A language and Algorithm for Automatic Merging of Ontologies

Abstract

Nowadays, most of the important information resources that the people require are available through the Internet. The use of several sources in the Internet requires merging the information into a knowledge base in a reasonable way. We will use an ontology, an information technology that manages this knowledge in computers.

Merging is an important task and many languages and tools have been developed to describe and process Internet content but the current languages (DAML+OIL, RDF, OWL, etc.) lack a complete expressivenes. For this reason, we present two important improvements to facilitate knowledge interchange: 1) The OM (Ontology Merging) Notation that provides substantial improvements to these languages and 2) The OM Algorithm, that is totally automatic in comparison with others (Prompt, Chimaera, OntoMerge, FCA-Merge, IF-Map and ISI) where the user manually solves the most important problems found in the merging.

1. Introduction

These days computers are not anymore isolated devices but they are important entry points in the world-wide network that interchanges knowledge and carry out business transactions. Nowadays, using Internet to get data, information and knowledge interchange is a business and an academic need. Despite the facilities to access Internet, people face the problem of heterogeneous sources, because there are no suitable standards in knowledge representation. This paper addresses this need of businesses and academia.

Many answers that people require involve accessing several sources in the Internet, which are later manually merged in a "reasonable" way. Merging the information is an important task. Many languages and tools (DAML+OIL [5], RDF [8] and OWL [12]) have been developed to describe and process Internet content but, unfortunately, they lack enough expressiveness to detail knowledge representation.

It is required that the computer deciphers the information (said, in a document written in a natural language) and converts it to a suitable notation (its knowledge base) that preserves relevant knowledge. This knowledge base can be an ontology. An ontology manages the knowledge through nodes that are joined through *relations*, to describe a knowledge domain. Current works that merge ontologies (Prompt [13], Chimaera [11], OntoMerge [6], FCA-Merge [9] and If-Map [14]) rely on the user to solve the most important problems found in the process: inconsistencies and adequate knowledge extraction. This paper describes two important contributions to obtain better advantage of the Web resources:

1) A new notation to represent knowledge using ontologies, called OM (Ontology Merging) Notation, and
2) An automatic algorithm to merge ontologies, called OM Algorithm.

The OM notation provides several improvements to current languages that define ontologies. Two of them are: (a) a new type of relation called *Partition*; (b) a node or concept can also be defined as a relation.

Likewise, the merging algorithm that we will explain is totally automatic. This algorithm solves by itself all the problems found in the process. That is to say, the user does not take part in the process.

2. OM Notation

In the context of sharing knowledge, ontologies provide a clear, syntactic and formalized structuring of a set of nodes also called concepts that are related to each other, under a knowledge domain and that is common to many people and machines.

OM Notation represents ontologies through a structural design with labels similar to XML, these labels identify the description of the concepts and their relations. The labels and their descriptions are shown in table 1.

Binary and n-ary relations are described in OM Notation. That is, a relation can have several values and these can be concepts. For example, the concept *Zebra* has a relation *Color* that is connected to two elements *White* and *Black*.

Table 1 Labels used in the OM Notation**.**

<concept> *c* </concept>	Where *c* represents the name of the concept.
<language> *l* </language>	Where *l* represents the language in which the words are defined.
<word>w_1,w_2…w_n</word>	Where w_1,w_2…w_n represent the words that describe the concept *c*.
<arity> *a* </arity>	Where *a* is a positive number that describes the arity of the concept *c*.
<relation> *n* = *v* </relation>	Where *n* represents the name and *v* represents the value of the relation. The value *n* and *v* are concepts. *v* can be a list if the relation has several values.
<part> *c* </part>	The concept that contains this relation is part of the concept *c*.
<member> *c* </member>	The concept that contains this relation is member of the concept *c*.
<subset> *c* </subset>	The concept that contains this relation is a subset of the concept *c*.
<type> *c* </type>	The concept that contains this relation is a type of the concept *c*.

The relations are properties or characteristics of the node or concept where they are defined. An example of this is the relation *Eat*, shown in figure 1. Other relations exist, such as the hyponymous relation, that are expressed through nested concepts. For example, *plant* is a subset of *physical_object*.

Relations are implicit or explicit. The implicit relation indicates a structural relation (parent-son). For example, the relation "part of" exists between holonymous and meronymous sets.

