$} where $P_i$ is a pre-requisite constraint set, and $P_i$ = {$p_{i1}$, $p_{i2}$, . . . . $p_{in}$}, $p_{ij}$ is a term node.
  MI_SET ← {$E_1$, $E_2$, . . . . $E_n$} where $E_i$ is an mutually-inclusive constraint set, and $E_i$ = {$e_{i1}$, $e_{i2}$, . . . $e_{in}$}, $e_{ij}$ is a term node.
  ME_SET ← {$M_1$, $M_2$, . . $M_n$} where $M_i$ is a mutually-exclusive constraint set, and $M_i$= {$m_{i1}$, $m_{i2}$, . . $m_{in}$}, $m_{ij}$ is term node.
  TS_TERMS ← {Terms already selected for the ER Model}
  INCLUDE ← {ø}; VISITED ← {ø}; OPEN ← {S}; cons_violation ← false;

  while OPEN is not empty and cons_violation is false do
     begin
        N ← first term in OPEN;  delete N from OPEN and put it in VISITED;
        if (N ∉ TS_TERMS) and (N ∉ INCLUDE) then
           begin
              for every element $M_i$ in M_SET do
                 /* if the term participates in a mutually-exclusive relationship, check to see if another term from that
                    relationship has already been included in the model */
                 if N ∈ $M_i$ then
                    for every element $m_{ij}$ in $M_i$ except N do
                       if($m_{ij}$ ∈ TS_FEATURES) or ($m_{ij}$ ∈ INCLUDE) then
                          begin
                             cons_violation ← true;
                             inform the user of "mutually-exclusive constraint violation" and get feedback;
                          end;
           end
     end;
     /* gather pre-requisite and mutually-inclusive terms for N */
                          . . . . . . .
     /* for each of these terms check for mutually-exclusive constraint violation */
                          . . . . . . .
```

Figure 5. A Fragment from the Term (Entity) Selection Heuristic

Heuristic for Term Deletion

This heuristic ensures consistency when deleting a term from the solution. When a particular term is deleted, there cannot be any other terms that depend upon it. In other words, if the term to be deleted is a prerequiste or a mutually-inclusive term for another term that is currently included in the model, then that term should not be deleted. On the other hand, when a term is deleted, its prerequiste and mutually inclusive terms can also be deleted from the model provided no other terms require them. The constraint enforcement heuristic for term deletion is shown in Figure 6.

System Architecture

The ontology creation and conceptual modeling approach is implemented in the Ontology Management and Database Design Environment (OMDDE). Its architecture, shown in Figure 7, follows the traditional three-tier client-server architecture with an HTML client, a Web server that supports java technologies, and a relational database. The client is a basic Web browser used to browse or modify existing ontologies, or create new ones. Using these ontologies, a user can also create and validate E-R models for database design. This server side consists of the following components: (1) an ontology management component, (2) a database design component, and (3) repositories.

```
begin
  D ← node corresponding to the term to be deleted;
  TS_TERMS ← {Terms already selected for the target system}
  DELETE ← {ø}; VISITED ← {ø}; OPEN ← {ø};
  req_violation ← false;

  /* gather all the terms that require the term to be deleted and check if any of those terms are selected for the model */
  generate the set P of predecessors (terms requiring D) of D;
  if (P ? {ø}) then
      for every element $p_i$ in P do
          if ($p_i$ ∈ TS_TERMS) then
              begin
                  if req_violation is false then req_violation ← true;
                  inform the user of "pre-requisite constraint violation" and get feedback;
              end;

      /* if none of the terms selected for the model requires this term, it can be deleted from the model */
  if req_violation is false then
              append D to DELETE;

  /* if a term is deleted, check to see if other terms that were automatically included because of this term can also be deleted */
               .......
