ER2OWL: Generating OWL Ontology from ER Diagram

Muhammad Fahad

Mohammad Ali Jinnah University, Islamabad, Pakistan
mhd.fahad@gmail.com

Abstract. Ontology is the fundamental part of Semantic Web. The goal of W3C is to bring the web into (its full potential) a semantic web with reusing previous systems and artifacts. Most legacy systems have been documented in structural analysis and structured design (SASD), especially in simple or Extended ER Diagram (ERD). Such systems need up-gradation to become the part of semantic web. In this paper, we present ERD to OWL-DL ontology transformation rules at concrete level. These rules facilitate an easy and understandable transformation from ERD to OWL. The set of rules for transformation is tested on a structured analysis and design example. The framework provides OWL ontology for semantic web fundamental. This framework helps software engineers in upgrading the structured analysis and design artifact ERD, to components of semantic web. Moreover our transformation tool, ER2OWL, reduces the cost and time for building OWL ontologies with the reuse of existing entity relationship models.

1 Introduction

Ontology is regarded as the formal specification of the knowledge of concepts and the relationships among them [7]. They require formal syntax and semantics to represent domain concepts. They have played a key role for describing semantics of data not only the applications of semantic web but also revolutionized the traditional knowledge engineering [14]. There are many languages proposed to build ontologies e.g. RDFS, OWL, LOOM, OIL etc. In 2004, W3C has made OWL as a standard to build ontologies [9], because of its decidability and high level of expressivity. OWL describes many types of semantics about terms to facilitate high mechanisms for reasoning; like its hierarchal information, its relation with others and its own description in the form that are machine-interpretable and machine-understandable.

However, many legacy systems have been documented using Structured Analysis and Structured Design (SASD) [1]. The most common artifact of SASD is the

Please use the following format when citing this chapter:

Fahad, M., 2008, in IFIP International Federation for Information Processing, Volume 288; *Intelligent Information Processing IV*; Zhongzhi Shi, E. Mercier-Laurent, D. Leake; (Boston: Springer), pp. 28–37.

Entity Relationship (ER) Diagram or Extended ER Diagram. There are many problems in understanding and upgrading a legacy system. To use old system as a component of emerging semantic web, these systems need up-gradation. Software engineers require OWL ontology to aid in the up-gradation of these systems. The new OWL ontology can act as the basis for design and implementation of the new system with in the semantic web. Both ERD and OWL represent entities and their relationships. This provides an intuitive transformation of the ERD to OWL ontology. Translating ERD into OWL is indeed seems like a worthy goal of current model driven architecture, in order to integrate legacy systems into the Semantic Web. Extended ER has a bigger domain then ER, so from now onward we refer ERD formed from Extended ER notations. There are so many conventions and notations used for conceptual modeling of ER, here we are using the notations of Chen [10].

As ontologies have been used by couple of industry applications, our transformation tool will open up a number of advantages. Especially ontologies played an important role for heterogeneous database integration [4]. Heterogeneous databases are represented by the ontologies and can combine together efficiently. As semantic web is emerging, schema integration becomes the main hindrance in achieving its goal. Here we have developed concrete rules that aid in schema integration. We demonstrate an example to facilitate usage of these rules. They also help in mapping between relational database schema and OWL ontologies for deep annotation. "*Deep Annotation means the process of creating ontological instances for the database-based, dynamic contents by reaching out to the 'deep Web' and directly annotating the underlying database of the dynamic Web site*" [3].

Ontology integration is one of the active research areas of present era, as different models are used to build up different domain ontologies such as RDBMS, XML, etc. This framework helps the ontologists to resolve model conflict [7] in ontology integration. Those ontologies that are build-up on RDBMS model are transformed into OWL and then different OWL ontologies are combined together through conventional methods.

Rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the framework for transforming ERD to OWL. Section 4 concludes the paper and shows future directions.

2 Related Work

Vasilecas et al. proposed an approach to automatically transform enterprise ontology into conceptual model [6]. They used metamodels of ontology and conceptual model to facilitate transformations from one domain to another. A prototype was developed to show the effectiveness and automation of their proposed technique. They build ontology in Protégé 2000 and Power Designer was used to im-

plement ER model. They transform class of ontology in protégé to entity in ER, and slot (property) to attribute, directed binary relation to inheritance and ontology constraints to entity constraints.

Xu et al. proposed mapping rules between relational database schema and OWL ontology for deep annotation [3]. Ontological annotation is used for dynamic web page contents extracted from the database. Their Framework, DPAnnotator, translate the ER Schema of the relational database into OWL ontology. They provide a D2OMapper tool, which automatically creates the mappings by following their rules.

