Deriving Conceptual Schema from Domain Ontology: A Web Application Reverse Engineering Approach

Sidi Benslimane, Mimoun Malki, and Djelloul Bouchiha
Computer Science Department, University of Sidi Bel Abbes, Algeria

Abstract: *The heterogeneous and dynamic nature of components making up a web application, the lack of effective programming mechanisms for implementing basic software engineering principles in it, and undisciplined development processes induced by the high pressure of a very short time-to-market, make web application maintenance a challenging problem. A relevant issue consists of reusing the methodological and technological experience in the sector of traditional software maintenance, and exploring the opportunity of using reverse engineering to support effective web application maintenance. This paper presents reverse engineering approach that help to understand existing undocumented web applications to be maintained or evolved, through the extraction from domain ontology of conceptual schema describing a web application. The advantage of using ontology for conceptual data modelling is the reusability of domain knowledge. As a result of it the conceptual data model will be made faster, easier and with fewer errors than creating conceptual data model in usual way. We demonstrate the value of the approach by providing an implementation that exhibits appropriate characteristics.*

Keywords: *Reverse engineering, web application, ontology, conceptual schema, conceptual modeling.*

Received July 29, 2008; accepted December 23, 2008

1. Introduction

The new possibilities offered by web applications are pervasively and radically changing several areas. Unfortunately, web applications must cope with an extremely short development/evolution life cycle. Usually, they are implemented without producing any useful documentation for subsequent maintenance and evolution, thus compromising the desired high level of flexibility, maintainability, and adaptability that is de-facto necessary to compete and survive to market shakeout. To reconstruct already existing web applications that do not respect the development life cycle, the reverse engineering is essential. The reverse engineering of web application has been addressed in various ways. Some research works take an interest in the evolution of the presentation [19, 33] while others focus on restructuring HTML static pages into dynamic ones [27]. The literature also presents several approaches aimed at obtaining web sites abstract representation. Estiévenart *et al.* [10] extract page contents as XML documents structured by expressive DTDs or XML Schemas. Paganelli and Paterno [24] analyze web site code to automatically reconstruct the underlying logical interaction design. Gaeremynck *et al.* [12] translate the visual layout of HTML forms into a semantic model. Stroulia *et al.* [30] produce HTML UIs by integrating data of several web pages. Stroulia *et al.* [6] and Bellettini *et al.* [3] use UML diagrams to model views of web applications at different abstraction levels. Recently, some approaches consider the ontology creation as the goal for web application reverse engineering [4, 9, 32].

Most of web application reverse engineering approaches follows a strictly bottom-up abstraction process. This bottom-up approach is less adequate if a partial design for data structure (documentation, knowledge of a designer or domain expert) and/or domain ontology of the application is available. In practice, most conceptual schemas of information systems and databases are developed essentially from scratch.

This paper deals with a new approach to that development, consisting on analyzing HTML web pages to derive a first cut version of the conceptual schema modelling the web application based on domain ontology. The important requirement for developing conceptual data models is to reduce efforts, costs and time. This requirement can be implemented by the explicit use of domain ontology for automatic or semiautomatic generation of conceptual schema. A significant corollary of this research is that it is possible to generate a domain conceptual schema from a given ontology by using a set of transformation rules implemented within a transformation engine component.

A number of methods were proposed to develop conceptual data models, but only few deals with

knowledge reuse. The approach of deriving conceptual schemas of information systems and databases from ontologies has not been explored in detail in the literature, yet we believe it may yield important benefits. Gibson and Conheeney [13] suggest that requirements models should be "developed by specializing, refining or adapting selected parts of the relevant domain model, if this is appropriate". Swartout *et al.* [31] use the SENSUS ontology to derive the conceptual schema for an air campaign planning system. Peterson *et al.* [25] present The Knowledge Bus a system which generates database and programming interfaces, having as input a conceptual schema derived from the Cyc ontology. Conesa *et al.* [5] focus on the problem of pruning ontologies. The input to method is the ontology and the set of concepts of interest. The problem arises when the reused ontology is large and it includes many concepts which are superfluous for the final conceptual schema. El-Ghalayini *et al.* [8] study the role that ontologies can play in establishing conceptual data models during the process of information systems development. A mapping algorithm has been proposed to generate a conceptual data model from a given domain ontology.

However, most of the proposed transformation methods make some assumptions that are not usually available (e.g, the presence of formal specification of the information system requirements (domain events, queries) or the explicit definition of the concepts of interest). Moreover when the domain ontology is large, the final conceptual schema may include many superfluous concepts and relation.

2. Overview of the Approach

In this section, we describe how conceptual schema modeling a web application, can be derived from domain ontology using useful information extracted from HTML pages. The approach consists of four phases. The first phase is devoted to the extraction of useful information from HTML pages, including forms, tables, and lists that represent candidate elements for the next phase. The second phase is concerned with the identification of relevant ontological constructs. The identification is based on a matching between the candidate elements and the constructs (concepts, relations, attributes and axioms) of domain ontology using semantic distance techniques. The third phase consists in inferring new constructs (concepts and/or relations) before generating the conceptual schema. The last phase consists in generating a conceptual schema through the identified constructs. The following paragraphs describe each of these phases.

