SOLVING SEMANTIC CONFLICTS IN AUDIENCE DRIVEN WEB DESIGN

Olga De Troyer
Vrije Universiteit Brussel, WISE
Pleinlaan 2 B-1050 Brussel Belgium
Olga.DeTroyer@vub.ac.be

Peter Plessers
Vrije Universiteit Brussel, WISE
Pleinlaan 2 B-1050 Brussel Belgium
Peter.Plessers@vub.ac.be

Sven Casteleyn
Vrije Universiteit Brussel, WISE
Pleinlaan 2 B-1050 Brussel Belgium
Sven.Casteleyn@vub.ac.be

ABSTRACT

In an audience driven approach to website design, the requirements of the different audiences are modeled as separated tiny conceptual schemas comparable to views. This has several advantages, but until now it had the disadvantage that it was necessary to integrate the different schemas derived later on. In this paper we present an approach that overcomes this disadvantage. This is done by linking the concepts used to describe the information and functionality of the web site to an ontology already during conceptual modeling. Later on, the ontology is used to do the actual integration. Besides solving the integration issue, the approach has several other advantages. The role of an overall domain expert is limited; the ontology can be used to assist the designer during the modeling process; the ontology can be reused in other designs; and last but not least the use of an ontology paves the way for the semantic web.

KEYWORDS

Web Engineering, Audience driven Web Design, Conceptual Schema Integration.

1. INTRODUCTION

Web site design methods will become indispensable to avoid a Web crisis comparable to the software crisis in the 70s. If easy to maintain and well structured web sites and web applications are a priority, the need for supporting design methods becomes indispensable. More and more methods recognize the benefits of a user-centered approach (Norman, 1986; Preece, 1994). One such design method is WSDM (De Troyer, 1998; De Troyer, 2001a). In this method the approach used is called 'audience driven' to avoid confusion with the use of the term 'user-centered' in the field of Human Computer Interaction (HCI) where it has a somewhat different meaning (given further on). Methods that use a 'user-centered'-like approach put the emphasis on the users already in an early stage of the design; recognize that different types of users may exist; and that different types of users may have different needs and requirements. In this way, more usable systems can be built. Note that in the HCI community, in a user-centered approach the users are actually involved in the development process. This is not always possible for web applications because (on the WWW) most of the users are unknown, they cannot be interviewed in advance and neither be involved in the development process. This is the reason why WSDM's approach is called audience driven.

As in a user-centered approach, WSDM puts the emphasis on the different kinds of users (called audience classes), and starts the design with the identification of the requirements of the different audience classes.

Next, the different requirements are modeled separately, resulting in a number of schemas called *chunks*. Every chunk models one requirement of a specific audience class. Modeling the individual requirements of the different audience classes in separated chunks has several advantages:

1. It allows the developer concentrating on the needs of the actual users rather than on the information (and functionality) already available in the organization. The information already available in the organization is not necessarily the information needed by the users and the way the information is organized and structured in the organization is not necessarily the way the users need it.
2. It gives consideration to the fact that different types of users may have different requirements: different information or functional requirements, but they may also require a different structure or terminology for the information. By modeling the requirements for each audience class separately we can give due consideration to this. E.g. we can avoid that the user is overloaded with information of which most is not relevant to him.

The disadvantage of using separated chunks is that they need to be integrated later on. This is needed to relate the information modeled in the different chunks to each other (different chunks may deal with the same information). Also an integrated schema may be useful for the implementation of the web site if one wishes to maintain the information in a database. In this case, the integrated schema will be the conceptual schema of this database. It is possible that the web site will use an existing database. In this case the different chunks need to be defined as views on the schema of this database. How this should be done is outside the scope of this paper. Here, we will only consider the case that the chunks need to be integrated. The need to integrate the different chunks looks like a serious disadvantage of the method. However, we have developed an approach to overcome this. This is done by linking (already during modeling) the concepts used to describe the information and functionality of the web site to an ontology. In this way the integration can be done in a semi-automatic way. In this paper, we explain this approach and we also identify the additional benefits of using this approach. The most important one is that it paves the way to the semantic web.