A set is holonymous of another when its semantic notion represents the whole of an object; therefore *bicycle* is holonymous of *handle-bar*. A set is Meronymous when it represents a part of an object; therefore *handle-bar* is meronymous of *bicycle*.

Other implicit relations exist, such as: Hyperonymous and Hyponymous, where a term is hyperonymous of another if the meaning of the first includes the second concept.

```
<concept>thing
   <language> English
      <word>thing, something, object, entity </word>
   </Language>
   <concept>physical_object
      <language> English
         <word> concrete_object, physical_object</word>
      </Language>
      <concept>plant
         <language> English <word>plant, tree</word> </Language>
         <concept>fruit
                  <idiom>Ingles <word>fruit, citric</word> </Language>
         </concept>
      </concept>
      <concept>human being,
         <language> English
         <word> person, people, human being</word></Language>
         <relation>eats=tropical_fruit, citrus</relation>
         <relation>Partition=age {0<age<=1 : baby; 1<age<=10 : child;
          10<age<=17 :puberty; 17<age<=29 : young; 29<age<=59 :
          mature; age>59 : old;}</relation>
      </concept>
   </concept>
</concept>
```

Figure 1. Representation of an ontology in OM Notation.

An ontology with nodes and relations is shown in the figure 2. Circles and arrows are nodes and relations respectively.

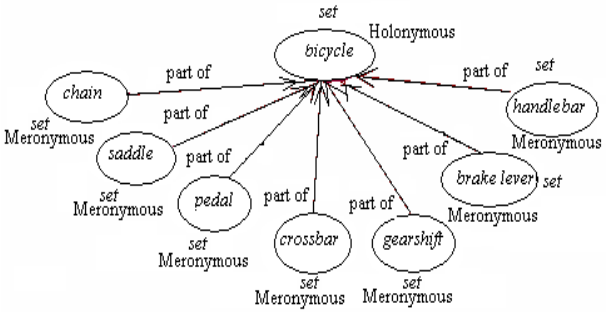

Figure 2 Graphical representation of an ontology With the relation "part of."

Other implicit relation is "type of." This is the same that "subset". It is not shown in this paper.

The explicit relations provide additional semantics to the nodes, describing properties, characteristics or actions that distinguish a concept from others. For example, the relation *activity* between *Port of Salina Cruz* and *Commercial activity*, *Turistic activity* and *Fishing activity* are shown in figure 3. Other examples are presented here:

1. *Apple* Color *Yellow*
2. *Apple* Form *Round*
3. *Cat* Drinks *Milk*

2.1. Relation of type Partition

A Partition of a set S is a collection of subsets of S such that whatever two elements of this collection are mutually exclusive and all of them are collectively exhaustive.

Partitions are not represented in languages [5], [8] and [12]. Nevertheless, we represent partitions in the following way:

Partition=*nomPart*{$range_1$:$value_1$; … $range_n$:$value_n$}
Where *nomPart* represents the name of partition, *range* is the characteristic that distinguishes this set of other sets in the partition. The *range* can be an interval, a list of elements or simply a character. The *value* represents the value of the range, the name of interval, a list or a character. This value can be a node or concept. For example:
<relation>
Partition=age {0<age<=1:baby;1<age<=10:child;
10<age<=13:teenager;13<age<18:young;18<=age<40:
adult; 40<=age<60: mature; 60<=age: old}
</relation>
The graphic representation of a partition is shown in figure 3; the small, black circle represents the partition.

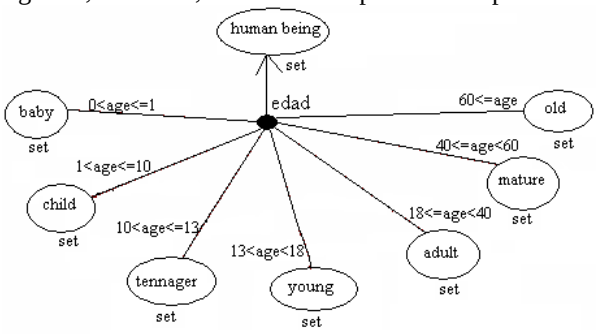

Figure 3 Graphical representation of a partition

Partitions are a form of classifying a concept, to be able to infer on this later. The inference from partitions is not a subject of paper.