2.1. Extraction of Candidate Elements

The extraction phase aims to retrieve the pertinent elements coded on each web application's HTML page. It is performed in four steps:

- Pre-processing: this step takes as input HTML pages, corrects them, proceed to some cleaning by removing stop words and eliminating useless tags such as those of layout (e.g., <b>, <i>), and preserving useful tags, which carry information to be processed in the following stages (e.g., <form>, <table>, <td>, <tr>, <ul>, <li>). The result of this step is a coded sequence describing the structure of the HTML page.
- DOM construction: the second step permits the generation of the DOM (Document Object Model) representations of cleaned HTML pages in order to facilitate their manipulation. DOM representations describe the physical views of the web application HTML pages (one physical view per HTML page).
- Candidate elements identification. In the third step, DOM representations are parsed to obtain a set of elements. An element is either: forms, tables, or lists. Each element has a name and a set of attributes (fields).
- Morphological analysis: in the last step, a morphological analysis is applied to the obtained elements and their attributes. It consists in performing word stemming (lemmatization). Auxiliary information like stop words list, English lexicon (WordNet in this particular case) are used to perform the necessary linguistic transformation (e.g., morphological analysis of 'running-away' is 'run away').

The result of the extraction phase is a set of candidate elements that are useful for the identification of relevant constructs of domain ontology in the next phase.

2.2. Ontological Constructs Identification

During this phase a set of ontological constructs are detected while matching candidate elements to the constructs of domain ontology. The matching aims to quantify how much two entities are alike by calculating semantic distance between them.

2.2.1. Matching Strategies

The semantic distance calculation in our approach is based on different similarity measures. The matching is achieved at three levels: name-based matching, lexical-based matching, and structure-based matching.

Name-based matching: is to compare elements with equal names. At this level string similarity *SimN* measure is used based on edit distance formulated as:

$$SimN(e_1,e_2)=\max(0,\frac{\min(|e_1|,|e_2|)-ed(e_1,e_2)}{\min(|e_1|,|e_2|)})\in[0,1] \quad (1)$$

where *ed* is the edit distance formulated by Levenshtein [18]. It measures the minimum number of token insertions, deletions, and substitutions required to transform a string e_1 into another string e_2. In

addition, string similarity can be defined and measured based on common substrings (e.g., representedBy ≅ representative).

Lexical based matching: the technique reviewed above is efficient but need to be completed. Two terms may be similar even if they are completely differently spelt. This is the example of synonyms. More generally, two terms having a related sense deserve to be somehow related. In order to be able to capture these relations between the terms, it is necessary to get their semantics. At this level, the lexical-based similarity measure explores the semantic meanings of the word constituents by using external resources, like user-defined lexica and/or dictionaries (e.g., Wordnet, [11]) to help identify synonyms in matching. Several approaches have been proposed, especially those relying on WordNet. For two entities e_1 and e_2, the lexical semantics similarity measure *SimL* can be given using the WordNet synsets (i.e., term for a sense or a meaning by a group of synonyms) based on the formula:

$$SimL(e_1, e_2) = 1 / length(e_1, e_2) \quad (2)$$

where *length* is the length of the shortest path between two entities e_1 and e_2 using node-counting.

Structure based matching: at this level, the attributes of elements are matched according to a strategy of calculation. If the attributes of two elements are equal, the elements are also equal. To calculate the semantic distance between two groups of entities, many strategies are proposed. Some of strategies are used for hierarchical clustering [20]:

- Single linkage, where the distance between groups is defined as the distance between the closest pair of objects.
- Complete linkage, where the distance between groups is the distance between the most distant pair of objects. -Average linkage, where the distance between two clusters is defined as the average of distances between all pairs of objects.

To calculate the similarity between sets in ontologies, [21] uses a strategy called Multidimensional scaling which is a statistical technique for the calculation of the semantic distance between two sets of entity. For that, each entity is described through a vector representing the similarity to any other entity contained in the two sets. For both sets a representative vector can be created by calculating an average vector over all individuals. Finally, the cosine between the two set vectors is determined through the scalar product as the similarity value. The formula of calculation is as follows:

$$SimS(E, F) = \frac{\sum_{e \in E} \vec{e}}{\left|\sum_{e \in E} \vec{e}\right|} \times \frac{\sum_{f \in F} \vec{f}}{\left|\sum_{f \in F} \vec{f}\right|} \quad (3)$$

with entity set $E = \{e_1, e_2, ...\}$,

$$\vec{e} = (sim(e, e_1), sim(e, e_2), ..., sim(e, f_1), sim(e, f_2), ...)$$

F and f are defined analogously.

