From ER to Ontology with Natural Language Text Generation

Csaba Veres[1], Jennifer Sampson[2], Clare Atkins[3]

[1,2]Department of Computer Science, The Norwegian University of Science and Technology, Trondheim, Norway
{Csaba.Veres, sampsonj}@idi.ntnu.no
[3]School of Business and Computer Technology Nelson Marlborough Institute of Technology Nelson, New Zealand
catkins@nmit.ac.nz

Abstract. We describe the automation of a novel technique (NaLER) which was originally designed to facilitate legacy database model validation. The NaLER technique uses natural language sentences built from live database content to elicit validation judgments from domain experts. However, during implementation we discovered that the method we had adopted for the automation had a serendipitous side effect in that the legacy model first had to be mapped to an upper ontology. This normally difficult process was significantly eased by the sentence templates which are defined as part of the NaLER technique. It is this novel process of mapping, and the choice of ontology it entails, which forms the focus of the paper. We therefore describe here the process of mapping to the upper model, and investigate how the motivation for modeling impacted on the choice of modeling language. Finally we describe the prototype of a tool and how it fits with the development methodology.

1 Introduction

The NaLER (Natural Language for ER) [2] method is designed to help with data schema validation by exposing users to natural language expressions about instance data, allowing them to comprehend the way actual data is represented in a live database. Since database schemas represent agreed meanings about the data, user validated data models can be used as a valuable input to the creation of a consensual ontology for corporate knowledge management. Mapping a database to an ontology may be necessary when an organisation wishes to define a common understanding of the structure of all data sources in the domain. The resulting ontology conceptualises and structures the domain knowledge within a community of interest. Ontologies are useful for making explicit logical statements about a domain and for reasoning about those statements.

Creating a quality ontology based on legacy databases can be difficult, especially when the underlying business rules are incorrect or no longer valid. We suggest that the NaLER method for maintenance and verification is useful in

clarifying business rules that are important in ontology creation. With this assumption in mind we describe first the techniques that help ontology engineers revisit and verify business rules represented in the legacy data model, and second we provide a method and implementation for mapping a relational model represented by an entity relationship model or extended entity relationship model (E-R/R) to the Penman Upper Model [5], an abstract domain independent upper ontology which we argue is ideal given the requirements of the task. We will discuss this upper model more fully in subsequent sections.

Originally developed to fill a gap in the relational database design process [1], NaLER is an appropriate tool for identifying the semantics of existing data structures. [12] suggests that a major issue is that "...[d]ata models...are so poor at 'capturing the meaning' and people are so effective at intuitively accommodating these weaknesses that [people] project meaning onto the data structures rather than abstracting meaning out of them". NaLER was intended to provide a formal way of capturing the meaning represented by the database constructs and thus to reduce, if not eliminate, the assumptions and individual interpretations that can be projected on to them. Although several E-R/R methods have created mechanisms for presenting model information as natural language sentences (e.g. [10]; [3]), they are concerned primarily with interpreting the relationships between entities. NaLER extends the principle by requiring more detail in these 'relationship' sentences, specifically primary key information, and by constructing such sentences for all the objects in the data model [2]. The construction of NaLER sentences can also assist in determining semantic equivalence between two different structural representations together with their correspondence to 'facts' in the universe of discourse that they are describing, a necessary pre-requisite to the successful mapping of pre-existing data structures to an ontology. [7] suggested that "two data bases are equivalent if they represent equivalent facts about a certain slice of reality." However as they also conclude, this necessitates relying on "...a common understanding of natural language". Any reliance on natural language interpretations of such structures while increasing accessibility will also introduce ambiguity and thus lead once again to the situation described by [12].

To assist in minimising this problem, NaLER, drawing on the NIAM-CSDP [11] method which it was designed to complement, advocates the use of 'example data' to instantiate the sentences for the purpose of semantic validation. As a database design aid, it was intended that the examples used would be those collected during the initial stages of requirements analysis. However, to create a meaningful description of existing database objects, data stored within those objects would be required. This technique, a form of NIAM-CSDP in reverse was tested by [13] and found to be effective in providing a means by which users could evaluate the validity of their implemented databases. Use of the NaLER technique thus provides for the creation of a set of natural language example sentences, of minimised ambiguity, using the terms of the original data model, the names of existing database objects and sets of data extracted from these objects. These sentences can thus be used to both validate the data structures

themselves and to check for consistency with the universe of discourse and other structural representations of it. As such we believe that it can facilitate the mapping of a data model to an ontology and its integration with other ontologies.