The problem of integrating chunks in WSDM is strongly related to the problem of *schema integration* and more in particular *view integration*. When designing a large information system, the task can become too complex for a single person. Therefore, the modeling can be divided amongst several persons. Each of these focuses on a particular part of the universe of discourse and constructs a conceptual schema, called a *local schema*, for that part. Afterwards, the local schemas are integrated in a *global schema.* This situation is very similar to the problem of chunk integration, but nevertheless there are some remarkable differences:

1. Web sites can offer information as well as services (functionality) to the user. In WSDM, chunks are used to model information as well as functional requirements. In information system design, only information modeling is taken into account during the database design and functionality is not modeled using local schemas.
2. When defining constraints in local database schemas, these constraints are defined to allow checking the integrity of the database and are valid for the entire domain modeled. This is not the case for constraints in chunks. Because chunks are used to model the requirements of a certain audience class, the constraints in a chunk only express possible constraints formulated in the requirement. These are not necessarily the constraints that apply in general for the information considered. The constraints in a chunk are only valid for the audience class of that chunk and in the context of the requirement modeled. Because of this different interpretation of constraints, the integration of constraints has to be treated differently.

Due to these differences, the integration techniques developed in the context of databases cannot be used as such. The paper is structured as follows. In section 2, a short overview of WSDM and chunks are given. Section 3 reviews the work already done in the context of schema integration. In section 4 we present our approach. In section 5, advantages and further research are considered and section 6 gives conclusions.

2. WSDM

2.1 Overview

Here, we give a short overview of WSDM (Casteleyn, 2001; De Troyer, 2001a; De Troyer, 2001b; Casteleyn, 2002). The first step is to define t*he Mission Statement*. The Mission Statement should express the

purpose and the subject of the web site and declare the target audience. Based on this Mission Statement a two-step Audience Modeling phase is performed. In the first step, Audience Classification, the different kinds of users are identified and classified. In the next step, Audience Class Characterization, the characteristics of the different Audience Classes are given. The result of the Audience Modeling is a set of *Audience Classes*, ordered into an *Audience Class Hierarchy*, together with an informal description of all their requirements and their characteristics.

Next, the Conceptual Design is done. This phase is divided in two steps: Information & Functional Modeling and Navigational Design. During Information & Functional Modeling, the information & functional requirements of the different Audience Classes are modeled, resulting in a set of chunks for each audience class. Next, as discussed in the introduction, these chunks need to be integrated into a single model. During Navigation Design, we design the conceptual structure of the web site and model how the members of the different Audience Classes will be able to navigate through the site.

During Implementation Design the page structure as well as the 'look and feel' of the web site is defined, taking into consideration the usability requirements and characteristics of the Audience Classes. The Implementation design should also provide the specification for the logical data design (if needed). This could be any suitable data definition format: a relational database design, XML, RDF, …. The last phase, Implementation, is the actual realization of the web site using the chosen implementation environment.

2.2 Information & Functional Chunks

During conceptual design, the information and functional requirements of the different audience classes are elaborated, decomposed into elementary requirements and each elementary requirement is modeled by means of a *chunk*. Currently, Object Role Modeling (ORM) (Halpin, 2001) is used for this (an extended form of ORM is used to model functionality. The extensions are introduced in De Troyer (2001b).). ORM is a well-known database modeling method, similar to ER (Chen, 1979). ORM has all the standard data modeling features: *object types*, *relationships*, and several types of constraints. Typical for ORM is its distinction between lexical and non-lexical object types rather than between entities and attributes as in ER. Lexical object type instances correspond to things that can be written down or are utterable (e.g. a name, a number), while non-lexical object type instances are non-utterable things (e.g. a person, a paper, a track) that can only be referenced by humans through lexical object type instances or by pointing to it (e.g. we can point to a person or refer to a person by means of his name). Subtypes of object types have the meaning that each instance of the subtype is also an instance of the supertype, e.g. each Man is a Person. A relationship is modeled by means of the *roles* the object types involved play in the relationship (usually there are two roles).

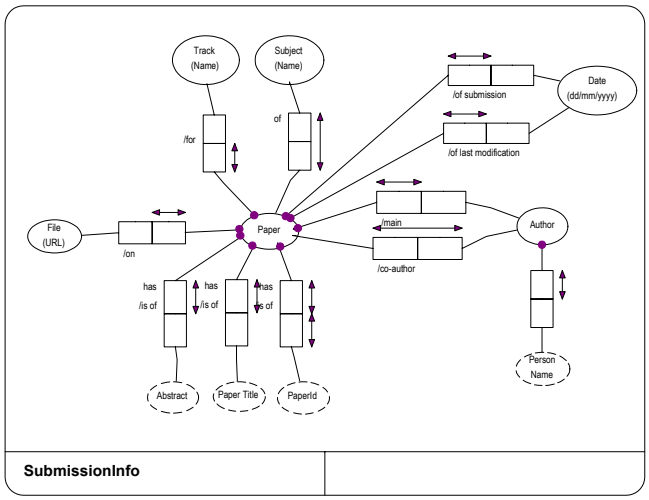

Figure 1: An example chunk

Graphically, non-lexical object types are represented as solid ellipses and lexical object types as dashed ellipses. Roles are represented as rectangles with their corresponding labels denoting the name of the role. A

double arrowed line on top of a role denotes a uniqueness constraint and a solid dot denotes a mandatory role. For a more detailed specification of ORM, we refer to Halpin (2001).