2.2. A concept can be a relation

Binary relations are represented as follows:
$r(C_{name}, C_{value})$
Where, *r* represents the name of relation, C_{name} represents the name of the concept of the relation, C_{value} represents the concept value of the relation. An example is:
Mother (*Mary Ball Washington*, *George Washington*)
Mary Ball Washington is mother of *George Washington*, but *Mother* can be a concept that contains more information of the meaning of *Mother* and other concepts related to this. Other contributions exist but will not be explained in this brief space.

3. OM Algorithm for automatic merging of ontologies

Current works that merge ontologies ([6], [9], [11], [13] and [14]) need the intervention the user for this important process. Our OM algorithm is the unique (until now) because it merges ontologies in an automatic form. OM executes the following general steps:
Given three ontologies *A*, *B* and *C*, and concepts *a*, *b*, *c* that belongs to *A*, *B* and *C* respectively:

1. For each $a \in A$, OM obtains $b = sim(a,B)$, the concept in *B* most similar to *a*, as well as *sv*, the similarity value between *a* and *b* [1].

1.a. If $sv > 0$, we will use the *b* obtained in (1) in order to merge *a* and *b* obtaining $c = ext(a,b)$, where *ext* is explained below;

1.b. If $sv = 0$, this means *a* has no similar concept in *B*, therefore concept *a* is added to the new ontology *C*.

2. C, the resulting or merged ontology, is computed as: $C = \{c \mid c$ is obtained in (1.a)$\} \cup \{a: sim(a,B) = 0\} \cup \{b:sim(b,A) = 0\}$. C consists of all the *c*'s obtained in (1.a), plus all "unique" *a*'s that have no similar concepts in *B*, plus all "unique" *b*'s that have no similar concepts in *A*.

The function *sim*(*concept*, *ontology*) of the algorithm COM [1] is a similarity search function that takes the *concept* and looks for its more similar concept in the *ontology*, giving back the most similar *concept* and a *sv* (similar value) with value between 0 and 1.
The function *ext*(*a*, *b*) of the OM Algorithm [3] extends *a* by adding to it, the relations of *b* that *a* lacks, and enriches those relations in *a* that are synonymous with their equivalent relations in *b*. In this step, inconsistencies are detected between names and values of a relation. An inconsistency is a fact of the ontology *B* that contradicts a fact of the ontology *A*. More on [7].
In the process of merging ontologies the following cases appear.

3.1. Verification of the arity in a relation

Remember that a relation in OM Notation can also be a concept. The arity of a relation is the number of values that the relation can take. If the relation takes only a value it is said that it is mono-valuated arity. For example, the arity of relations *Mother* and *Father* is mono-valuated; because a person can have only one *Mother* and only one *Father*.
A relation is a multi-valuated arity if it can take several values. For example, the *political position* that a person can have. If $r_A(C_{name}, C_{value}) = r_B(C_{name}, C'_{value})$, where the index *A* in r_A means "from the ontology *A*", the OM Algorithm verifies the arity of relation r_A before copying the value C'_{value} to the resulting ontology. If this relation is a multi-valuated arity, the resulting

ontology receives the new value; else, it may be that $C_{value} = C'_{value}$, no copy is performed; else (if $C_{value} \neq C'_{value}$), OM tries to solve the problem [a unary relation having two distinct values] using the Confusion Theory [2].

3.2. Union (addition) of a new relation

The addition of a new relation in *A* to the resulting ontology occurs when the name and value of the relation in *A* are different from the names and values of *all* the relations in *B*, that is to say; they are totally different concepts and they aren't synonymous. In this case, the names and values of these relations are added to the resulting ontology.

3.3. Union of a relation with elements that are synonymous

In order to know if a concept $a \in A$ has a most similar (synonymy) concept $b \in$ in *B*, OM applies COM [1] Algorithm. COM returns *b*, the most similar concept, a similarity value *sv*. If $sv \in [0.8, 1]$, *b* is considered synonym of *a*. In this case, *ext* enriches *a* with suitable relations and words from *b*, as explained above.