Exposure of NaLER as a technique has been limited because up until now NaLER was not implemented in a tool, requiring that the large number of exemplar sentences be manually constructed. The current work began with an implementation of NaLER with the aim to allow the full potential of the technique to be realised. But instead of attempting a somewhat mindless, template based algorithm for generating sentences from instance data, the implementation we decided on is novel in that it used a flexible, domain independent ontology-based system for sentence generation in order to produce the sentences. This resulted in a two-step process. First the data model is mapped to an upper ontology, which is then used to generate sentences of the instance data by the system. The first step is performed by a database administrator, who can use the mapping phase to perform an initial verification on the data model, by noting if any of the mappings are suspect. The second phase is fully automated and outputs an arbitrary number of natural language sentences for domain expert validation. While this process may appear overly complex, we feel it adds two benefits. First, all legacy databases are mapped to the same upper model, thereby facilitating interoperability. Second, the legacy database is validated by end users before it is mapped to an ontology.

The paper is structured as follows. In the following section we outline the steps for generating the NaLER sentences. In section 3 we introduce a sentence generation system that will be used for producing the NaLER sentences. In this section we will also consider the ontology which forms our upper model. We argue that the ontology is both useful as an upper model and necessary given the goals of sentence generation. In section 4 we will discuss the process of mapping data models to the upper model. In section 5 we describe an implementation, and conclude in section 6.

2 NaLER steps

NaLER is designed for use with a relational data model, represented diagrammatically by an ER/EER diagram. (It is important to note that we are assuming the existence of a data model which corresponds to the database. In the absence of such a model, some amount of reverse engineering is clearly required before the NaLER method can be applied). The primary objective is to provide a way of translating relational data models into a format that can be verified by users. This is important on its own as the ability to understand and read accurately the information content of EER models, has a much wider application. It is a fundamental skill required by any person involved with EER models in almost any capacity. Not only the modellers themselves and the users whose requirements have been sought, but other end users, such as domain experts, auditors, systems analysts, database designers and administrators, also have a need to read a model. However, it can be notoriously difficult for non experts to read

ER models accurately, which in itself makes NaLER a useful technique. Table 1 summarizes the seven steps of the NaLER method (from [2]).

Table 1. The seven steps of the NaLER method.

Step	Description
1.	Document the model conventions
2.	Check the assumptions
3.	Simple entities
3.1	Construct the sentence for the primary key sentences
3.2	Construct attribute sentences
3.3	Construct relationship sentences
4	Construct super/sub-type sentences
5	Complex entities
5.1	Construct relationship sentences
5.2	Construct primary key sentences
5.3	Construct attribute sentences
6	Populate with examples
7	Produce the NaLER description

The following discussion will briefly describe the most relevant steps and illustrate using the example student data model fragment in figure 1. Note that the model is highly simplified and does not, for example, include entity attributes.

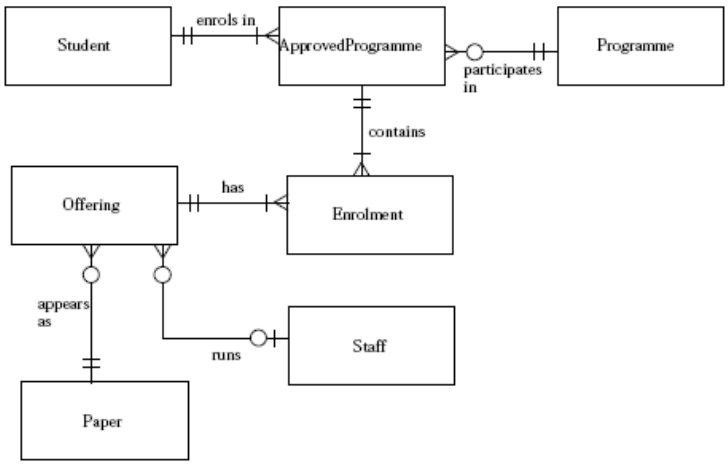

Fig. 1. Student Data Model to be used in the examples.

[1] describes a set of pre-requisites and assumptions for EER models with respect to the different CASE tools used. These assumptions are to be documented and checked in steps 1 and 2. An example of one such assumption is as follows: Relationships are optional unless clearly annotated as mandatory? (refer to [1] for a full list of model assumptions).