Figure 1 shows an example of a chunk. The example is taken from the Conference Review Case Study (http://www.dsic.upv.es/~west2001/iwwost01/). It is a chunk for the audience class Author and shows the information an author is allowed to see about a paper that he submitted to the conference.

3. SCHEMA INTEGRATION IN GENERAL

A lot of research on schema integration has been done in the past. This has resulted in a clear understanding of the problems involved. The first fundamental problem in integration is the existence of semantic conflicts, also referred to as *semantic heterogeneity* (Hull, 1997). Two objects with the same name can refer to different concepts in the domain; and two objects with different names can refer to the same concept. Another problem is the fact that a domain can be modeled in different ways. This results in schemas that are differently structured but which describe the same domain (or part of it). This problem is referred to as *structural heterogeneity* (Hull, 1997). Next to these issues, a number of requirements for the integration process have been formulated in the past. For our integration of chunks, the following requirements have been adopted from Batini (1986):

- Completeness and Correctness: The integrated schema must cover all information concepts present in any chunk correctly. Note that functionality will not be presented in the integrated schema (there is no need for this because the role of the integrated schema is to link the concepts in the different chunks and to be useful as a database schema later on).
- Minimality: If the same concept is represented in more than one chunk, it must be represented only once.
- Understandability: The integrated schema should be easy to understand for the designer and the end user. This implies that among the several possible representations of results of integration, the one that is (qualitatively) the most understandable should be chosen.

The suggested solutions for schema integration found in the literature are mostly variations on the same concept: *integration assertions* (Seth, 1989; Savasere, 1991; Johannesson, 1993a). These assertions are made by the schema integrator and define certain relationships (equality, subtype of, ...) between the different objects in the conceptual schemas. These integration assertions are necessary to solve the problem of semantic heterogeneity. To deal with the problem of structural heterogeneity, the notion of *schema transformation* was introduced. The goal is to standardize the schemas by transforming them into some *normal form* before integrating them. A number of basic schema transformations were developed. For more detail, we refer to Johannesson (1993b). Most solutions found in the literature based on integration assertions follow the same framework. This framework is composed of four main steps (Batini, 1986):

1. *Pre-Integration phase*: Used when different schemas use different languages to model the domain. The purpose is to translate the local schemas into schemas using a common language.
2. *Schema Analysis phase*: Involves analyzing and comparing schemas in order to determine correspondences, in particular different representations of the same concept. Possible existing conflicts between local schemas need to be identified and resolved. Three main activities can be identified in this schema analysis phase:
   - *Name comparison*. There are two problematic relationships between names: *synonyms* and *homonyms*.
   - *Detecting inter-schema properties*. Constructs appearing in different schemas may have certain constraints in common, e.g. an object type in one schema that is a subtype of an object type in another schema.
   - *Structural comparison*. To identify cases where the same aspects of the domain have been modeled using different constructs.

   The first two items are in most cases solved by identifying a set of integration assertions, the last one by using schema transformations.
3. *Schema merging*: Using the information gathered in the previous phase, the process of merging is a rather automatic activity.
4. *Restructuring phase*: Used to restructure the integrated schema, constructed in the previous phase, for enrichment, quality improvement or error correction.

4. THE INTEGRATION APPROACH OF WSDM

In our approach to integrate chunks we exploit the fact that we know in advance that the chunks need to be integrated. In addition, an ontology is used to capture the semantic information needed. Furthermore, the approach is based on the classical schema integration framework, given in section 3. The ontology used in our chunk integration approach is described in section 4.1; section 4.2 discusses the different phases used in the integration process and how these differ from the classical phases.

4.1 The Ontology

Semantic information is needed to detect and to solve possible conflicts between different chunks. An ontology is used to for this. From previous research in schema integration we know that the semantic information needed is mainly the different kind of relationships that can exist between the different concepts in the domain. Ontologies are defined as an explicit specification of a conceptualization (Gruber, 1993). They describe concepts in a domain as well as relationships between these concepts and the terminology used. Therefore, ontologies are well suited for a formal description of this type of semantic information.