Example. Given the relation in *A*: *Surface* (*Oaxaca*, *Surface of Oaxaca*) and the relation in *B*: *Territorial extension* (*Oaxaca*, *Territorial extension of Oaxaca*), we want to merge them. To do this, *sim* (*Surface*, *B*) is applied. This function gives back *sv* = 1 with the concept *Territorial extension*. Thus, OM does not fuse both relations but it enriches the relation in *A*, which is *Surface*, with the new words and properties of *Territorial extension* in *B*, copying the enriched relation to the resulting ontology *C*.

3.4. Confusion in the name of relations

During the copy of the relation r_A, it is possible that there is no similar relation in *B*, but that another name r_B exists in *B* with the same value as r_A has in *A*. The confusion arises when both relations r_A y r_B share the same value. The OM Algorithm looks for the synonymy between the names of relations (it could be that r_A and r_B are synonyms, although they have different names); that is, it applies COM to the names of the involved concepts r_A and r_B. This step is applied when the relations are concepts. If COM returns *sv* between 0.8 and 1, then they are synonyms, otherwise they are not. Other forms to find the synonymy between the relations, are not explained here due to brevity. If they are not synonymous, OM solves the problem using Confusion [2]. For example:

Given a relation $r_A \in A$ = *Hydrology* (*Oaxaca*, *Main river of Oaxaca*), and $r_B \in B$ = *River* (*Oaxaca*, *Main river of Oaxaca*). We see that r_A and r_B have the same values, but different names, and they are not synonyms. To solve this, a hierarchy of concepts is used where the names of the relations are represented. Figure 4 shows this hierarchy. In the hierarchy the number of levels is obtained. It is to say, level of the depth is 2. The value of the Confusion [2] conf(*r*, *s*) is obtained by counting the *descending* links in the path from *r* to *s*, and dividing by *h*, the height of the hierarchy (deepest number of levels). Thus, conf(*River*, *Hydrology)* = 0/2. Whereas conf (*Hydrology*, *River)* = 1/2 = 0.5. OM selects the lowest value, 0 in this case, so OM chooses 0 and uses *River*, the most specific concept according to the hierarchy. Consequently, *C* is enriched with *River* (*Oaxaca*, *Main river of Oaxaca*).

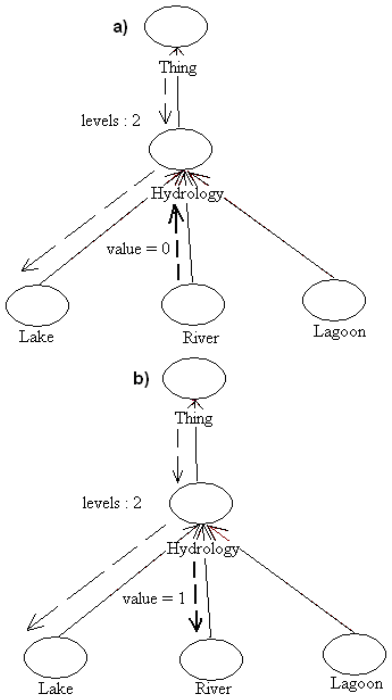

Figure 4 The Confusion of using *River* instead of *Hydrology* is 0. This is shown in a), and the Confusion of using *Hydrology* instead of *River* is 0.5. This is shown in b)

3.5. Confusion in the value of the relations

Given two equal relations r_A and r_B, but their values v_A and v_B (in *A* and *B*, respectively) are different, the arity of the relation is verified. If it is mono-valuated, the Confusion algorithm [2] is applied to v_A and v_B, using the same procedure in 3.4, but over the values.

Example: given the relation r_A = *Birthplace* (*Benito Juárez*, *San Pablo Guelatao*), and r_B = *Birthplace* (*Benito Juárez*, *México*), the arity of r_A is checked. It is mono-valued (impossible to be born in two places, unless they are synonyms or one is a "subset" or "part of" the other). Hence, OM looks for the synonymy of *San Pablo Guelatao* and *Mexico* (could it be that San Pablo Guelatao is just a synonym for Mexico?). Not true. Therefore, OM computes conf(*San Pablo Guelatao*, *Mexico*) = 0/5 and conf(*Mexico*, *San Pablo Guelatao*) = 3/5. The result is min(0, 3/5) = 0, as figure 5 shows.