Below we illustrate some key steps in the process. Step 3 is concerned with extracting the sentences that relate to the simple entities within the model by completing three subtasks (note Sn stands for "Sentence number"). Step 4 concerns hierarchically related entities. Step 6 involes populating the abstract sentences with instance data.

3. For simple entities -
   (a) Construct the sentence for the primary key:
   Sn: Each $\langle E_{name} \rangle$ is uniquely identified by $\langle E_{primary\,key} \rangle$.
   Sentence 1: Each *Student* is uniquely identified by *Student no*.
   (b) Construct the sentence for the attributes:
   Sn: Each $\langle E_{name} \rangle (E_{primary\,key})$ must have only one $\langle E_{attribute\,name} \rangle$.
   Sentence 2: Each *Approved Program* (*Studentno*, *Program code*) must have only one *Head Approval*.
   (c) Construct the sentence for the relationships, so that for each binary relationship that the entity participates in, we construct two sentences.
   Sn: Each $\langle E_{1,name} \rangle (\langle E_{1,primary\,key} \rangle)$ $\langle R_{optionality} \rangle$ $\langle R_{name} \rangle$ $\langle R_{cardinality} \rangle$ $\langle E_{2,name} \rangle (\langle E_{2,primary\,key} \rangle)$.
   Sentence 3: Each *Student* (*student no*) *may enrol in many Approved Programs* (*Student no*, *Program code*).
   SnR: Each $\langle E_{2,name} \rangle (\langle E_{2,primary\,key} \rangle)$ $\langle R_{optionality} \rangle$ $\langle R_{name} \rangle$ $\langle R_{cardinality} \rangle$ $\langle E_{1,name} \rangle (\langle E_{1,primary\,key} \rangle)$.
   Sentence 3R: Each *Approved Program* (*Student no*, *Program code*) *must be enrolled by at least one student* (*student no*).
4. Construct super/sub-type sentences, so that for each subtype entity we construct a sentence as:
   Sn: Each $\langle Sub\,E_{name} \rangle (\langle E_{primary\,key} \rangle)$is a$\langle Super\,E_{name} \rangle (\langle E_{primary\,key} \rangle)$.
   Sentence 5: Each *Student* (*Student no*) is a *Person* (*Person Number*) (not shown in the model due to space constraints).
6. Populate the sentences with examples:
   This step is where the sentences are populated with valid examples. In some cases there is a paradigmatic change to reflect the instantiated nature of the sentences. For example sentence 3 becomes
   Student (111) *may enrol in Approved Program* (111, 666).

An important point to reiterate is that the domain experts are only expected to check and validate the natural language sentences produced in step 6. In the next sections we describe how we implement step 3, which needs input from a database administrator.

3 The KPML Sentence Generation System

The Komet-Penman Multilingual (KPML) [1] natural language generation system was originally designed as a tool for exploring natural language grammar, but also as a domain general system which could be used to provide plugin functionality for system developers who needed to add text generation capabilities to their applications. The key to this functionality is the Upper Model (UM), an interface ontology that mediates between the formal components that determine the grammatical expressions, and domain specific conceptual entities from independently constructed application ontologies. The basic procedure, then, is to find an appropriate UM concept to subsume each domain concept, which results in the domain concept inheriting the necessary features for text generation in the KPML system.

The UM is a linguistically motivated upper ontology that contains only concepts which have an impact on grammatical expression. [4] argues that the UM is suitable as a general upper ontology for organizing domain knowledge because it provides a theoretical framework for the ontology structure. But its framework is unique because the UM differs both from highly targeted domain specific ontologies and from abstract, domain independent upper ontologies. [4] argues that if different ontologies are constructed for very different and highly specific domains and reasoning tasks, then this results in highly diverse and possibly irreconcilable ontologies. Hence the need for some unifying framework. But large, general purpose ontologies that are meant to serve a diversity of domains and reasoning tasks are too underspecified for this task: the requirement to simply "represent the world" is not sufficient to construct a useful organization of the knowledge. The linguistic UM is offered as a solution because its construction derives from the most general but still formally specifiable task that is common to all domains: expression of knowledge about the domain in natural language. The interesting theoretical claim is that the concepts which are needed for generating natural language can unify the conceptual spaces of different subject domains. This seems intuitively plausible because natural language is clearly the most expressive general language for expressing domain facts, and its conceptual core a good candidate for a unifying ontology. It is this property that makes the UM a suitable foundation for the conversion of legacy data models into ontologies.

