Integrating STEP Schemata using Automatic Methods

Hercules Dalianis[1] and Eduard Hovy

Dept. of Computer and Systems Sciences
The Royal Institute of Technology (KTH)
and Stockholm University
Electrum 230
S-164 40 Kista, SWEDEN
ph. (+46) 8 16 49 16
email: hercules@dsv.su.se

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90292-6695
USA
ph. (+1) 310-822-1511 ext 731
email: hovy@isi.edu

Abstract

This paper describes a procedure to merge two STEP/AP EXPRESS schemata, using automated methods to assist the merging process. The desire to integrate or merge two schemata or ontologies is very common today (for example, it may be necessary to integrate two models describing car manufacturing from two different car manufacturing companies who just merged and need a common car manufacturing model). This task is usually carried out manually and can be very time-consuming. By (semi-)automating the process, the speed and accuracy of the integration process can be increased.

We have developed a method that partially automates the process of schema integration, by using various matching algorithms. Each algorithm proposes a set of cross-schema links, sorted by expected reliability, which can then be validated by a human. The algorithms employ concept names, super- and subclass taxonomizations, attributes, and natural language text definitions. The methods are implemented as a set of Perl Programs.

[1] This work was partly funded by STINT - The Swedish Foundation for International Cooperation in Research and Higher Education and Volvo Research Foundation, Volvo Educational Foundation and Dr Pehr G Gyllenhammar Research Foundation.

1. Introduction

Modern society uses formal models, conceptual models, requirements engineering schemata, and ontologies for various purposes. One problem common to all these domains is cross-ontology integration; for example, the need to integrate a STEP AP schema describing the production of a car with a STEP AP schema describing the electronic circuits (of a car) to obtain a single common STEP AP schema. In this scenario there is great risk that the same concept, defined in both schemata, will appear twice in the merged schema. How can we find the redundant concepts? Since concepts can have the same meaning but different names (synonyms), or the same name but different meanings (homographs/homonyms), this is not a trivial problem. For example, in two STEP Application Protocols (AP) studied in this paper, AP212 and AP214, the two concepts *document* and*archived_document* are synonyms, while the two distinct concepts *document_type* and *document_type* are homographs/homonyms, being defined differently in the two ontologies. This problem is typical of model integration in all types of modeling.

2. STEP and Ontologies

STEP stands for **ST**andard for the **E**xchange of **P**roduct model data, and is an ISO 10303 standard [Al-Timimi and MacKrell 96]. STEP has been developed by industry for the exchange of product model data between different platforms, such as CAD/CAM platforms.

In the STEP standard, Application Protocols (AP's) are standardized schemata within each domain, expressing the standardized concepts. AP's exists in several domains, including automotive manufacturing, ship building, electrotechnical plants, and process industry. The Application Protocols are expressed in the data modeling language EXPRESS [Schenk and Wilson 94]. EXPRESS is a static modeling language of entity-relationship type.

Although one can view each AP as a domain ontology, AP's are not very hierarchical. Ontologies can be described as hierarchical conceptual models whose content ranges from very general concepts to very domain specific concepts. In the Artificial Intelligence community a large research effort has been devoted in ontology research, both in the construction of general purpose and domain models and in their re-use in other applications.

One of the first investigations of ontology reuse is found in [Biggerstaff and Richter 87], in which different possibilities and approaches of reusing different components in software development are discussed. In [Maiden and Sutcliffe 92] we find a description of domain abstraction and the mapping to a new domain. [Neches et al. 91] discuss the

sharing, reuse and extension of knowledge bases, pointing out four bottlenecks in sharing and reuse, and proposing a solution to overcome them. [Gruber 93] investigates the sharing and reuse of ontologies over domains and representation languages. [Wielinga and Schreiber 94] discuss the separation of knowledge into various layers in an ontology to permit reuse of the ontologies. [Dalianis 97] describes the modeling of a distribution electrical network using the KACTUS library of ontologies in the technical domain to support the modeling.