Colomb et al. discussed the issues in mapping metamodels in the ontology development metamodel using QVT [2]. They suggest many ways to integrate several metamodels in one structure. First approach is to take one metamodel as basic, and represent others by subclasses. Second approach suggests taking one metamodel as basic and translating others into it. Third approach suggests representing the metamodels separate and providing transformation rules from one domain to another using QVT. They gave QVT transformations between UML and DL (Description Logic), ER and DL, and OWL to DL. QVT does not only help in transformation process but also keeps track of the association between source and target model elements.

Kupfer et al. proposed an approach that allows the database schema and the ontology to change and evolve, without breaking their connection with each other during maintenance [4]. They gave the automatic mappings from database schemas to database ontologies, with maintaining connection with each other when one of the artifact changes. They called this process, the Coevolution process that tracks changes of the database schema into related database ontology.

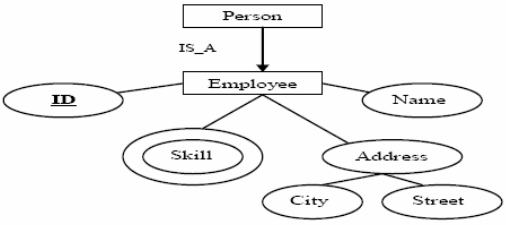
In Ontology Definition Metamodel, researchers provided ER to OWL mappings [5]. They used abstract syntax for representation of mapping specification. Understanding their mapping specification is much tedious task. A new researcher needs much effort to transform ERD to OWL ontology, as they did not provide any explanation, case study or tool support. Furthermore their proposal provides a starting point, but never provides formal rules that could be used to automate the transformation. It also needs further elaboration and examples to help aid in transforming ERD to OWL. While transforming conventional ERD to OWL, we find many inconvenience especially transforming association entity and unary, binary and ternary relationships between entities, as they did not incorporate association entity in their metamodel and differentiate between unary, binary and ternary relationship. Besides these, they did not included ER metamodel and mapping rules in their latest document and thus did not elaborate them further. Thus we feel a need to extend their work and propose a framework, which facilitates an easy and thorough transformation.

A more related work of our domain is done by Upadhyaya et al. He presented the implementation details of their tool, ERONTO, that extracts ontologies from Extended ERD [8]. His proposal provides a good starting point but lack some features, as the tool does not produce complete mappings from ERD to OWL, and

user himself has to check the incompleteness and write glue code himself. They call the generated OWL as a "*near-complete ontology*" and conclude that users track the portions that are missing and enhance the generated ontology themselves. They gave an inappropriate mapping of composite attribute to OWL Class, which increases the overhead to produce ObjectProperties that relate OWL Class (corresponding to Entity) to another OWL Class (corresponding to composite attribute), and cardinalities restrictions associated with both the classes. Furthermore they did not tell how the Restriction code is generated that is equivalent to cardinality in conjunction with modalities i.e. cardinality of N represents 1..* or 0..* for mandatory and optional modalities respectively. Ignoring restrictions in the code of generated OWL Class equivalent to associative entity makes the code inconsistent. As associative entity does not exist without the existence of corresponding entities, and the only way to apply this constraint is to produce restrictions in that class. Moreover they did not handle multivalued attributes, unary relationships 1:N or N:M etc. Another serious incompleteness error [11,12,13] that ERONTO produces is the functional property omission error for single valued property (attribute). This type of error creates inconsistencies by allowing attribute to accept many values and create ambiguity.

3. Transformation Framework

SASD uses the Entity Relationship Diagram (ERD) to model data. Both ERD and OWL represent entities and their relationships and this gives an intuitive transformation of the ERD to OWL ontology: ERD entities become OWL classes; ERD attributes become Datatype Properties of corresponding OWL class.




```
<owl:Class rdf:ID="Person"/>
 <owl:Class rdf:ID="Employee">
  <rdfs:subClassOf rdf:resource="#Person"/>
 </owl:Class>
 <owl:InverseFunctionalProperty rdf:ID="ID">
  <rdf:type rdf:resource=" #FunctionalProperty"/>
  <rdfs:range rdf:resource=" #int"/>
  <rdfs:domain rdf:resource="#Employee"/>
  <rdf:type rdf:resource="#DatatypeProperty"/>
 </owl:InverseFunctionalProperty>
```

```
<owl:DatatypeProperty rdf:ID="Skill">
  <rdfs:range rdf:resource="#string"/>
  <rdfs:domain rdf:resource="#Employee"/>
 </owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID="Name">
  <rdfs:range rdf:resource="#string"/>
  <rdfs:domain rdf:resource="#Employee"/>
  <rdf:type rdf:resource="#DatatypeProperty"/>
 </owl:FunctionalProperty>
```

Fig. 1. ER Entity to OWL Class mapping.

Bidirectional Relationships become two Object Properties. Cardinalities on the ERD can be used in conjunction with modalities to produce restrictions for the OWL class.