2.2.2. Matching Process

A single similarity measure may be unlikely to be successful. Hence, combining different similarity measures is an effective way. For this purpose, many approaches combining the results of several independently executed matching algorithms are proposed [7]. There are two kinds of approaches to combine multiple similarity measures: hybrid and composite combination. Hybrid approach is the most common where different matching criteria (e.g., name, structure) are used within a single algorithm. Typically these criteria are fixed and used in a specific way. By contrast, a composite matching approach combines the results of several independently executed matching algorithms, which can be simple of hybrid. This allows for a high flexibility, as there is the potential for selecting the matching algorithms to be executed based on the matching task at hand. In this paper, all the methods and strategies described above have been implemented, and can be selected for the calculation of semantic distance throughout the matching process of our reverse engineering approach. We have developed an algorithm that takes as input a set of candidate elements, and produces as output a set of relevant ontological constructs. A matching process is carried out between three vectors.

- The Vector of candidate Elements (V_E) consists of useful information extracted from HTML pages. A vector element can be either: form, table, or list. Each element E_i is described by a name and a set of attributes.
- The Vector of ontological Concepts (V_C) contains the concepts of domain ontology. Each concept E_c is described by its name and a set of reattached properties.
- The Vector of ontological Relations (V_R) represents the relations of domain ontology. Each relation E_r is described by its name and the original and the result concepts that it relates. E_r can be either: taxonomic (Is-as), or no-taxonomic relation.

The vectors are matched together while calculating their semantic distance. The match returns a value between 0 and 1, where 1 stands for perfect match and 0 for bad match. For the intermediate values, we define a threshold. If the match result is greater than or equal to the threshold then the two elements are equivalent, otherwise they are different. Relevant ontological constructs are identified by two rules.

Rule i_1*:* concepts identification. let $E_i \in V_E$ *a* candidate element, $E_c \in V_C$ *an ontological concept,* $E_r \in V_R$ an ontological relation, E_{c1}, $E_{c2} \in V_C$ the concepts that E_r relates. If match result between E_i and

E_c is greater than or equal to the threshold, then E_c is marked as identified concept. If match result between E_i and E_r is greater than or equal to the threshold, then E_{c1} and E_{c2} are marked as identified concepts.

Rule i_2: relations identification. Let $E_r \in V_R$ an ontological relation, E_{c1}, $E_{c2} \in V_C$ the concepts that E_r relates. If E_{c1} and E_{c2} are marked as identified concepts, then E_r is marked as identified relation. Ontological relation can exist without specifying any concepts that it might relate to [1], but this is not the case in conceptual models. Therefore, we believe that it is sufficient to map only relations that are related to concepts.

Algorithm (ontological constructs identification).

Input:
Vector V_E of the extracted useful-information;
Vector V_C of domain ontology concepts;
Vector V_R of domain ontology relations;
String similarity measure between elements names SimN;
Lexical similarity measure between elements names SimL;
Structure similarity measure between elements groups SimS;
Threshold k.

Output:
A vector of identified concepts.
A vector of identified relations.

```
BeginAlgo
 // Concepts identification
 For each element Ei ∈ VE do
  For each element Er ( VR do
   If SimN(Ei.name, Er.name) > k or
   SimL(Ei.name, Er.name) > k then
        marked Ec1 and Ec2, the tow concepts related
        by Er as identified concepts
   Endif
  Enddo
 Enddo
 For each element Ei ( VEdo
  For each element Ec ( VC do
   If SimN(Ei.name, Ec.name) > k)  or
   SimL(Ei.name, Ec.name) > k )  then
       marked Ec as identified concept
  ElseIf SimS(Ei.group, Ec.group) > k then
      marked Ec as identified concept
   Endif
  Enddo
 Enddo
  // Relations identification
  For each element Er ∈ VR do
   If Ec1 and Ec2 the concepts that Er relates are
        marked as identified concepts in VC then
        marked Er as identified relation
  Endif
 Enddo
EndAlgo
```

2.3. Enrichment

Enrichment phase consists in inferring new constructs (concepts and/or relations) before generating the conceptual schema describing the web application. The following rules summarize the mechanisms that permit the deduction of new constructs.

Rule e_1: taxonomic enrichment. Let $E_c \in V_C$ an ontological concept. If E_c is marked as identified concept in the previous phase then all the sub-concepts and the super-concepts of E_c are marked as identified concepts in V_C.

Rule e_2: no-taxonomic enrichment. Let $E_r \in V_R$ an ontological relation, E_{c1}, $E_{c2} \in V_C$ the original (domain) and the result (range) concepts that E_r relates. If the concept E_{c1} is marked as identified concept in V_C, then the concept E_{c2} is also marked as identified concept in V_C, and vice-versa.

2.4. Conceptualization

Conceptual modelling plays a crucial role in the process of information systems development. Conceptual models translate and specify the main data requirements of the user requirements in an abstract representation of selected semantics about some aspects of a real-world domain. Systems analysts seek to capture and represent all relevant problem domain entities and their relationships. In addition, conceptual modelling languages and notations were introduced to represent conceptual models using a collection of modelling elements.