A different class of use of ontologies is for lexicon acquisition and building of natural language processing systems. The SENSUS ontology [Knight and Luk 94] at the USC/Information Sciences Institute is built from a number of ontologies, among them the Penman Upper Model [Bateman et al. 89] and WordNet [Miller 95]. SENSUS contains over 70.000 concepts. The SENSUS ontology has been used for a number of natural language applications, including machine translation [Knight and Graehl 97] and text summarization [Hovy and Lin 97]. [Okumura and Hovy 94] link a bilingual dictionary to an ontology for use within machine translation. [Rigau and Agirre 95] discuss a similar approach using the semantic proximity of lexical objects to decide where in the ontology they should be aligned.

Within the requirements engineering community, similar problems of schema integration arise. [Johannesson 92] uses a set of integration assertions to find the proximity of two different concept in a conceptual model. This method is not, however, automated; the integration assertions are proposed by the user.

Until now nobody has used automatic methods to discriminate between concepts in STEP schemata that are semantically close. We demonstrate this in the next section.

3. The Integration of STEP AP's

We chose the following two STEP AP's to be integrated: AP212 and AP214. AP212 is the electrotechnical application protocol, containing 352 concepts, such as *Organization, Person, Point, Curve*. AP214 is the automotive design application protocol, containing 501 concepts. Within each AP, some concepts are related through super- or subclass links. With almost each concept a natural language text description is associated as its definition.

The objective of the integration of the two AP's was to make the resulting merged schema non-redundant and consistent. The problem was to ensure that no concept overlaps occurred.

Since AP212 and AP214 have 352 and 501 concepts respectively, one needs in principle to make 352 x 501 = 176,352 comparisons to find all the right matches. If the AP's have a complete internal superclass taxonomization structure, the number of comparisons can be trimmed by limiting the search to appropriate subtrees. However, in

the case of AP212 and AP214, a total of only 49 superclass and subclass links are defined (all for the general area of points and curves).

Our approach was to identify and extract the common concepts using a combination of several methods. We implemented the following matching algorithms:

- Concept name match
- Definition match
- First sentence definition match
- Superclass match
- Subclass match
- Attribute match

and then implemented various combination functions that combined and weighted their linking suggestions.

To preprocess the STEP files we used the Perl programming language [Wall et al. 96], which is excellent for string processing. Macintosh and UNIX Workstations were used for executing the Perl programs, MacPerl 5 and Perl 5 respectively. We compiled a set of Perl programs that can be used as a standalone integration module.

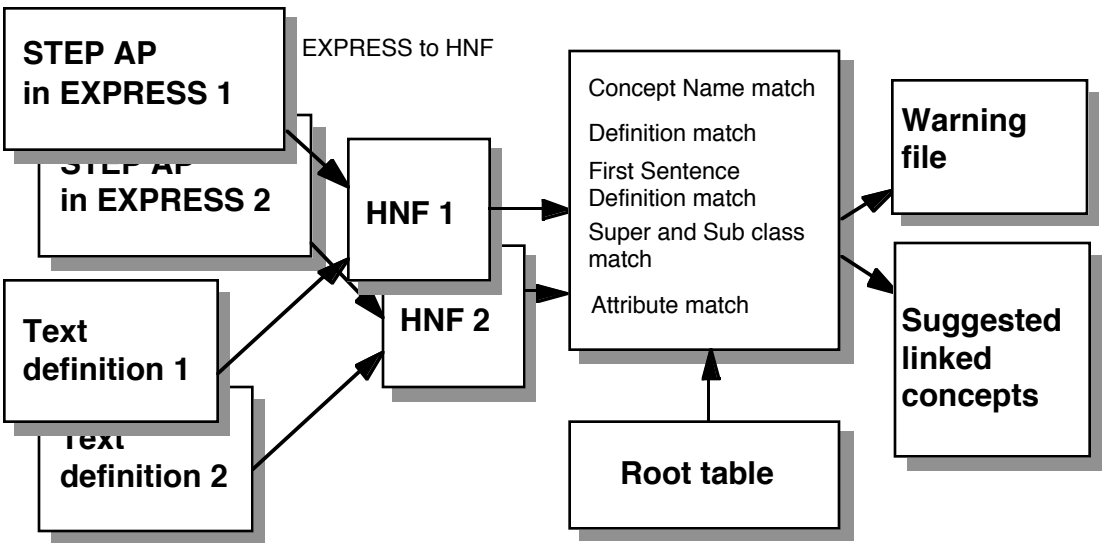

Figure 1. Overview of the schema merging process.