```

Figure 6. A Fragment from the Term (Entity) Deletion Heuristic

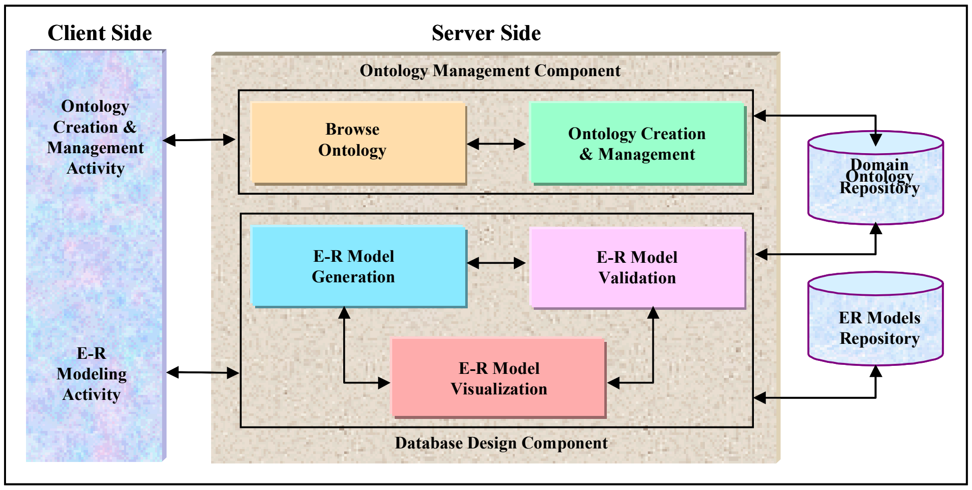

Client Side
Server Side
Ontology Management Component
Ontology Creation & Management Activity
Browse Ontology
Ontology Creation & Management
Domain Ontology Repository
E-R Model Generation
E-R Model Validation
ER Models Repository
E-R Modeling Activity
E-R Model Visualization
Database Design Component

Figure 7. Ontology Management and Database Design Environment Architecture

Ontology Management Component

This component facilitates the creation, storage, and use of ontologies. Domain analysts can create ontologies independently and store them in the domain ontology repository. The ontology management component consists of (1) a browse ontology module and (2) an ontology creation and management module.

Browse Module: Since ontologies contain a great deal of information about a domain, a novice user can browse an ontology to gain an understanding of the domain. 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.

Ontology Creation and Management Module: This module assists the user in creating or modifying an ontology. It facilitates the basic steps needed for creating an ontology and the management and evolution of existing ontologies. As new information is discovered about a specific application domain, the corresponding ontologies are extended to incorporate new knowledge. This module also executes consistency checking rules to ensure that the components of the ontology are consistent with each other. During ontology creation and evolution, new pieces are added to an existing ontology. If the new fragments do not produce a violation, the change is accepted. For example, if the user adds a new term and its defining constraints, the consistency checking module ensures that the new constraints do not conflict with existing ones.

Database Design Component

E-R Model Generation: Based on the requirements specified for the database application, appropriate terms and relationships are identified and checked for consistency. This module then generates the E-R model and passes it to the visualization module for displaying the final E-R model. This module can also create fragments of the overall E-R model to be provided to the user for closer examination.

E-R Model Validation Module: This module contains rules for consistency checking while selecting or deleting constructs from the E-R model. It is primarily responsible for checking the consistency and comprehensiveness of the generated E-R model.

Browse/Visualization Module: While creating an E-R model, at any point in time, the user can view the E-R diagram that has been created up to that point.

Repositories

The database design assistant has two knowledge repositories: (1) domain ontology repository and (2) E-R models repository.

Domain Ontology Repository: The domain ontology repository consists of ontologies for various application domains. The user can browse these ontologies to learn more about that application domain. These ontologies may be created by different domain analysts or stored in a library and may evolve over time.

E-R Models Repository: This repository contains E-R models that have been created over time. For each model, meta information about the functionalities supported by the model is also stored. By examining the E-R models in a domain and corresponding functional requirements, generic relationships that exist between entities can be identified. This information can be used to create a new E-R diagram for an application within that domain with similar entities and functional requirements.

Prototype Implementation

A prototype of the Ontology Management and Database Design Environment (OMDDE) has been developed using the java technologies, Servlet/JSP (Java Server Pages), and Jess, a java-based expert system shell (Friedman-Hill 2002). This development environment was chosen because it would make the system easily usable on the World Wide Web. The Model-View-Controller (MVC2) architecture is used, which separates core 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 make use of Jess rules and facts that are stored in the knowledge-base server. The visualization module is implemented using JSP technology. The E-R model generation and validation activities utilize 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. Consistency checking and E-R model generation are accomplished through rules written in Jess that use domain specific knowledge in the ontology and the E-R model repositories.