To the extent that upper ontologies are useful, there is considerable debate about which might be the most appropriate one to use. For example both SUMO[2] and Cyc[3] offer very comprehensive but differing ontologies with their corresponding upper ontology structures. And, while both can be used in natural language applications, neither has the feature that the ontology is a direct reflection of the possible linguistic expressions. In designing the current application we found that the directness of the mapping from the data model to the ontology to the natural language sentences made the implementation relatively straightforward.

[1] http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/README.html
[2] http://www.ontologyportal.org/
[3] http://www.cyc.com/

The Upper Model which is used in KPML 4.0 consists of 410 concepts organized in a single hierarchy. Part of the top level of this hierarchy is shown in figure 2.

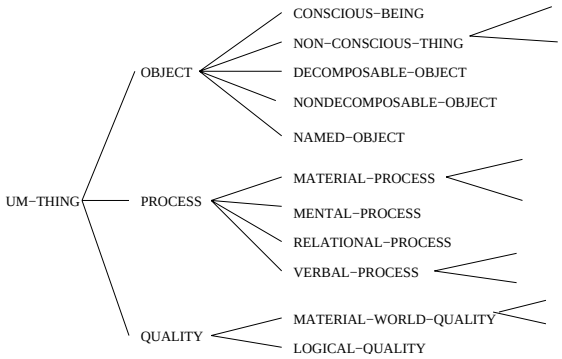

Fig. 2. Part of the Penman Upper Model

It may be noted that 410 concepts is relatively few for representing all possible world concepts. However, it is an empirical question whether or not it is sufficient to express required concepts in their veridical linguistic form, and whether such an ontology is useful for facilitating inter-operation between data models which have had subsumption relations defined against the same upper ontology. Of course, the reader should remember that the nodes which are formed in the ontology will include the properties that are transferred from the entity attributes, which will be important in questions of interoperability. Nevertheless, the interesting empirical and methodological point is that the choice of modeling constructs and methods is determined to a large extent by the requirements of the task.

One unusual feature of the ontology is the ubiquitous use of reification for modeling relationships between concepts. Inherently relational concepts like *behind*, *under*, *similar*, are modeled as concepts subsumed by *two-place-relation*. [5] justify this decision by noting that knowledge representation languages typically provide mechanisms for defining concepts and relations among them, but "... in Natural Language this distinction is often blurred, so that it is not always clear whether a concept or a relation should be used to represent a given property" [5] (p. 6). For example the spatial locating concept *behind* would normally be taken as a relation holding between two concepts. Instead, the upper model defines it as a regular concept by reifying the relation, then defining the *domain* and *range* as the participants in the relationship. This impacts on the way ER concepts are mapped to the UM since all terms are mapped to UM concepts. On the other hand this eliminates the pesky indeterminacy and confusion since

the modeler does not need to worry if a data model construct should be mapped to an ontological concept or to a relation.

4 Mapping Domain Concepts to the Upper Model

This phase of the process is responsible for ensuring that the correct sentences are generated, and for mapping the data model onto the upper ontology structure. As we will see, the sentence generation has two complementary roles. First, it generates canonical sentences for verification by domain experts. Second, the mapping necessary for generating the correct surface form of the expression actually guides the selection of the most appropriate ontology concept.

The mapping was conducted manually by first narrowing the possible UM concepts and then selecting the one that provided the closest matching output. The procedure then is to select a candidate UM concept as subsumer, generate the sentence, and compare the target with the generated sentence. If there is a mismatch, the candidate is refined, most commonly by choosing a super- or sub-class of the original candidate. The interesting part of this process is to use the generated linguistic string to arrive at the most suitable conceptual mapping. The assumption is that the empirically derived linguistic constraints on concept/text correspondence as embodied in the UM and KPML are to a large extent veridical. It is a matter of design principle that the ontological components of the Upper Model exist because they produce canonical surface forms for the available conceptual specifications. It therefore follows that we have identified the most acceptable ontological mapping when the generated form matches the target form[4].