In this section, we propose a set of mapping rules to generate a conceptual model from the domain ontology. However, the concepts used in knowledge representation languages in a machine readable form, i.e., ontology, are very close to those used to represent data in conceptual models; both models have much in common. Although the aim of each model is different, reverse engineering the domain ontology assists in developing the conceptual model.

In general, any Conceptual Model (CM) can be considered as a 4-tuple: CM= (CE, CR, CA, CS), where CE, CR, CA, CS stand respectively for entities, relationships, attributes, and constraints. Whereas, ontological structure is a 5-tuple O=(C, A^C, R, H^C, X), where:

- C is a finite set of concepts.
- A^C is a collection of attribute sets about concepts.
- R is a finite set of no-taxonomic relations; each relation has a pair of concepts (domain and range).
- H^C is called concept hierarchy or taxonomy, which is a directed relation $H^C \subseteq C \times C$.
- X is a set of axioms that describe additional constraints on the ontology.

The rules below briefly summarize the transformation rules used in this research and are part of conceptualization phase.

Each rule is followed by an example illustrating the mapping between Ontology Web Language (OWL) elements [29], and Unified Modelling Language (UML) constructs [23]. However, we believe that our approach can be applied to any similar language.

Rule c_1: ontology concept C becomes CM entity. The ontology concept name becomes the CM entity name. OWLClass element is transformed into entity element in UML. All classes in OWL are identified by URI. An entity in UML is identified by name.

Example: according to rule c_1, the OWL class "Destination": <owl: class rdf: ID="Destination"/>
Is translated as shown in Figure 1.

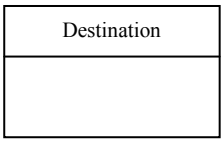


Figure 1. UML Class.

Rule c_2: the collection of attribute sets A^C about concept *C* becomes a CM Attribute of the corresponding CM Entity. OWLDataTypeProperty element is transformed into an attribute element in UML. An attribute in UML represents a common characteristic of some entity instances. OWL data type properties are used to link individuals to data values. A data type property is defined as an instance of the built-in OWL class owl:DatatypeProperty.

Example: according to rule c_2, the OWLDataTypeProperty elements of the OWL class "Destination":

```
<owl:Class rdf:about="# Destination" />
<owl:DatatypeProperty rdf:about="#DestinationID">
   <rdfs:domain rdf:resource="#Destination"/>
   <rdfs:Datatype rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#Name">
   <rdfs:domain rdf:resource="#Destination"/>
   <rdfs:Datatype rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
</owl:Class>
```

Are translated as shown in Figure 2.

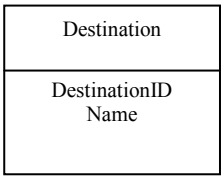


Figure 2. UML Class with properties.

Rule c_3: taxonomic relation H^C that expresses (IS-A relation) becomes CM Generalization/Specialization relationship. OWLsubClassOf element is transformed into Generalization/ Specialization relationship in UML.

Example: according to rule c_3, the OWL hierarchical relationships:

```
<owl:class rdf:about="UrbanArea">
   <owl:subClassOf rdf:resource="#Destination"/>
</owl:class>
<owl:class rdf:about="RuralArea">
   <owl:subClassOf rdf:resource="#Destination"/>
</owl:class>
```

Are translated as shown in Figure 3.

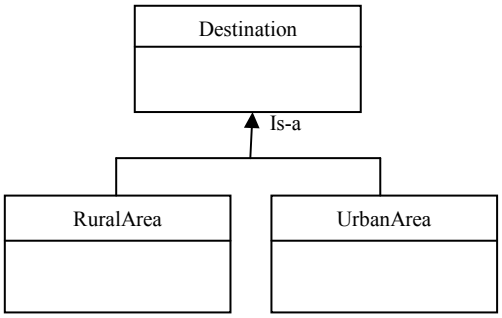


Figure 3. UML Generalization/ specialization relationship.

Rule c_4: no-taxonomic relation *R* is translated as follow:

- Relation having concept as domain and range becomes a CM Association relationship. The association has the domain as the source CM class and the range as the target CM Class. The relation local name becomes the target class name.
- Relation that expresses part whole relation becomes CM aggregation/ composition relationship.

OWLObjectProperty element is transformed into Relationship element in UML. An object property in OWL relates an individual to other individuals. An object property is defined as an instance of the built-in OWL class owl: ObjectProperty. Relationships element in UML represents connections, links, or associations between two or more entities.

Example: according to rule c_4, the OWLObjectProperty element "HasAccommodation":

```
<owl:ObjectProperty rdf:about="#HasAccommodation">
   <rdfs:domain rdf:resource="#Destination"/>
   <rdfs:range  rdf:resource="#Accommodation"/>
</owl:ObjectProperty>
```

Is translated as shown in Figure 4.