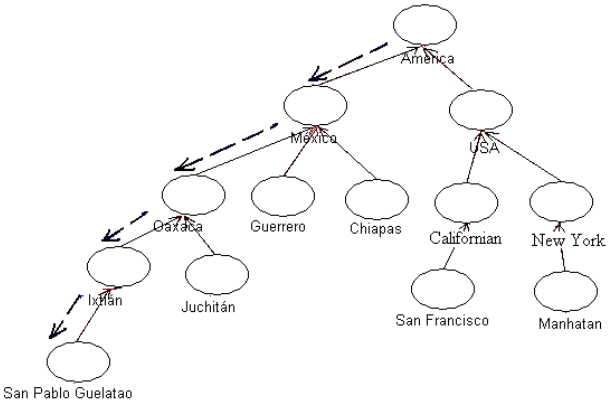

Figure 5. Graphical representation of the hierarchy that indicates the number of levels.

Therefore OM decides to conserve the most specific value, and it adds to *C* the relation r_A = *Birthplace* (*Benito Juárez*, *San Pablo Guelatao*) in *A*.

3.6. Partitions that must not be merged

If the names of the partitions in *A* and *B* are equal or synonymous and the ranges also are equal, but the values of these are different, OM recognizes the values of the ranges as synonymous, enriching the value of the partition in *A* with the values of the partition in *B*. That is to say, copies the partition in *A* into *C*, and complements these values (of *A*) with the words or concepts in *B*.

3.7 Verification of nested relations

During of fusion of ontologies, nested relations are also copied. OM expunges from the resulting ontology, redundant relations. These relations arise when three concepts in *C* exist related in certain way. For example, $c1_C$, $c2_C$ y $c3_C$ with the following relations: $c1_C \subset c2_C$, $c2_C \subset c3_C$ and $c1_C \subset c3_C$; the redundant relation is: $c1_C \subset c3_C$. Therefore, OM eliminates it from ontology *C*. The redundant relations do not only exist in those of type <subset>, also in those of type <part> and <member>. Example: Figure 6 shows two ontologies *A* and *B* that merge to obtain *C*. The arrows represent the similarity between the source concept in *A* (where the arrow starts) towards the target concept in *B* (where the arrow arrives). In the Figure 6 it is possible to observe that the concepts *Turtle criptodira* in *A* and *Turtle pleurodira* in *A* have *Turtle* as their parent, but in *B* *Turtle* is the grandparent of them.

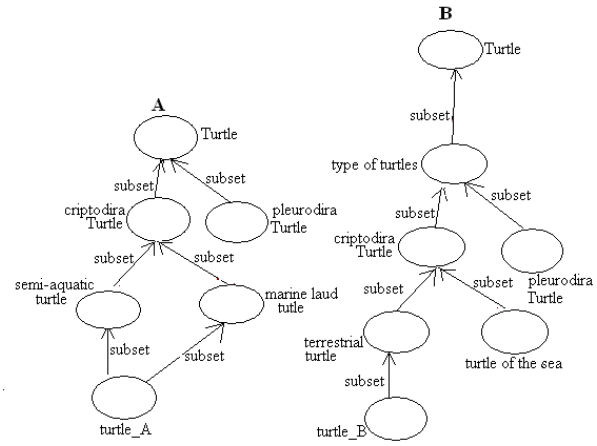

Figure 6. A and B ontologies with the relations in *Turtle* that it will generate nested relation

Figure 7 shows the result of the merge into ontology *C*, where the redundant relation *subset* between the concepts *criptodira turtle* and *pleurodira turtle* has been eliminated.

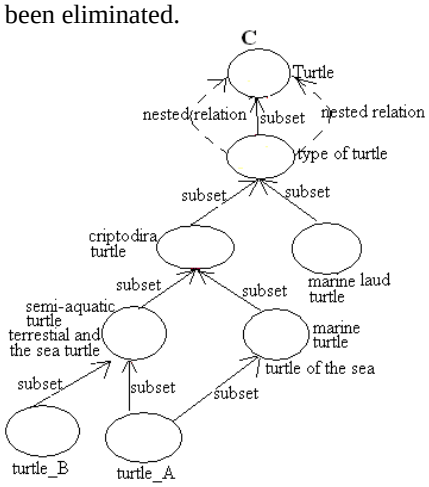

Figure 7 Graphical representation of an ontology with a nested relation.