The ontology used for the integration will be built during conceptual design. During the modeling of the different chunks the semantic information needed for this ontology is collected. We deliberately have incorporated the collecting of semantic information into the conceptual modeling process. Although this may slightly slow down the modeling process we believe that it is a better and a less time consuming solution than collecting it afterward. If semantic information has to be entered after the modeling is finished (like in the case of integration assertions) more errors will be made and more time will be needed (we may not always remember the exact meaning of a concept). An additional advantage of incorporating this phase in the modeling process is that, in case multiple designers are involved, each designer can enter the semantic information for his own concepts and relate it directly to the concepts already entered by the other designers or reuse concepts from other designers. In our ontology we distinguish the following types of *concepts*:

- *Object Concepts.* As in ORM we distinguish between Lexical and Non-lexical Concepts. E.g. 'person' is a Non-lexical Concept, 'person name' is a Lexical Concept. We make this distinction because we can use it to reduce the amount of information that needs to be specified by the designer (see section 4.2).
- *Relationships.* Relationships in the ontology express the relationships that may exist in the domain between objects. An example is 'works-for'; the Object Concepts involved are 'person' and 'company'. Note that the concept 'relationship' used here is different from the concept 'relationship' in ORM. In fact the concept of 'relationship' used in the ontology community is similar to the concept 'role' used in ORM. To be consistent with the ontology terminology we prefer to call the concept 'relationship' rather than 'role'.
- *Tuples.* A tuple is a grouping mechanism for concepts. As an example, the combination of the Lexical Object Types 'first name' and 'family name' is a tuple. In case of Object Concepts, tuples can be compared to complex object types.

Each concept in the ontology is uniquely identified by an *identifier*; has a set of *labels* that are possible names for this concept (e.g. 'film', 'movie'), one label will be the *preferred label* and should be unique; has a *comment* that is a short textual description. Relationships also refer to the domain and the range of the relationship. Currently DAML (Connolly, 2001; Horrocks, 2002) is used as format for the ontology. Below is an example of a Non-lexical Concept "Plane" and a Relationship "has number of seats" in DAML:

```
<daml:Class rdf:ID="023">
  <preferredLabel>Plane</preferredLabel>
  <label>Airplane</label>
  <subtypeOf rdf:resource="#014" />
<rdfs:subClassOf rdf:resource="#NonLexicalConcept"/>
</daml:Class>
```

```
<daml:Property rdf:ID="024">
  <label>has number of seats</label>
  <comment>number of seats of a plane </comment>
  <domain rdf:resource="#023" />
  <range rdf:resource="#016" />
  <rdfs:subPropertyOf rdf:resource="#relationship" />
</daml:Property>
```

The ontology also contains *dependencies* that exist between concepts. The following types of dependencies are distinguished:

- ***EquivalentTo***. Used to express that a concept is equal to another concept (at a semantic level). E.g. we can express that the concept *name* is equal to the tuple concept (*first name*, *family name)*.

- ***SubtypeOf***. Used to express that one concept is a subtype of another concept, which means that its population is a subset of the population of the other concept. E.g., the concept *Man* is a subtype of the concept *Person*.
- ***OverlapWith***. Used to express that two concepts are partially overlapping, this means that their populations have a nonempty intersection. E.g., *Student* and *Employee* are overlapping concepts because some student can be an employee and some employee can be a student.
- ***Part of***. Used to express that one concept is a part of (or component of) another one concept. E.g., the concept *first name* is a part of the concept *name*.

Part of the information in the ontology is entered by the designer(s) during conceptual design while making the chunks; the rest is derived from the information provided in the ontology. This is discussed in section 4.2.

To be useful for the integration process, the concepts in the ontology need to be linked with the elements used in the chunks. Therefore, every chunk element (object type, role, relationship, …) will refer to exactly one ontology concept. In this way, we can have two chunk elements with different names that refer to the same ontology concept. E.g. by using the same ontology concept for the lexical object types *EmployeeId* and *AdminNumber* we state that an EmployeeId is the same concept as an AdminNumber (*EmployeeId* and *AdminNumber* will be in the set of labels of this concept). Similar, we can have two different chunk elements with the same name that refer to different ontology concepts.

4.2 The Phases of the Integration Process

We now sketch the different phases of the chunk integration process. As already indicated the phases are based on the classical integration framework. However, some modifications were necessary.