As an example consider in figure 1. the relationship between *Staff* and *Offering*, expressed by the target NaLER sentence "*each Staff may run one or more Offering*". First we note that *Staff* is categorized simply because it is subsumed quite naturally by the UM concept *Person*, and there is no justification for trying the more specific concepts *female* and *male* that are defined in the UM. The relation in the model is named by a verb as typically prescribed [8], in this case *run*. Verbs in natural language introduce predicates that typically denote an action, occurrence, or a state of being [5], which gives us some clue about its ontological status. To determine which UM concept the relationship name falls under, we note that the initial division in the UM is between *object*, *process*, and *quality*. Of these, it is clear that *object* or *quality* are not appropriate, leaving *process* or one of its subclasses as the only possible subsumer. Further, since all relationships in the UM are reified, it turns out that every concept representing a relationship is itself subsumed by *process*, or more specifically by the UM

[4] This is not to claim that the Upper Model ontology is a universally "correct" ontology in some sense. On the contrary, like all empirical artifacts it undergoes constant evaluation and revision. Nevertheless this evaluation is theory driven and traceable.

[5] This is a more or less linguistically un-sophisticated description as commonly construed, e.g. in http://en.wikipedia.org/wiki/Verb. The more technical syntactically motivated definitions are, however, not necessary for our purposes.

concepts *one-place-relation* and *two-place-relation* which are subsumed by *process*. So naturally all relationships in ER diagrams should map to one of these concepts.

One strategy for finding the appropriate mapping is to begin with *process* or one of its most general subclasses representing a relation. In this case "running an Offering" seems like a process undertaken by the staff member so we begin with *process*. But this generates the surface string "*Each Staff may run*". Any mention of the *Offering* is suppressed because the process is assumed to be simply "running". In order to bring in the affected entity we descend two levels in the hierarchy to the concept *directed-action* which is the first to introduce an ACTEE role. This time KPML generates the sentence "*Each Staff may run at least one Offering.*" which is equivalent to the target sentence.

Finally, the entity "Offering" needs to be mapped to an appropriate UM concept. The astute reader may have realized that an initial assignment must have been made for "Offering" in order to generate the target sentences. In fact, this means that the choice of assignments is often iterative because the surface form is a function of all assignments. Ideally the assignments would be made in parallel, simultaneously satisfying all requirements. In this case the default assignment was the most general UM concept, *UM-THING*, which was obviously sufficient to produce the required target. However, we try other assignments because it is somewhat unsatisfying to remain with such a general one, and because we must ensure that the generated string will not be changed with more specific assignments. But it is not natural to construe of "Offering" as either an *object*, *process*, or *quality*, so we explore deeper in the ontology. We note that the *object* hierarchy is associated with the following somewhat non-committal gloss: "An entity which is not a process or a quality."[6] Object therefore subsumes concepts such as *ordered-set*, *time-interval*, and *non-conscious-thing*. Of these, the latter immediately subsumes *abstraction* which is a reasonable candidate to subsume "Offering". In making this assignment we find the target sentence generated as required.

Consider now a somewhat more difficult example with the relationship between student and ApprovedProgram: the target NaLER sentence is "*Each Student must enrol in at least one ApprovedProgram*". As in the previous example, *Student* is subsumed by the UM concept *Person*. But "enrol-in" is somewhat trickier. One possibility is to try the previously successful assignment to *directed-action*, where the "Student" acts on the "ApprovedProgram" (for now assigned to UM-THING). However this generates the following surface form: "*A Student enrols an ApprovedProgram.*" From this we see that a *directed-action* must describe an event in which the actor somehow causes something to happen to the actee. This was appropriate only in the previous example. A further option is to map to *nondirected-action* which are "... those material-actions which either have no 'actee', or whose 'actee' is not created or affected by the action" (LOOM

[6] Definitions can be found in the LOOM knowledge base available at http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/resources/merged-upper-model.zip

knowledge base). While this seems plausible, it creates a further problem because the "ApprovedProgram" can no longer be the actee. Browsing the UM ontology reveals that *spatio-temporal* relationships receive prominent representation, and suggests that an "ApprovedProgram" can be regarded as temporal entity which contains enrolments. This assignment generates the correct target sentence and has the interesting side effect that it alerts the data modeler that perhaps "ApprovedProgram" should have a property for the dates during which it was offered, which is another aspect for verification.

Let us briefly mention another interesting aspect to this method. It is of course entirely possible that the relationship in the original data model had been labeled with a different word like, say, *take*: "*Each Student must take at least one ApprovedProgram.*" In this case the assignment of "take" to directed-action does generate the NaLER sentence: "*Each STUDENT must take at least one ApprovedProgram.*" But the difference in the mappings to the UM concepts should alert the architect to some subtle differences in the possible interpretations of the documented models. We just argued that "ApprovedProgram" is best viewed as a sort of container for "Students", with possible attributes to reflect this role. The alternative designation, in analogy with "*Staff - runs - offering*", suggests a more active role in which the actor has a primary effect on the actee. Of these two alternatives the former seems more appropriate, showing that the structure of the ontology highlights possible design decisions and inconsistencies in the data model.