Our schema merging process takes as input two or more STEP AP EXPRESS files and produces as output a file that gives guidelines of which concepts in the STEP AP EXPRESS files are redundant or overlapping. This process consists of the following steps:

The **first step** is to translate the EXPRESS files to a normalized form HNF (Hercules-Hovy Normal Form) and to attach the corresponding text description to each concept. An example of the HNF format can be seen in Section 3.1 below.

A warning file is also created informing about which concepts do not have text definitions.

The **second step** is to compare the concepts in the two STEP AP HNF schemata: each concept and its associated information (super- and subclasses, natural language text descriptions, attributes) in one HNF file is matched against each concept in the other HNF file, and the most similar concepts (highest matching scores) are retained.

The **third step** is to combine the individual scores, sort the concept match suggestions, and present them to the user for human validation. (To prevent the result files from becoming too large, we used a cut-off value and recorded only the match suggestions with a weighted matching score of over 0.17). Concepts whose matches the user has accepted can then be merged, and concepts with rejected matches can be entered into the merged ontology separately.

3.1 HNF

In order to apply our matching procedure to other kinds of ontologies in the future, we defined a simple neutral format upon which the matching algorithms operate.

AP212
Entity: document
Attribute: (id:identifier)
Attribute: (name:label)
Attribute: (description:text)
Attribute: (kind:document_type)
Attribute: (ur1:id)
Definition: A Document is a reference to digital data or non-digital data that are not within the scope of ISO 10303. Each Document may be one of the following: a Change_description (see clause 4.2.24), a Drawing (see clause 4.2.77), a General_description (see clause 4.2.102), or a Specification_document (see clause 4.2.226). The data associated with a Document are the following: ó alternate_designation; ó custodian; ó described_object; ó designation; ó document_title; ó found_in; ó language; ó type_of_document. Each Document has described_object defined by zero, one, or more Assembly_relationship objects. Each Assembly_relationship has described_object defined by zero, one, or more Document objects....

AP214
Entity: archived_document
Superclass: (external_document)
Attribute: (archiving_format:undefined_object)
Attribute: (found_in:undefined_object)
Definition: An Archived_document is a piece of product data that is archived in a non-digital form. An Archived_document is a type of External_document (see 4.2.162). The data associated with an Archived_document are the following: -- archiving_format; -- found_in.
Example: EXAMPLE 26 -- Paper plots of technical drawings, microfiche, or paper documents such as calculations or test reports are examples for an Archived_document.

Figure 2. Two synonym concepts in HNF form

We created a concept rewriting routine to convert the formal AP concept definitions into HNF, to attach the concepts' text descriptions and attributes (which are stored in separate files, by STEP tradition) to the concept, and to save all the information of interest associated with each concept. An example of HNF is shown above in Figure 2:

3.2 Matching Algorithms

In this subsection we define the six matches.

Concept name match (CNM)

The concept name match is based on the assumption that two AP concepts denoting the same entity in the world are likely to have the same or similar names. (We assume that STEP AP creators work in English.) Ideally, the match is an exact one-to-one letter correspondence, which would give a value of 1.00; or less ideally a substring match that measures the length of overlap between the two name strings. (In this work, we ignore case differences: *Document* and *document* are deemed equal.) We define the score as the quotient of the substring length and the mean value of the lengths of the two matched strings. For concepts *i* and *j*, we use the formula:

$$\mathrm{CNM}(i,j) = \mathrm{length}(\mathrm{name}_i \cap \mathrm{name}_j) \,/\, ((\mathrm{length}(\mathrm{name}_i)+\mathrm{length}(\mathrm{name}_j))/2)$$

The resulting suggested link *i–j* is saved in a file together with the score. A slightly more sophisticated name match is used in [Hovy 98], who rewards substring matches if they both end at common intraname delimiters such as "-" and "_". Still more sophisticated matches that ignore delimiters, or allow them to be replaced with capital letters, can be defined.