1. *Pre-integration phase.* While in traditional schema integration this phase is used to translate the local schemas into a common language, this is not needed here because all chunks are modeled using the same language (an extended form of ORM). We use this step for a different purpose. In fact, our pre-integration phase already starts when creating the different chunks. Its goal is to collect the necessary semantic information for the concepts introduced during modeling. The best moment to collect semantic information about a concept is when it is introduced. Therefore, if a new concept is used (by a designer) it is introduced in the ontology and possible dependencies with other concepts should be identified (by the designer). The designer can also provide a comment explaining in an informal way the meaning of the concept. Other designers can use these comments to quickly identify relevant concepts in the ontology and to investigate possible dependencies. Also note that the designer does not have to enter semantic information for all concepts introduced. In fact, only semantic information for lexical object types and roles need to be given. The semantic information for non-lexical object types can be derived (see next phase).
2. *Schema Analysis phase.* In the previous phase, semantic information about lexical object types and roles is collected. In this phase the gathered information is analyzed to establish the dependencies that exist between the concepts for the non-lexical object types in the different chunks. This is done by means of a set of rules. However, due to space limitations these rules are omitted. They are given in De Troyer (2003) (in that paper also formal definitions for the different concepts used are given). No user interaction is needed for this; the semantic information gathered in the previous phase is sufficient. Note that also this phase is actually performed together with the Pre-Integration phase (and thus during modeling). Whenever semantic information is added or changed, the information is analyzed and the dependencies between the concepts are updated.

   Note that the *Name Comparison* activity, as described in section 3, is actually performed in the Pre-Integration phase instead of in this phase. If a designer introduces a certain concept in the ontology, it cannot have the same preferred label as another already existing concept in the ontology. This means that a unique label must exist for the ontology concept, while different names are possible in the chunks. The identification of inter-schema properties as described in section 3 is divided over this phase and the previous phase. The inter-schema properties concerning lexical object types and roles are handled in the Pre-Integration phase; the ones concerning non-lexical object types are taken care of in this phase.
3. *Schema Merging phase.* When conceptual modeling is finished, the ontology constructed defines all concepts and relationships used in the chunks. This information can then be used to construct the integrated schema. Non-lexical Concepts in the ontology will result in non-lexical object types in the

integrated schema; Lexical Concepts in the ontology, which are not a subtype of some other Lexical Concept, will result in lexical object types (ORM does not support subtypes of lexical object types); and relationships will result in roles. Subtype dependencies between Non-lexical Concepts will be modeled as subtype links between the corresponding non-lexical object types. Also the constraints defined in the chunks need to be integrated. Because chunks can be functional chunks as well as information chunks this need some attention (more information on this can be found in De Troyer (2003)).

4. *Restructuring phase.* As in the classic schema integration framework, the purpose is: enrichment, quality improvement and error correction. Errors can occur due to inconsistent chunks or by conflicting semantic meanings. The designer should correct these errors and run the integration process again.

5. ADVANTAGES & FURTHER RESEARCH

As already indicated during the explanation, this approach has many advantages:

- By capturing and generating semantic information during the design phase it is possible *to give advice* to the designers during modeling. If a designer uses a concept that is already known to the ontology, the system may suggest relationships and other concepts already used for this concept. This can *ease the conceptual design process*, and *a higher degree of consistency* can be achieved. It gently encourages the designer of a chunk to use the same names and to consider the relationships used in other chunks.
- The *role of the overall domain expert* can be *limited* considerable. In traditional database integration solutions a domain expert, who should know the entire domain, has to provide all the semantic information during the integration process to solve any possible conflicts. This is a cumbersome and erroneous task. In our approach, the role of the domain expert is minimized because each individual designer can provide semantic knowledge and only needs to have knowledge about the part of the domain he or she is designing. In this way, the task of the overall domain expert will be a lot smaller. In case of large domains, this is a great benefit because it is practically impossible for a single person to know every detail of a large domain. In addition, *less semantic information* than usual need to be given by the designers because some of the semantic information can be derived. Only if conflicts are detected the designer has to solve them. This has the benefit of being *an additional consistency check*.
- The ontology created throughout the conceptual design phase can be *reused* when the web site needs to be extended or adapted.
- Linking the conceptual design to an ontology *paves the way to the semantic web*. In the semantic web, the information in a web site is annotated with ontology concepts. This allows exploring the knowledge available in the ontology e.g. during querying. In our approach, the annotation with ontology concepts comes for free because each chunk concept will be related to a concept in the ontology.
- The approach is *not limited to web site design*. It is also useful in information system modeling.