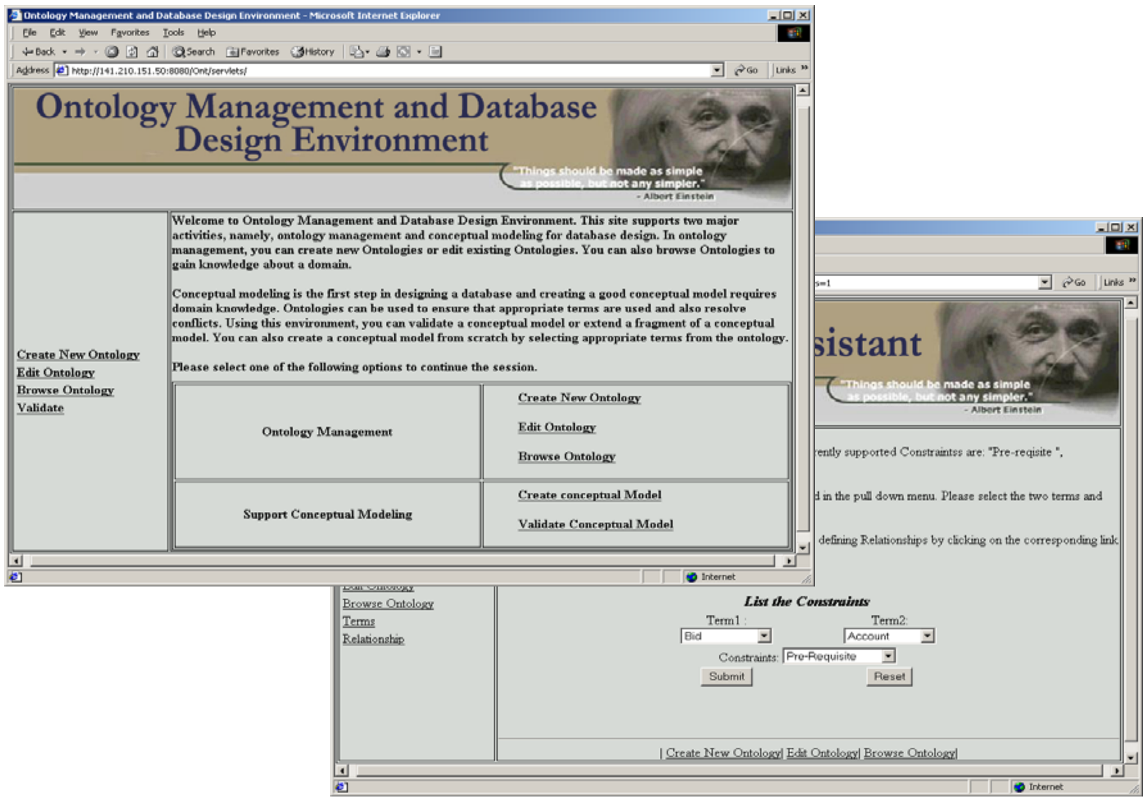
Sample Session

This section describes a sample session with the OMDDE prototype to illustrate the creation of an ontology and how that ontology can be used to generate and validate an E-R model. Upon accessing OMDDE, the initial screen, shown in Figure 8, is presented. The user (database designer) selects ontology management functions or conceptual modeling functions.

As shown in Figure 8, OMDDE supports the following ontology management activities: (1) creating a new ontology, (2) modifying an existing ontology, and (3) browsing the ontologies within a particular domain. The conceptual modeling functions supported by OMDDE enable the user to either (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 missing constructs.

Ontology Creation and Management Activities

The ontology management functions of the OMDDE system are implemented within the Ontology Management Assistant subsystem. The conceptual modeling (E-R model) functions are implemented within the Database Design Assistant subsystem. When the user selects Create New Ontology, the system prompts him or her for the domain and ontology name. When the user submits this information, the system presents another screen that contains the links: Define Terms, Define Relationships, and Define Constraints. Using these links, the user can define terms for the ontology and establish relationships and constraints between terms. Currently, the user interface supports only binary relationships and constraints. The system presents the terms that have been added to the ontology and the user can select two terms to establish a relationship or constraint between them. The relationships currently supported are "is_a," "synonym," and "related_to"; constraints supported are prerequiste, mutually inclusive, mutually exclusive, and temporal. When a new relationship or constraint is created, consistency-checking rules are enforced to ensure that relationships and constraints are consistent.