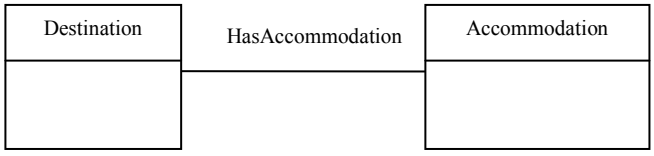


Figure 4. Binary UML association.

Rule c_5: ontological axioms X are translated into CM constraints.

Example: Table 1 describes some correspondences between OWL axioms and UML constraints.

Table 1. Correspondences between OWL axioms and UML constraints.

OWL Axioms	UML Constraints
AllValuesFrom	Specializes
SomeValuesFrom	Refines
Oneof	Enumeration
DisjointwWith	Disjoint
UnionOf	Cover
Cardinalities	Multiplicities

3. Overall Architecture and Implementation Issues

The rules presented above are general and can be adapted to most conceptual modelling languages. The conceptual modelling language we have used is the UML, but we believe that our results could be applied to any similar language. On the other hand, the method is fully automatic. To evaluate our ontology-based web application reverse engineering approach, a prototype has been implemented.

3.1. Architecture of the Implemented Tool

We have implemented the proposed approach in Java. The developed tool interacts with the Java WordNet and Jena APIs to parse HTML pages, compute semantic distance, and generate conceptual schema. It provides a set of features for personalizing the operations and calculations performed during the reverse engineering process. Details on the obtained conceptual schema can also be visualized. The overall Framework is shown in Figure 5. The system architecture involves the following components:

- User interface: allows the acquisition of HTML pages of web application, as well as domain ontology. It allows user to fix semantic distance threshold and to choose method and strategy for calculating this distance. In addition, it allows viewing the resulting conceptual schema, and a final report detailing all calculations and operations performed throughout the reverse engineering process.
- Extractor engine: represents an implementation of the extraction phase in the reverse engineering process. It is used to extract useful information from the acquired HTML pages.
- Identifier engine: represents an implementation of the identification and enrichment phases in the reverse engineering process. It covers calculation of semantic distance, identification of a set of ontological concepts and relations, and enrichment of the identified set.
- Conceptualisation engine: is responsible for reverse engineering domain ontology to corresponding conceptual data model, it has been implemented according to the rules mentioned above, and executed using the identified set of ontological constructs.

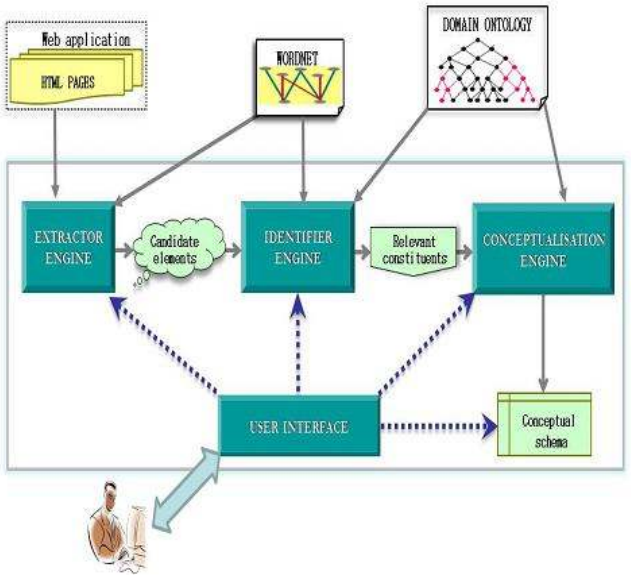


Figure 5. The overall framework.

3.2. Tool Evaluation

To evaluate the quality of the match calculation, we compare the match result returned by the automatic matching Process (P) with manually determined match Result (R). We determine the true positives, i.e., correctly Identified (I) matches.

Based on the cardinalities of these sets, the following quality measures are computed:
Precision = $|I|/|P|$, is the faction of the automatic discovered mapping which is correct. It estimates the reliability for the match prediction. Recall = $|I|/|R|$, is the fraction of the correct matches (the set R) which has been discovered by the mapping process. It specifies the share of real match that is found.

Precision and recall have been used extensively to evaluate the retrieval performance of retrieval algorithm in the information retrieval field [2]. A single measure, the F-measure [17], is also computed in order to compare the overall results of the four strategies. This measures a combination of precision and recall and is defined as follows:

$$F - mesure = \frac{(2 \times \text{Re } call \times \text{Pr } ecision)}{(\text{Re } call + \text{Pr } ecision)} \quad (4)$$

To illustrate our experiments, we have chosen the web site of USA tourism travel guide [15] as shown in Figure 6, and the tutorial ontology for a semantic web of tourism [26] as running example.

Figure 6. Web site of USA tourism travel guide.