Definition match (DM)

The definition match is based on the assumption that two concepts referring to the same entity in the world will have similar natural language definitions. Ideally, the two definitions will be word-for-word identical, giving a match score of 1.00. In practise, terms may be pluralized, variant form of words may be employed, etc. Therefore the words in each definition are demorphed (i.e., converted to root form) before matching, and duplicates and stop words ("a", "the", etc.) are removed before counting the number of shared words. To demorph, we used a root table [Lin 98] extracted from WordNet [Miller 95]. The normalized demorphed definition of each concept in one HNF file is matched against the normalized demorphed definitions of all concepts in the other HNF file. For concepts *i* and *j*, we define the measure as:

$$\mathrm{DM}(i,j) = \text{number-of-words}(\mathrm{def}_i \cap \mathrm{def}_j) \,/\, ((\mathrm{length}(\mathrm{def}_i) + \mathrm{length}(\mathrm{def}_j))/2)$$

If the score is above a threshold value (0.2) , the resulting suggested link *i–j* is saved in a file together with the score. A similar heuristic is used in [Knight and Luk 94; Hovy 98].

First sentence definition match (FSDM)

The first sentence definition match is identical to the definition match. However, we use only the first sentence of a definition, since investigation of the typical content of text definitions showed that the most important portions of the definitions appeared in the first sentences. Subsequent sentences in one AP tended to contain references to ISO standards, while subsequent sentences in the other contained examples and sometimes confusing additions.

Superclass match and Subclass match (SUPC and SUBC)

The superclass match checks whether the name of the superclass of the target concept *i* is exactly the same as the name of the superclass of a candidate matching concept *j* in the other STEP AP. If a match is found the suggested link *i–j* is saved in a file.

The subclass match checks whether concept *i* has some subconcept with exactly the same name as some subconcept of concept *j* in the other STEP AP. For $sub_{i,j}$ the number of identically named subconcepts shared by *i* and *j*, and $mean_{i,j}$ the average number of subconcepts for *i* and *j*, we use the formula:

$$\mathrm{SUBC}(i,j) = (1 - 1/(1+mean_{i,j})) * (sub_{i,j} / mean_{i,j})$$

The second half of this formula reflects the proportion of the possible shared attributes that are in fact shared, and the first half is a factor that rewards matches with more shared attributes. The resulting link *i–j* is saved in a file together with the score.

Attribute Match (ATT)

The attribute match checks whether some attribute(s) of the target concept *i* are the same as some attribute(s) of a candidate matching concept *j*. For $attribs_{i,j}$ the number of matched attribute names between *i* and *j* and $mean_{i,j}$ the average number of attributes of *i* and *j,* we use the formula:

$$\mathrm{ATT}(i,j) = (1 - 1/(1+mean_{i,j})) * (attribs_{i,j} / mean_{i,j})$$

If this match occurs above threshold value, the resulting link *i–j* is saved in a file together with the score. A similar match is used in [Okumura and Hovy 94, Rigau and Agirre 95].

3.3 Merging Algorithm

Each match is implemented as a separate Perl program, and the resulting concept link suggestions and their scores are written to separate files. At the end, another Perl program

reads all the different suggestion files, integrates their scores, and sorts them by decreasing score, to calculate which of the concepts are semantically closest to each other.