The approach is validated by means of a prototype tool. In the current approach, the ontology used is constructed during the conceptual design. However, for some domains an ontology may already exist. Therefore in further research we will investigate how such an existing ontology can be used. For this purpose it may be useful to consider integration methods for ontologies (McGuiness, 2000; Noy, 2000).

The use of an ontology solves the problem of semantic heterogeneity, but not the problem of structural heterogeneity. Some provisions have been made for this; e.g. partOf and overlapWith dependencies will be useful in this respect. However, more research needs to be done on this topic.

6. CONCLUSIONS

In this paper, we propose to link the concepts used during conceptual modeling (to describe the information and functionality that will be offered by a web site) to an ontology. The approach is proposed in the context of an audience driven web design method. The audience driven approach to web design has the advantage that already from the start of the design process the designer should concentrate on the requirements of the different audience classes. One result of the design process is a number of chunks, which are tiny conceptual schemas modeling the requirements of the different audience classes. These chunks need to be integrated into the so-called Business Information Model. To do this integration, we collect, already

during conceptual modeling, the necessarily semantic information from the designer in an ontology. Based on the information in the ontology the integration process can be done nearly automatically. This approach has several advantages. Each designer enters the semantic information of the part he is modeling; the ontology can be consulted and used for advising the designers; and by linking the conceptual design to an ontology our approach also prepares for the semantic web.

REFERENCES

Batini, C. et al, 1986. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computer Surveys*, Vol. 18, No. 4, pp. 323-364.

Casteleyn, S. and De Troyer, O., 2001. Structuring Web Sites Using Audience Class Hierarchies. *Proceedings of DASWIS 2001 (ER) workshop*. Yokohama, Japan.

Casteleyn, S. and De Troyer, O., 2002. Exploiting Link Types during the Web Site Design Process to Enhance Usability of Web Sites. *Proceedings of the IWWOST 2002 (ECOOP) workshop*. Malaga, Spain.

Chen, P.P.S., 1976. The Entity-Relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, Vol. 1, No. 1, pp. 9-36.

Connolly, D. et al, 2001. DAML+OIL (March 2001) Reference Description.

De Troyer, O. and Leune, C., 1998. WSDM : A User-Centered Design Method for Web Sites. *Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference, Elsevier*, pp. 85-94.

De Troyer, O., 2001a. Audience-driven web design. *Information modeling in the new millennium*, Eds. Matt Rossi & Keng Siau, IDEA GroupPublishing, ISBN 1-878289-77-2

De Troyer, O. and Casteleyn, S., 2001b. The Conference Review System with WSDM. *IWWOST 2001 (http://www.dsic.upv.es/~west2001/iwwost01/)*, Valencia, Spain.

De Troyer, O. et al, 2003. Conceptual View Integration for Audience Driven Web Design. *Internal WISE Report*.

Gruber, T.R., 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220.

Halpin, T., 2001. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design, 1st Edition*. Morgan Kaufmann Publishers, San Francisco, USA.

Horrocks, I., 2002. DAML+OIL: A Description Logic for the Semantic Web. *IEEE Data Engineering Bulletin*, Vol. 25, No. 1, pp. 4-9.

Hull, R., 1997. Managing Semantic Heterogeneity in Databases: a Theoretical Prospective. *PODS '97, proceedings of the 16th ACM SIGMOD Int. Conference in Management of Data*, New York, USA, pp. 51-61.

Johannesson, P., 1993a. Schema Transformations as an Aid in View Integration. *Schema Integration, Schema Translation, and Interoperability in Federated Information Systems*.

Johannesson, P. 1993b. A Formal Basis for Schema Integration. *Schema Integration, Schema Translation, and interoperability in Federated Information Systems*.

McGuiness, D.L. et al, 2000. An Environment for Merging and Testing Large Ontologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Colorado, USA

Norman, D. A., 1986. Cognitive Engineering. *User Centred System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Assoc., pp. 31-65

Noy, N. F. and Musen, M. A., 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Seventeenth National Conference on Artificial Intelligence (AAAI2000)*.

Preece, J., 1994. *Human-Computer Interaction, 1st Edition*. Addison-Wesley Pub Co.

Savasere, A. et al, 1991. On Applying Classification to Schema Integration. *Proceedings of First Intl. Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan.

Sheth, A. and Gala, S., 1989. Attribute Relationships: An Impediment in Automating Schema Integration. *Workshop on Heterougeneous Database Systems*, Chicago, USA.