After conducting several tests, the following remarks were noticed:

- For the path measure, more the semantic distance threshold is small, more the conceptual schema becomes complex, less is the matching result and vice versa.
- The multidimensional scaling strategy gives more efficient results than the other strategies.
- The recursive application of enrichment rules will provide more complete conceptual schema. Nevertheless, a significant number of iteration can generate superfluous concepts and relations. Thus, is to the designer to define the necessary and sufficient number of enrichment rules iteration.

Fixing a threshold for semantic distance means that an error rate was tolerated. Modelling a domain of interest is not deterministic, but it is heuristic. There is not a unique correct model for a situation, but only adequate or inadequate models [14]. Using path measure as similarity measure between element names, multidimensional scaling strategy as semantic distance between element attributes, and a threshold as 0.7, we generate the conceptual schema presented in Figure 7.

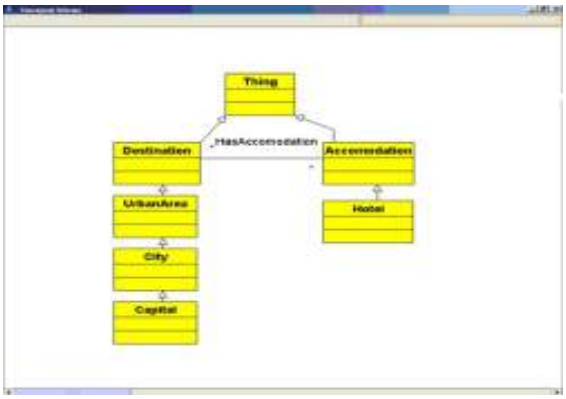


Figure 7. Conceptual schema with threshold as 0.7.

While changing the threshold to 0.3, we obtain more complex conceptual schema as shown in Figure 8.

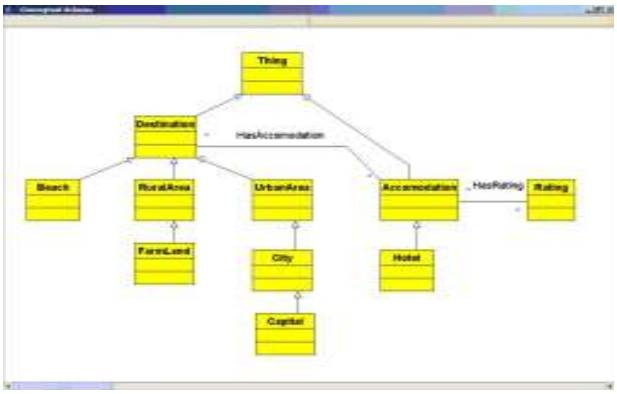


Figure 8. Conceptual schema with threshold as 0.3.

For space purposes we will only present the results graphically as shown in Figures 9 and 10.

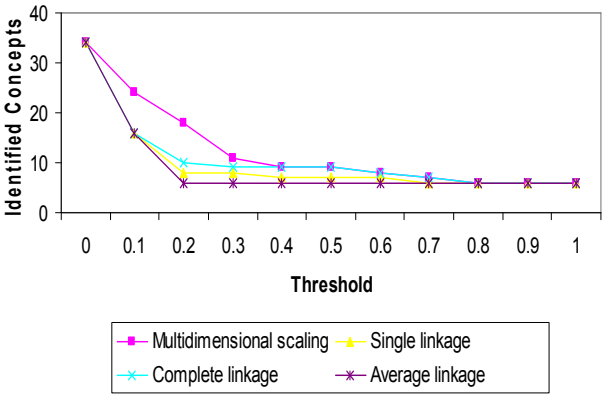


Figure 9. Identified concepts using path measure.

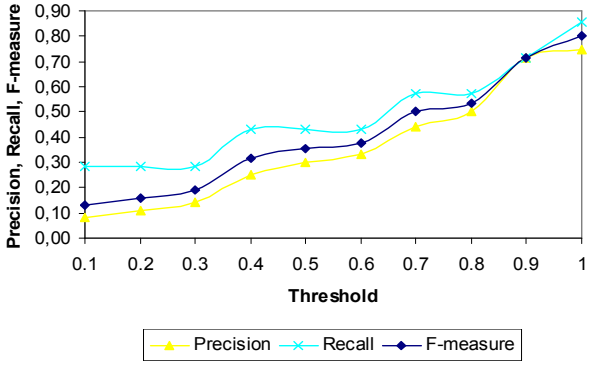


Figure 10. Turn precision, recall, F-measure curves using multidimensional scaling strategy.

4. Discussion

Similarities do exist between conceptual data models and ontological models with respect to abstracting and modelling the universe of discourse. But, their purposes are different. In general, the incompatibilities between them can be summarized as follows:

- Motivation: one of the main goals of ontology is to standardize the semantics of existing concepts within a certain domain. While an ontology represents knowledge that formally specifies a shared/agreed understanding of the domain of interest, conceptual models describe the structural

requirements for the storage, retrieval, organization, and processing of data in information systems such that the integrity of the data is preserved.