Ontology Management and Database Design Environment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://141.210.151.50:8080/Ont/servlets/
Go Links
Ontology Management and Database Design Environment
"Things should be made as simple as possible, but not any simpler." - Albert Einstein

Welcome to Ontology Management and Database Design Environment. This site supports two major activities, namely, ontology management and conceptual modeling for database design. In ontology management, you can create new Ontologies or edit existing Ontologies. You can also browse Ontologies to gain knowledge about a domain.

Conceptual modeling is the first step in designing a database and creating a good conceptual model requires domain knowledge. Ontologies can be used to ensure that appropriate terms are used and also resolve conflicts. Using this environment, you can validate a conceptual model or extend a fragment of a conceptual model. You can also create a conceptual model from scratch by selecting appropriate terms from the ontology.

Please select one of the following options to continue the session.

Ontology Management	Create New Ontology Edit Ontology Browse Ontology
Support Conceptual Modeling	Create conceptual Model Validate Conceptual Model

Create New Ontology
Edit Ontology
Browse Ontology
Validate
Internet
sistant
"Things should be made as simple as possible, but not any simpler." - Albert Einstein

rently supported Constraintss are: "Pre-reqisite ",

d in the pull down menu. Please select the two terms and

defining Relationships by clicking on the corresponding link

List the Constraints

Term1 : Bid
Term2: Account
Constraints: Pre-Requisite

Figure 8. OMDDE Initial Screen and Defining Constraints between Terms

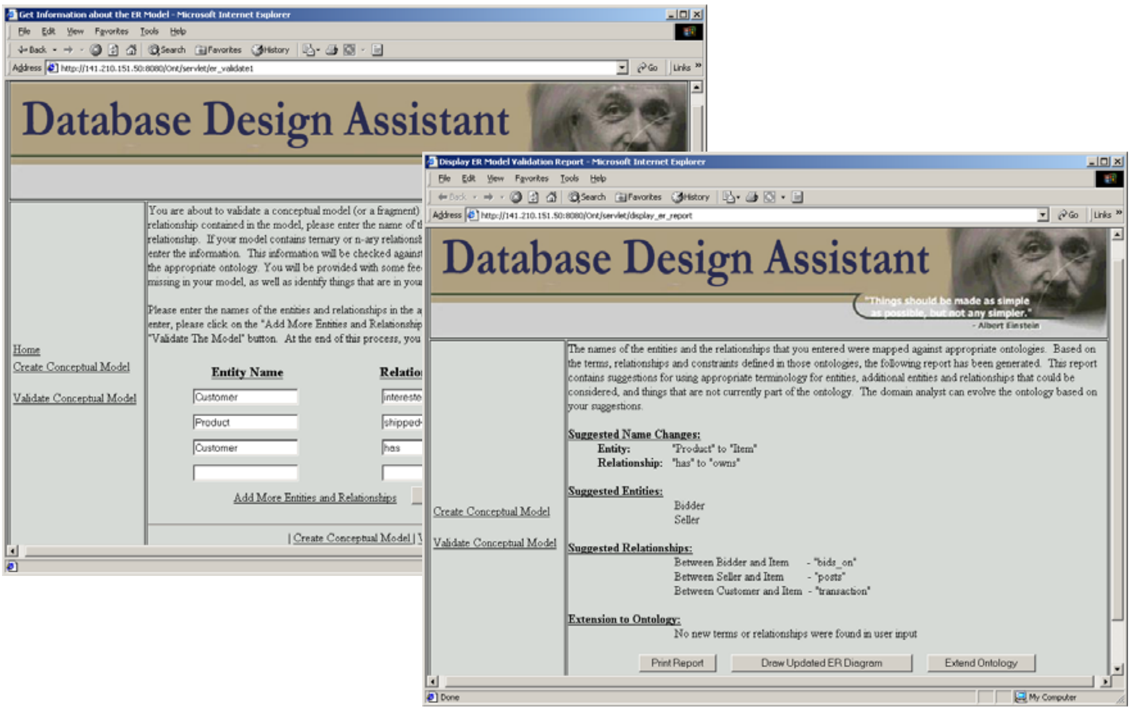
When the user selects the the Edit Ontology link (Figure 8), OMDDE presents the names of ontologies that currently exist within each application domain and prompts the user to select the ontology to be modified. When a particular ontology is selected for modification, all aspects of the ontology can be updated and the system ensures that the changes are consistent. To modify an existing ontology, OMDDE presents a similar set of screens that were used in ontology creation for updating terms, relationships, and constraints.