Since the best integration formula was not immediately apparent, and since each score individually was normalized between 0 and 1, we adopted a simple linear combination function:

$$a_1*\mathrm{CNM}(i,j) + a_2*\mathrm{DM}(i,j) + a_3*\mathrm{FSDM}(i,j) + a_4*\mathrm{SUPC}(i,j) + a_5*\mathrm{SUBC}(i,j) + a_6*\mathrm{ATT}(i,j)$$

We varied the strengths of coefficients a_1,... in different directions, but found little practical difference in the final ranking of the scores. Eventually we simply set all coefficients equally to unity (e.g., $a_1 = a_2 = a_3 = a_4 = a_5 = a_6 = 1$), and renormalized the result by dividing by 6:

$$\mathrm{integration\_value}(i,j) = (a_1*\mathrm{CNM}(i,j) + a_2*\mathrm{DM}(i,j) + a_3*\mathrm{FSDM}(i,j) + a_4*\mathrm{SUPC}(i,j) + a_5*\mathrm{SUBC}(i,j) + a_6*\mathrm{ATT}(i,j)) / 6$$

This function ranges between 0 and 1, with 1 indicating a perfect match. This solution is not optimal, and represents a point on which further research is required.

The results of this calculation are sorted in decreasing order and written to a result file. Finally, another Perl program filters out all duplicates and already matched concepts, leaving a list of suggested concept matches, sorted by expected goodness, for human validation.

3.4 The Findings: Merging AP212 and AP214

Overall, the algorithm produced 74 match suggestions with combination scores above our somewhat arbitrary threshold of 0.17.

We measured Precision (the correctness of the matching suggestion) in ranges of ten at a time. We classified each suggestion as either Correct, Incorrect, Close (if the match was not strictly correct, but if we could find no concept that was more correct), and Unclear (as concept definitions are often not specific enough).

Of the top-scoring (10 suggestions, 9 were correct and 1 was close, for a precision score of 90% (being strict, without partial credit for close matches). Of the second-best 10 suggestions, 8 were close, 1 was incorrect, and 1 was unclear, giving a strict Precision of 80% and a lenient Precision of 90% (counting close matches). Of the fifth-best group of 10 suggestions, 4 close, 5 incorrect and 1 was unclear giving, a strict Precision of 0% and a lenient Precision of 40%. Overall Precision for the top-ranking 74 match suggestions was 34% (strict) and 61% (lenient). See Table 1, below.

Inspection of the match suggestions indicated the importance of concept names: the first suggestion of differently-named synonym (non-homonym) concepts occurred at rank 27

with integration score of 0.2634. In the top 74 suggestions, 25 were homonyms, of which 20 concepts were correct and 2 close. Thus a poor person's matching algorithm may employ just Name Match with fairly reasonable results.

Match rank	Correct	Close	In-correct	Un-clear	Strict Precision	Lenient Precision	Score
1–10	9	1	0	0	0.90	1.00	>0.3438
11–20	8	0	1	1	0.80	0.80	>0.3156
21–30	4	4	1	1	0.40	0.80	>0.2427
31–40	3	4	2	1	0.30	0.70	>0.2142
41–50	0	4	5	1	0.00	0.40	>0.1984
51–60	1	4	5	0	0.10	0.50	>0.1881
61-70	0	1	7	2	0.00	0.10	>0.1721
71–74	0	2	2	0	0.00	0.50	>=0.1702
Σ=74	25	20	23	6	0.34	0.61	

Table 1. Ranking matches between AP212 and AP214

One may therefore argue that these results are skewed by the high likelihood of correct matches for homonyms. If we remove all homonyms, we will just have the total number of non-homonym matches above threshold that is 49, of which 5 are correct and 18 close, giving for synonyms a strict Precision value of 10% and a lenient Precision of 47%. (If a non-homonym is correct then it is a synonym).

Type	N	Correct	Close	Strict Precision	Lenient Precision
Homonyms	25	20	2	0.80	0.88
Non-homonyms	49	5	18	0.10	0.47
Σ=74	74	25	20	0.34	0.61

Table 2. A comparison between homonyms and synonyms.
(The table do not show incorrect or unclear concepts)

Recall is much more difficult to measure. Without manually reading all 352 x 501 = 176,352 possible pairs, we have no idea how many correct matches the algorithms missed. Any shortcuts we could think of to find likely matches have already been implemented in the match algorithms!