- Usage: ontology plays a significant role at run-time to browse ontology concepts to form semantically correct queries, and perform some advanced reasoning tasks [14]. So, ontology is sharable and exchangeable at run-time, while conceptual data models are off-line model diagrams [16] and their queries are usually to retrieve a collection of instance data [22].
- Evolution: generally an ontology (based on description logics) is a logical and dynamic model that can deduce new knowledge relations from the stored ones, or check for its consistency. However, conceptual models are static and are explicitly specified at design, but their semantic implications might be lost at implementation-time.
- Model Elements: in ontology, elements can be expressed either by their names or as Boolean expression in addition to using axioms such as cardinality/type restrictions, or domain/range constraints for classes or properties. On the other hand, conceptual model are concerned with the structure of data in terms of entities, relationships and a set of integrity constraints. For example primary key and functional dependences play very important roles within databases, but this is not always the case in the ontology since it concentrates more on how the concepts are semantically interrelated.

5. Conclusion

In this paper we have proposed a reverse engineering approach of semi-structured and undocumented web application. The proposed approach provides a reverse engineering rules to derive from domain ontology a conceptual schema modelling a web application. The reverse engineering process consists in four phases: extracting useful information; identifying a set of ontological constructs representing the concepts of interest; enriching the identified set by additional constructs; and finally deriving a conceptual schema.
A prototype has been developed to implement the proposed approach. Some validation experiments have been carried out and they showed the usefulness of the proposed approach and highlighted possible areas for improvement of its effectiveness. We have formalized the method independently of the conceptual modelling language used. However, the method can be adapted to most languages. On the other hand, our method can be used with any domain ontology.

The strong point of our approach is that it relies on a very rich semantic reference that is domain ontology. However, it is not possible to transform all elements from domain ontology into conceptual data model straight forward because ontology is semantically richer when data conceptual model.

The developed approach provides very satisfactory and encouraging results and supports the potential role that this approach can play in providing a suitable starting point for conceptual data model development. Nevertheless the derived conceptual schema should undergo a validation process that needs to be performed by domain specialist. Moreover, by using WorldNet we can analyze only English web applications. This problem can be solved in future work by using multilingual lexical knowledge.

References

[1] Baclawski K., Kokar M., Kogut P., Hart L., Smith J., Holmes W., Letkowski J., and Aronson M., "Extending UML to Support Ontology Engineering for the Semantic Web," *in Proceedings of 4th International Conference on UML*, Toronto, pp. 342-360, 2001.

[2] Baeze R. and Ribeiro B., *Modern Information Retrieval*, Addison Wesley, 1999.

[3] Bellettini C., Marchetto A., and Trentini A., "WebUml: Reverse Engineering of Web Applications," *in Proceedings of 19th ACM Symposium on Applied Computing*, Cyprus, pp. 1662-1669, 2004.

[4] Benslimane M., Malki M., Rahmouni M., and Rahmoun A., "Towards Ontology Extraction from Data Intensive Web Sites: An HTML Forms Based Reverse Engineering Approach," *The International Arab Journal of Information Technology*, vol. 5, no. 1, pp. 34-44, 2008.

[5] Conesa J., Palol X., and Olive A., "Building Conceptual Schemas by Refining General Ontologies," *in Proceedings of 14th International Conference on Database and Expert Systems Applications DEXA*, Czech Republic, pp. 693-702, 2003.

[6] Di G., Lucca A., Fasolino F., Pace P., Tramontana U., and Carlini D., "WARE: A Tool for the Reverse Engineering of Web Applications," *in Proceedings of the European Conference on Software Maintenance and Reengineering*, Budapest, pp. 241-250, 2002.

[7] Do H. and Rahm E., "Coma: A System for Flexible Combination of Schema Matching Approaches," *in Proceedings of 28th International Conference on Very Large Data Bases*, China, pp. 610-621, 2002.

[8] El-Ghalayini H., Odeh M., and McClatchey R., "Deriving Conceptual Data Models from Domain Ontologies for Bioinformatics," *in Proceedings of 2nd International Conference on Information and Communication Technologies from Theory to Application (ICTTA)*, Syria, pp. 22-26, 2006.

[9] Embley D., "Toward Semantic Understanding: An Approach Based on Information Extraction Ontologies," *in Proceedings of the 15th Australasian Database Conference*, Germany, pp. 3-12, 2004.

[10] Estiévenart F., François A., Henrard J., and Hainaut L., "A Tool Supported Method to Extract Data and Schema from Web Sites," *in Proceedings of the 5th International Workshop on Web Site Evolution*, Amsterdam, pp. 3-11, 2003.

[11] Fellbaum C., *WordNet: An Electronic Lexical Database*, MIT Press, 1998.

[12] Gaeremynck Y., Bergman L., and Lau A., "MORE for Less: Model Recovery from Visual Interfaces for Multi Device Application Design," *in Proceedings of the International Conference on Intelligent User Interfaces*, USA, pp. 69-76, 2003.