3.8 Contributions of the OM Algorithm

1. Totally automatic, requires no human intervention.

2. It handles partitions as well as subsets.
3. It handles nodes (concepts) in an ontology that are "shallowly" described by just a word, a word phrase or a set of them.
4. Relation among nodes can also be concepts (that is, they can be nodes).
5. It detects inconsistencies (contradictions) in the knowledge in ontology *A* versus the knowledge in *B*, using inconsistency measurements [7] and confusion [2].
6. It solves some of the detected contradictions in (5), through inconsistency measurement [7].

3. Tests on real cases

OM has merged ontologies in the domain of geographic zones, description of animals, biographies and description of tools and products. The ontologies were obtained manually from several documents collected from the Web, describing the same topic (turtles, say) in different manners. The obtained ontologies were merged (automatically) by OM.
The validation of results has been made manually; we have found that OM produces very acceptable results.
The work herein reported is a summary of the Ph D. thesis [3] of one of the authors, and uses COM, a software [1] that, given a concept c_A in ontology *A* finds the most similar concept c_B in ontology *B*, as well as its similarity value *sv*.

5. Conclusion

A notation has been created to define ontologies. This notation possesses some improvements with respect to existing ontology languages.
OM, an algorithm to fuse ontologies has been implemented and tested; it tries to preserve the semantic of the source ontologies. It detects the inconsistencies during the merge and it solves them. OM works totally automatic, this is a great improvement to the current fusion algorithms, since these make the fusion in a semi-automatic form. That is, the user carries out the important points of the fusion. The OM notation and algorithm are part of the answer to the great necessity to give the computer (as an important entry point to the Web) the ability to accumulate knowledge and make business transactions without human intervention.

6. References

[1] A. Guzmán and J. Olivares, "Finding the Most Similar Concepts in two Different Ontologies", *Lecture Notes in Artificial Intelligence* LNAI 2972, Springer-Verlag. 129-138. ISSN 0302-9743, 2004.

[2] A. Guzmán and S. Levachkine, "Hierarchies Measuring Qualitative Variables", Lecture Notes in Computer Science LNCSW 2945, *Computational Linguistics and Intelligent Text Processing*, Springer-Verlag. 262-274, ISSN 0372-9743, 2004

[3] Alma-Delia Cuevas-Rasgado. *Ontology Merging using semantic properties* Ph. D. thesis in progress. CIC-IPN, Mexico.

[4] A. Pérez , and M. C. Suárez, *Evaluation of RDF[S] and DAML+OIL Import/Export Services within Ontology Platforms*. LNAI 2972, 109-118. 2004

[5] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinnes, P. F. Patel-Schneider, L. Andrea Stein, *DAML+OIL Reference description*, March 2001, W3C Note 18 December 2001, http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218

[6] D. Dou, D. McDermott, and Peichen Qi. *Ontology Translation by Ontology Merging and Automated Reasing*, Yale University, Computer Science Departament New Haven, CT 06520.

[7] Edith Adriana Jimenez Contreras. *Quantifying inconsistencies in sentences (facts) with symbolic values*. Ph. D. thesis in progress. CIC-IPN, México.

[8] F. Manola, E. Miller. *RDF Primer*. *W3C Recommendation*. 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[9] G. Stumme, A. Maedche. *Ontology Merging for Federated ontologies on the semantic web*. Institute for Applied Computer Science and Formal Description Method [AIFB] University of Karlsruhe D-76128 Karlsruhe, Germany.

[10] Knowledge Interchange Format. *Draft proposed*, American National Standard [dpANS] NCITS. T2/98-004.

[11] L. Deborah McGuinness, R. Fikes, J. Rice and S. Wilder, *The Chimaera Ontology Environment Knowledge*, System Laboratory CommerceOne Stanford University, Stanford, CA Mountain View, CA .

[12] M. K. Smith, Electronic Data System, C. Welty, IBM Research, D. L. McGuinnes, Stanford University, *OWL Web Ontology Language Guide*, W3C Recommendation. 2004.

[13] N. Fridman Now and A. Mark. *PROMPT: Algoritm and Tool for Automated Ontology Merging and Alignment*, Stanford Medical Informatics, Standford University, CA.

[14] Y. Kalfoglou and M. Schorlemmer. *Information-Flow-based Ontology Mapping*. Advanced Knowledge Technologies [AKT] Departament of Electronics and Computer ScienceUniversity of Southamptom.