Overall, we found the results rather encouraging: of the large number of possible matches, the algorithms identified just 74 that had to be validated, of which about 1/3 were correct and about 1/4 close.

3.5 Control: Merging Ontology AP210 and AP212

As a control we carried out the integration experiment with one new ontology, AP210. AP210 is the electronic assembly, interconnection and packaging design Application Protocol, and has 568 concepts. We merged AP210 with AP212, the electrotechnical Application Protocol. AP212 has 352 concepts, giving 199,936 possible combinations. We found 14 suggested matches with integration values above 0.2092.

Match rank	Correct	Close	In-correct	Un-clear	Strict Precision	Lenient Precision	Score
1–10	8	0	1	1	0.80	0.80	>0.2265
11-14	2	1	0	1	0.50	0.75	>=0.2092
Σ=14	10	1	1	2	0.71	0.78	

Table 3. Ranking matches between AP210 and AP212

The first suggestion of differently-named synonym (non-homonym) concepts occurred at rank 1 with integration score of 0.3536. We had altogether 10 homonyms of which 8 were correct and 1 close, and 4 synonyms of which 2 were correct

3.6 Conflicts and Restrictions

When integrating different matching algorithms' results, conflicts can occur when they make contrary suggestions. This can occur in the following situations:

- Not enough information, or no definitions at all, are attached to a concept. This can prevent two closely related concepts from being merged. A warning mechanism were constructed to inform the user.
- Two concepts that are synonyms (e.g., *group_relationship* and *assembly_component_relationship)* are matched with each other, but there exist also homonyms (e.g., *group_relationship* and *group_relationship*) with perfect name match that do not score well by their definitions.
- A whole group (an aggregate) of possible merges appears. For example, *dimension* does not have corresponding concept *dimension* in the other schema, but just a set of subconcepts (e.g, *curve_dimension, angular_dimension, location_dimension* etc.), therefore the concept *dimension* can replace all these subconcepts. A mechanism covering this type of cases would be desirable.

4. Conclusions and Future Research

Our research and implementation demonstrates a semi-automated method that is efficient and accurate in finding related concepts across STEP AP schemata, using a set of very simple heuristics. It is perhaps surprising that such simple methods work as well as they do, and it remains to be seen how well the methods presented in this paper will work for ontologies that differ more widely. One experiment we intend to try is the merging of an AP schema with a natural language (NL)-oriented ontology WordNet or SENSUS. Similar merging techniques for merging NL and Artificial Intelligence Ontologies such as SENSUS [Knight and Luk 94], WordNet [Miller 95], and MIKROKOSMOS [Mahesh 96] report similar levels of accuracy [Rigau and Agirre 95; Hovy 98].

Even if the ontologies or schemata themselves are fairly similar, a question remains about the similarity of the individual features of concepts. For example, the definition match will perform better when concepts' definitions in the two ontologies are written by the same person, or under the same general guidelines. When not, suitable adaptations must be made; such as our creation of the First Sentence Definition Match to overcome the general disparity between definitions in AP212 and AP214.

One future research direction would be to cluster concepts which are semantically very close, and have a technique to merge them into one concept (c.f. *dimension* with *curve_dimension, angular_dimension, location_dimension)*. At the moment, we just allow monogamy, *dimension* merged with *curve_dimension*.

In order to facilitate the human's concept validation stage, we have created a program to convert STEP AP schema definitions into inputs for the natural language generator ASTROGEN [Dalianis 97]. Using this program, the user can ask to see an English description of any concept (in particular, those that are suggested to be merged). We have also grouped the merging algorithms and the generator into a single tool, complete with human validation interface, to support STEP AP integration (and similar work); see [Dalianis 98].

In general ontologies, that are used for real domain-level reasoning/inference, contain many formalized axioms and concept interrelationships, information that is not present in STEP AP's. With such formal nature, we are optimistic that additional, possibly more powerful, matching techniques will be found. Thus we do not believe that the potential of this work is exhausted, and hope that it will be extended to be useful also in other domains.