[13] Gibson D. and Conheeney K., "Domain Knowledge Reuse During Requirements Engineering," *in Proceedings of CAiSE*, New York, pp. 283-296, 1995.

[14] Guarino N., "Formal Ontology and Information Systems," *in Proceedings of the International Conference on Formal Ontology in Information Systems*, Italy, pp. 3-15, 1998.

[15] HM USA Travel Guide, http://www.hm-usa.com, 2006.

[16] Jarrar M., Demy J., and Meersman R., "On Using Conceptual Data Modeling for Ontology Engineering," *Computer Journal on Data Semantics*, *LNCS 2800,* vol. 3, no. 1, pp. 185-207, 2003.

[17] Larsen B. and Aone C., "Fast and Effective Text Mining Using Linear Time Document Clustering," *in Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, USA, pp. 16-22, 1999.

[18] Levenshtein I., "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *in Proceedings of Cybernetics and Control Theory*, USA, pp. 707-710, 1966.

[19] Lopez J. and Szekely A., "Web Page Adaptation for Universal Access," *in Proceedings of the 1st International Conference on Universal Access in Human Computer Interaction*, New Orleans, USA, pp. 690-694, 2001.

[20] Maedche A., *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers, Boston, 2002.

[21] Marc E., Peter H., Mark H., and Nenad S., "Similarity for Ontologies: A Comprehensive Framework," *in Proceedings of the 13th European Conference on Information Systems (ECIS 2005)*, Germany, pp. 1-60, 2005.

[22] Noy N. and Klein M., "Ontology Evolution: Not the Same as Schema Evolution," *Computer Journal of Knowledge and Information Systems*, vol. 6, no. 4, pp. 428-440, 2004.

[23] Object Management Group, "Unified Modelling Language," http://www.omg.org/uml/, 2002.

[24] Paganelli L. and Paterno F., "Automatic Reconstruction of the Underlying Interaction Design of Web Applications," *in Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, Italy, pp. 439-445, 2002.

[25] Peterson B., Andersen A., and Engel J., "Knowledge Bus: Generating Application Focused Databases from Large Ontologies," *in Proceedings of the 5th International Workshop on Knowledge Representation Meets Databases*, pp. 1-10, 1998.

[26] Protege, http:/ /protege. stanford. edu/ plugins /owl/owl-library/travel.owl, 2009.

[27] Ricca F. and Tonella P., "Using Clustering to Support the Migration from Static to Dynamic Web Pages," *in Proceedings of the 11th International Workshop on Program Comprehension*, USA, pp. 207-216, 2003.

[28] Rumbaugh J. and Blaha M., *Modélisation et Conception Orientées Objet,* Edition Française Revue et Augmentée, Paris, 1994.

[29] Smith K., Welty C., and McGuinness D., W3C Recommendation, http://www.w3.org/TR/owl-guide, 2004.

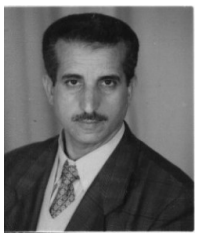
[30] Stroulia E., Thomson J., and Situ Q., "Constructing XML Speaking Wrappers for Web Applications: Towards an Interoperating Web," *in Proceedings of the 7th Working Conference on Reverse Engineering*, Australia, pp. 59-68, 2000.

[31] Swartout B., Patil R., Knight K., and Russ T., "Toward Distributed Use of Large Scale Ontologies," *in Proceedings of the 10th Knowledge Acquisition for Knowledge Based Systems Workshop*, Canada, pp. 103-126, 1996.

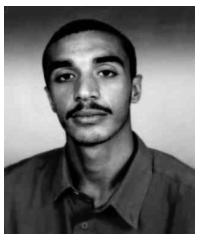
[32] Tijerino A., Embely W., Lonsdale W., Ding Y., and Nagy G., *Towards Ontology Generation from Tables*, Kluwer Academic Publishers, 2005.

[33] Vanderdonckt J., Bouillon D., and Souchon N., "Flexible Reverse Engineering of Web Pages with Vaquista," *in Proceedings of the 8th Working Conference on Reverse Engineering*, Germany, pp. 241-248, 2001.

Sidi Benslimane received his PhD degree in computer science from Sidi Bel Abbes University, Algeria, in 2007. His research interests include, semantic web, web engineering, ontology engineering, information and knowledge management, and process modelling.



Mimoun Malki is a Master of lectures at the Department of Computer Science at Sidi Bel Abbes University. He received the PhD degree in computer science from Sidi Bel Abbes University, Algeria, in 2003. He heads the evolutionary engineering and distributed information systems laboratory.



Djelloul Bouchiha received his MSc degree in computer science from the University of Sidi Bel Abbes in 2003. His PhD candidature in the Department of Computer Science, School of Computing at the University of Sidi Bel Abbes. His researches interests include schema integration, data integration, data dependency, and conceptual modelling.