The user can browse existing ontologies by selecting the Browse Ontology link as shown in Figure 8. After selecting a particular domain, the names of the ontologies within that domain are presented, from which the user can select one or all to browse. Specifically, the user can view the terms as well as the relationships and the constraints that are part of that ontology. Alternatively, when the terms are presented as hyper links, the user can click on a particular term to obtain its description and the relationships and constraints that exist for that term.

Conceptual Modeling Activities

When the user undertakes conceptual modeling activities, the Database Design Assistant subsystem controls the user interactions. Using this subsystem, the user can create a new conceptual model (E-R model) or augment and validate an existing conceptual model fragment. The panel shown in Figure 8 contains links for these options. The database designer can explicitly check the validity of the model in terms of whether the model uses appropriate terms and relationships and that it is not missing any major concepts. This is accomplished by mapping the entities and relationships of the current E-R model to the domain ontology and by enforcing the constraints and relationships through consistency checking rules. When starting with an existing E-R model (or a fragment), the designer can provide the names of the entities and the relationships that are part of this model. The user provides this information for every binary relationship that exists within the model, and the system recreates the complete E-R model. In a future version, the designer will be able to provide the entire E-R model in a predefined format. Once the E-R model information is provided, the system checks for missing and superfluous entities and relationships and presents a report to the user.

For every binary relationship in the model, the user provides the name of the relationship and the entities that participate in it. This interface is shown in Figure 9. The designer can enter additional binary relationships by clicking the Add More Entities and Relationships hyperlink, or start the validation process by clicking on Validate the Model button. At the end of the validation process, the system generates a report that contains information about suggested new entities, new relationships, name changes for entities and relationships, etc. As shown in Figure 9, the validation report for the current E-R diagram suggests adding Bidder and Seller entities, and three other relationships. The designer can print this report and draw the updated E-R diagram by clicking on the appropriate buttons. If the system finds terms that are not part of the ontology, this information is captured and sent to the domain analyst for extending the ontology.


Get Information about the ER Model - Microsoft Internet Explorer

http://141.210.151.50:8080/Ont/servlet/er_validate1

Database Design Assistant

Home
Create Conceptual Model
Validate Conceptual Model

You are about to validate a conceptual model (or a fragment) relationship contained in the model, please enter the name of t relationship. If your model contains ternary or n-ary relations enter the information. This information will be checked agains the appropriate ontology. You will be provided with some fee missing in your model, as well as identify things that are in you

Please enter the names of the entities and relationships in the a enter, please click on the "Add More Entities and Relationship "Validate The Model" button. At the end of this process, you

Entity Name	Relatio
Customer	intereste
Product	shipped-
Customer	has

Add More Entities and Relationships

| Create Conceptual Model | V

Display ER Model Validation Report - Microsoft Internet Explorer

http://141.210.151.50:8080/Ont/servlet/display_er_report

Database Design Assistant

"Things should be made as simple as possible, but not any simpler." - Albert Einstein

Create Conceptual Model
Validate Conceptual Model

The names of the entities and the relationships that you entered were mapped against appropriate ontologies. Based on the terms, relationships and constraints defined in those ontologies, the following report has been generated. This report contains suggestions for using appropriate terminology for entities, additional entities and relationships that could be considered, and things that are not currently part of the ontology. The domain analyst can evolve the ontology based on your suggestions.

Suggested Name Changes:
Entity: "Product" to "Item"
Relationship: "has" to "owns"

Suggested Entities:
Bidder
Seller

Suggested Relationships:
Between Bidder and Item - "bids_on"
Between Seller and Item - "posts"
Between Customer and Item - "transaction"

Extension to Ontology:
No new terms or relationships were found in user input

Figure 9. Gathering E-R Model Information and Validation Feedback

Assessment

Three subjects were asked to play the role of database analyst and use OMDDE for both creating an initial E-R model and validating it. One of the subjects was a professor of information systems, and the other two subjects were systems analysts with 6 and 10 years experience. The primary objective of the validation was to get feedback on the usefulness of the system for database design as well as the reasoning process embedded within the system. The following criteria were used: (1) face *validity* (a subjective evaluation of the performance of the system by people knowledgeable about the application domain), (2) *visual* *interaction* (interact with the system and provide face validity by visual interaction), (3) *construct validity* (theoretical base underlying the system under examination), and (4) *objectivity* (independent validators examining the system to remove bias in the evaluation). Although the number of subjects was small, the feedback provided showed the usefulness of the approach.