Acknowledgments

Great thanks to Chin-Yew Lin and Uli Germann at USC/Information Sciences Institute for their help in the art of programming Perl.

5. References

Al-Timimi, K. and J. MacKrell. 1986. STEP—Towards Open Systems. *STEP Fundamentals & Business Benefits*, CIMdata.

Bateman, J., R.T. Kasper, J.D. Moore, and R.A. Whitney. 1987. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. Unpublished research report, USC/Information Sciences Institute, Marina del Rey.

Biggerstaff, T. and C. Richter. 1987. Reusability Framework, Assessment and Directions. *IEEE Software* 4(2), pp. 41–49.

Dalianis, H. 1997. ASTROGEN - Aggregated deep and Surface naTuRal language GENerator, http://www.dsv.su.se/~hercules/ASTROGEN/ASTROGEN.html.

Dalianis, H. and F. Persson. 1997. Reuse of an ontology in an electrical distribution network domain. Proceedings of the AAAI 1997 *Spring Symposium Series, Ontological Engineering*, pp 25–32, Stanford University, California.

Dalianis, H. 1998. The VINST Approach:Validating and Integrating STEP AP Schemata Using a Semi Automatic Tool, Submitted to the The European Conference on Integration in Manufacturing, IiM-98, to be held in Gothenburg, October 6-8, 1998.

Hovy, E.H. and C-Y. Lin. 1997. Automated Text Summarization in SUMMARIST. *Proceedings of the ACL Workshop of Intelligent Scalable Text Summarization*, July 1997.

Hovy, E.H. 1998. Combining and Standardizing Large-Scale, Practical Ontologies for Machine Translation and Other Uses. *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*. Granada, Spain.

Johannesson. P. 1993. A Logical Basis for Schema Integration. In Third International Workshop on Research Issues in Data Engineering—Interoperability in Multidatabase Systems, E. Bertino (ed), IEEE Press, Vienna, pp. 171–181.

Gruber, T.R. 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5(2), pp. 199–220.

Knight, K. and S. Luk. 1994. Building a Large Knowledge Base for Machine Translation. *Proceedings of the American Association of Artificial Intelligence Conference (AAAI-94),* Seattle, Washington.

Knight, K. and J. Graehl. 1997. Machine Transliteration. *Proceedings of the ACL-97*. Madrid, Spain.

Lin, C-Y. 1998. Assembly of Topic Extraction Modules in SUMMARIST. *Proceedings of the AAAI Spring Symposium on Intelligent Text Summarization*.

Mahesh, K. 1996. Ontology Development for Machine Translation: Ideology and Methodology. New Mexico State University CRL research report MCCS-96-292.

Maiden, N.A. and A.G. Sutcliffe. 1992. Exploiting Reusable Specifications through Analogy, *Communications of the ACM* 35(4), pp. 55–64.

Miller, G.A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38 (11), pp. 39–41.

Neches, R., R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W.R. Swartout. 1991. Enabling Technology For Knowledge Sharing. *AI Magazine* 12(3).

Okumura, A. and E.H. Hovy. 1994. Lexicon-to-Ontology Concept Association using a Bilingual Dictionary. *Proceedings of the 1st American Machine Translation Association (AMTA) Conference*.

Rigau, G. and E. Agirre. 1995. Disambigating bilngual nominal entries against WordNet. *Workshop on the Computational Lexicon*. 7th European Summer School in Logic, Language and Information (ESSLLI'95). Barcelona, Spain.

Schenk, D. and P. Wilson. 1994. *Information Modeling the Express Way*. Oxford University Press.

Wall, L., T. Christensen, and R.L. Schwartz. 1996. *Programming Perl*. O'Reilly & Associates Inc.

Wielinga, B.J. and A.Th. Schreiber. 1994. Conceptual Modelling of Large Reusable Knowledge Bases. In K. Von Luck and H. Marburger (eds), *Management and Processing of Complex Data Structures*, Springer Verlag Lecture Notes in Computer Science no 777, pp 181–200.