5 Implementation

The specification that mediates between the domain ontology, the UM and the rest of the KPML grammar and generation system is called the Sentence Plan Language (SPL). Figure 3 shows a simple specification for the logical component of the example "*Each Staff may run at least one Offering.*"

```
(EN / DIRECTED-ACTION :LEX run
:ACTOR (P1 / PERSON :LEX staff :set-totality-q total)
:ACTEE (P2 / abstraction :LEX offering :AT-LEAST 1)
:modality may )
```

Fig.3. An example SPL specification for the sentence "*Each Staff may run at least one Offering.*"

In this example the surface form of lexical items is directly introduced by :LEX in the SPL expression, and its UM type by the expression immediately preceding. But it is also possible to define a lexicon which contains this information, in which case only the word needs to be specified in the SPL.

Note that the implementation of the generation mechanism falls somewhat short of the intended goal, in that the SPL must often contain some information outside the UM to produce the desired surface form. The SPL in figure 3

is quite simple, and some examples require even more non-UM terms to generate the correct sentence. This requires that application developers should know something about the details of the KPML system. However in the current work we decided that these details should be hidden from the user as much as possible, and we are developing a comprehensive tool that will simplify the mapping process. Only some of the necessary functionality has been implemented at the point of writing.

6 Conclusion

In this paper we have outlined a system that can help automate the production of exemplar NaLER sentences from legacy databases. Once the ontological structure is determined the generation is fully automatic. This will help with validation of the data model. But the novelty of the implementation is in the required mapping to the UM ontology, which is a step towards converting a legacy data model into an ontology which exposes the full range of rules and relationships inherent in the database, thus facilitating the exchange of knowledge.

Acknowledgement

This work was sponsored by the Norwegian Research Council, WISEMOD project, 160126V30 in the IKT-2010 program, and the ADIS project in the same program.

References

1. Atkins, C. INTECoM: An integrated approach to the specification and design of information requirements. In Becker, S (ed.) *Designing Developing Quality Complex Database Systems: Practices, Techniques, and Technologies.* Ideas Group Publishing, Hershey P.A, 2001.
2. Atkins, C. and Patrick, J.P. NaLER: A natural language method for interpreting entity-relationship models. *Campus-Wide Information Systems* 17(3), 85-93, 2000.
3. Barker, R. *CASE*Method: Entity Relationship Modelling*, Addison Wesley, Wokingham, England, 1990.
4. Bateman, J. A. The Theoretical Status of Ontologies in Natural Language Processing. *Proceedings of the workshop on 'Text Representation and Domain Model ling : Ideas from Linguistics and AI'*, held at the Technical University Berlin, October 9th - 11th, 1991. KIT Report 97, edited by Susanne Preu and Birte Schmitz. Cmp-lg Paper No: cmp-lg/9704010
5. Bateman, J. A., Kasper, R., Moore, J. and Whitney, R. A general organization of knowledge for natural language processing: The Penman Upper Model. Technical Report. Information Sciences Institute. Marina del Rey, California, (1989).
6. Bergamaschi, S., Castano, S. De Capitani di Vimercati, S., Montanari, S., & Vincini, M. An Intelligent Approach to Information Integration. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 253-267. I0s Press, 1998.

7. Biller, H. and Neuhold, E.J. Semantics of data bases: the semantics of data models. *Information Systems,* Vol. 3, pp. 11-30, 1978.
8. Chen. P. English Sentence Structure and Entity-Relationship Diagram. *Information Sciences*, Vol. 1, No. 1, Elsevier, Pages 127-149, May, 1983.
9. Dou, D. and LePendu, P. Ontology-based Integration for Relational Databases, *SAC'06*, Dijon, France, April 23-27, 2006.
10. Finkelstein, C. *An Introduction to Information Engineering: ¿From Strategy Planning to Information Systems.* Addison Wesley, Sydney, 1989.
11. Halpin T. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design* Prentice Hall, London, 2001.
12. Marche, S. Measuring the stability of data models. *European Journal of Information Systems* Vol 2 no 1, 1992.
13. Ryder, M.R. *Facilitating Evolution in Relational Database Design: A procedure to evaluate and refine novice database designers schemata,* Unpublished thesis, Massey University, New Zealand, 1996.