The subjects had considerable background in conceptual modeling and database design, but they were not familiar with ontologies and the role ontologies can play in conceptual modeling and database design. Hence, the subjects were provided with some training on ontologies and how they are used in database design. They were also familiarized with the OMDDE system by having them practice with an example.

The evaluation of the system focused on independently creating E-R models and validating them against the ontology. Each of the subjects was asked to create an E-R model in two different domains: online auction and travel. A brief description of the desired functionalities of the database application was provided and the subjects were instructed to consider typical user requirements for the auction and the travel domain. The subjects were also asked to provide written comments about the ease of use and usefulness of the system.

First the subjects interacted with the OMDDE system and created an E-R model. Then they were instructed to use the model validation feature of the system to validate their model for completeness and consistency. During the session, the subjects were also asked to browse the auction and travel ontologies and provide feedback on the content and usefulness of the artifacts (terms, relationships, and constraints). During E-R model validation, OMDDE makes suggestions for entities and relationships to be incorporated into the model. Thus, a comparison of the original E-R model created by the subjects and the suggested artifacts is made. Validating the user E-R model includes determining the usefulness of these new artifacts and the appropriateness of the suggested name changes for entities and relationships. The subjects were also asked to comment on the ease of use of the interface of the system and their impressions of how well the system would assist them in conceptual modeling and database design.

Excerpts from the panel's comments are provided in Figure 10. From the feedback, it is evident that the system performed reasonably well in validating the E-R model and suggesting new entities and relationships. It was also able to identify certain artifacts that need to be added to the ontology. 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.

"The ontology provides a structured way to store domain knowledge using terms, relationships and constraints. It can be used as a reference point for conceptual modeling and the tool clearly facilitates validating an E-R model. Such a tool is useful in ensuring that proper domain terminologies are used as well as infusing business rules. Further work is needed in making the tool a bit more flexible while validating the E-R model."
"The tool is easy to use for creating E-R models and then validating them. Ontology provides a useful way to look at different aspects of a domain and relate it to application development. Currently, the tool expects the user to input information about the E-R model in a certain format. May be a graphical interface where the user draws an E-R diagram and then the system validating it would be attractive. As is, it is a useful analysis tool."
"The tool provides a systematic way to capture information about a domain. This information could be used by a novice user in creating a conceptual model in that domain, particularly, if the user doesn't have any prior knowledge about that domain. This tool could be useful in incorporating best practices while designing a database application."

Figure 10. Comments from OMDDE Evaluation

Conclusion

This paper has demonstrated how domain knowledge, stored in an ontology, can be used to assist in the generation of more 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 and implemented. This framework utilizes ontology primitives such as relationships and constraints between terms to generate a consistent design. The prototype incorporates consistency-checking rules while selecting or deleting terms from the current conceptual model. Further work is needed to complete the prototype and assess its effectiveness for a variety of design tasks. The initial results using the prototype demonstrate the feasibility of the framework.

References

Alter, S. *Information Systems: A Management Perspectiv*e, Addison-Wesley, Boston, MA, 1999.

Bergholtz, M., and Johannesson, "Classifying the Semantics of Relationships in Conceptual Modeling by Categorization of Roles," in *Proceedings of the Sixth International Workshop on Applications of Natural Language to Information Systems,* A. M. Moreno and R. P. van de Reit (eds.), Madrid, June 28-29, 2001, pp. 199-203.

Berners-Lee, T., Hendler, T. J., and Lassila, O. "The Semantic Web," *Scientific American* (284:5), May 2001, pp. 34-43.

Bodart, F., Pate, A., Sim, M., and Weber, R. "Should Optional Properties be Used in Conceptual Modeling? A Theory and Three Empirical Tests," *Information Systems Research* (12:4), 2002, pp. 384-405.

Dahlgren, K. "A Linguistic Ontology," *International Journal of Human-Computer Studies* (43), 1995, pp. 809-818.

Dennis, A., and Wixom, B. H. *Systems Analysis and Design*, John Wiley, New York, 2000.

Embley, D., Campbell, D. M., Jiang, Y. S., Ng, Y. K., Smith, R. D., Liddle, S. W., and Quass, D. W. "A Conceptual-Modeling Approach to Web Data Eextraction," *Data & Knowledge Engineering* (31:3), 1999, pp. 225-305.

Farquhar, A., Fikes, R., and Rice, J. "The Ontolingua Server: A Tool for Collaborative Ontology Construction," Technical Report KSL-54-96-26, Knowledge Systems Laboratory, Stanford University, 2002.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., and Patel-Schneider, P. F. "OIL: An Ontology Infrastructure for the Semantic Web," *IEEE Intelligent Systems* (16:1), 2001, pp. 38-45.

Friedman-Hill, E. "Jess: The Expert System Shell," Sandia National Laboratories, Livermore, CA, 2002 (available online at **http://herzberg.ca.sandia.gov/jess**).

Gruber, T. R. "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition* (5), 1993, pp. 199-220.

Heflin, J., and Hendler, J. "Dynamic Ontologies on the Web," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000*), AAAI/MIT Press, Menlo Park, CA, 2000, pp. 443-449 (available online at **http://www.ksl.stanford.edu/software/ontolingua/**).

Kedad, Z., and Metais, E. "Dealing with Semantic Heterogeneity During Data Integration," in *Conceptual Modeling: ER'99, Eighteenth International Conference on Conceptual Modeling, Lecture Notes in Computer Science 1728*, J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Metais (eds.), Paris, France, November 15-18, 1999, pp. 325-339.

Lloyd-Williams, M. "Exploiting Domain Knowledge During the Automated Design of ObjectOriented Databases," in *Proceedings of the Sixteenth International Conference on Conceptual Modeling, Lecture Notes in Computer Science 1331*, D. W. Embley and R. C. Goldstein (eds.), Los Angeles, November 3-6, 1997.

Mason, O., and Grove-Stephensen, I. "Automated Free Text Marking with Paperless School," in *Proceedings of Sixth International Computer Assisted Assessment Conference*, M. Danson (ed.), Loughborough, July 9-10, 2002, pp. 213-219.

McGuiness, D. L., Fikes, R., Rice, J., and Wilder, S. "An Environment for Merging and Testing Large Ontologies," in *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, A. G. Cohn, F. Giunchiglia, and B. Selman (eds.), Breckenridge, CO, 2000, pp. 483-493.

Noah, S. A., and Williams, M. D. "Knowledge-Based Approaches to Database Design Diagnosis: Improving Performance with a Domain Specific Thesaurus Structure," in *ACI 2002: Proceedings of the 2002 IASTED International Conference on Artificial and Computational Intelligence*, N. Ishii (ed.), ACTA Press, Calgary, Canada, 2002, pp. 366-371. .

Parsons, J., and Wand, Y. "Emancipating Instances from the Tyranny of Classes in Information Modeling," *ACM Transactions on Database Systems* (25:2), 2000, pp. 228-268.

Shanks, G., Tansley, E., Nurelini, J., Toblin, D., and Weber, R. "Representing Part-Whole Relationships in Conceptual Modeling: An Empirical Evaluation," in *Proceedings of the Twenty-Third International Conference on Information Systems*, L. Applegate, R. Galliers, and J. I. DeGross (eds.), Barcelona, Spain, December 15-18, 2002, pp. 89-100.

Siau, K., Wand, Y., and Benbasat, I. "The Relative Importance of Structural Constraints and Surface Semantics in Information Modeling," *Information Systems* (22:23), 1997, pp. 155-170.

Sugumaran, V., and Storey, V. "Ontologies for Conceptual Modeling: Their Creation, Use, and Management," *Data and Knowledge Engineering* (42:3), 2002, pp. 251- 271.

Swartout, W. "Ontologies," *IEEE Intelligent Systems* (14:1), January-February, 1999, pp. 18-19.

Teorey, T. J., Yang, D., and Fry, J. P. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," *ACM Computing Surveys* (18:2), 1986, pp. 197-222.

Weber, R. "Are Attributes Entities? A Study of Database Designers' Memory Structures," *Information Systems Research* (7:2), June 1996, pp. 137-162.

Weber, R. "Ontological Issues in Accounting Information Systems," in *Researching Accounting as an Information Systems Discipline*, S. Sutton and V. Arnold (eds.), American Accounting Association, Sarasota, FL, 2002.

Welty, C., and Guarino, N. "Supporting Ontological Analysis of Taxonomic Relationships," in *Special Issue on ER 2000, Data and Knowledge Engineering* (39:1), 2001, pp. 51-74.