3.1 ERD to OWL Mapping Rules

The set of rules to transform an ERD into OWL ontology are outlined below:

Entity. Map each Entity in the ERD into OWL class in the OWL Ontology. In Fig. 1, Person and Employee are entities that are mapped to OWL classes.

Attribute. There are many types of attributes that belong to entity i.e. simple attributes, composite attributes and multi-valued attributes. These require separate ways to map into OWL datatype properties. While transforming attributes to properties, it should be taken care of making local unique names. Incase of two unique names associated with entities, our system appends the entity name with start of attribute like (*Entity: Attribute*). Moreover, Range mapping of values has to take care of mapping the datatypes of the ER domain to *XSD* datatypes.

Simple Attribute. Map Simple Attribute of entity into datatype property of corresponding OWL class. Domain of the datatype property is the Entity, and range is the actual datatype (int, string, etc) of that attribute. Range mapping of values has to take care of mapping the datatypes of the ER domain to XSD datatypes. One important point should be considered here is that as this attribute takes only one value so a special tag "functional" should be tagged with this datatype property, otherwise OWL DL allows datatype property to take many values by default. In Fig. 1, Name of Employee is a simple attribute.

Composite Attribute. There are two ways to map Composite Attribute to OWL datatype property. One is to map only their simple component attributes (city, street, country, etc) of composite attribute (Address) to datatype properties

```
<owl:FunctionalProperty rdf:ID="City">
  <rdfs:range rdf:resource="#string"/>
  <rdf:type rdf:resource="#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Employee"/>
 </owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="Street">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdf:type rdf:resource=" #DatatypeProperty"/>
  <rdfs:range rdf:resource=" #string"/>
 </owl:FunctionalProperty>
```

```
<owl:DatatypeProperty rdf:ID="Address">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdfs:range rdf:resource="#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="street">
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#Address"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="city">
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#Address"/>
</owl:DatatypeProperty>
```

Fig. 2. Composite Attribute to Datatype Property Mapping

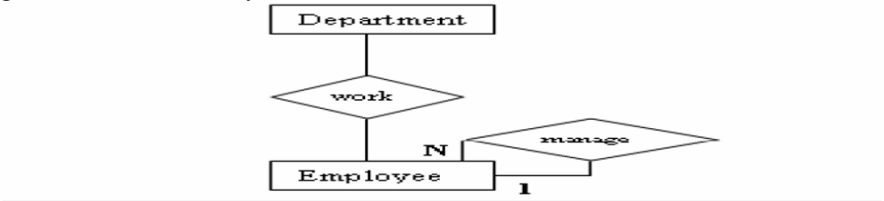
of corresponding OWL class, and ignore composite attribute (Address) itself. Second is to map composite attribute to datatype property and then map its simple, component attributes to *subproperty* of corresponding datatype property. The first

one is more preferred while working with transformation of relational databases because in Relational Schema, we do have only the instances of simple, component attributes and composite attributes are ignored. By using this rule one has not preserved the conceptual modeling of composite attribute and when performing reverse engineering from ontology to ER, we lost composite attributes. If someone wants to preserve such knowledge then second one is used effectively by analyzing datatype property to composite attribute and subproperty-of to its component attribute. All the datatype properties produce should be tagged as "functional" as they all get only one value. Fig. 2 shows both the method of transforming Composite Attribute to OWL ontology.

Multi-valued Attribute. Multi-valued Attribute is mapped to datatype property like simple attribute, but without a "*functional*" tag. For an example, Skill of an employee may have many values, so OWL DL property by default takes *many* values.

Primary Key. An attribute which stands as a primary key, is transformed into datatype property and is tagged with both "*functional*" and "*inverse-functional*". Functional tag restricts object to take only one value for a given subject, and inverse-functional restricts the subject to associate with only one object [9].

Subtype Relations (IS-A). Convert subtype relations in the ERD to subClassOf in the OWL ontology. In OWL ontology, *OWL:subClassOf* represents the generalization hierarchy.




```
<owl:ObjectProperty rdf:ID="work">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdfs:range rdf:resource="#Department"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="has_work">
  <owl:inverseOf rdf:resource="#work"/>
</owl:ObjectProperty>
```

```
<owl:Class rdf:ID="Employee">
   <owl:Restriction><owl:someValuesFrom>
      <owl:Class rdf:about="#Employee "/>
     </owl:someValuesFrom>
     <owl:onProperty>
       <owl:ObjectProperty rdf:resourse="#manage"/>
     </owl:onProperty></owl:Restriction>
  </owl:Class>
<owl:ObjectProperty rdf:ID="manage">
   <rdfs:domain rdf:resource="#Employee"/>
   <rdfs:range rdf:resource="#Employee"/>
</owl:ObjectProperty>
```

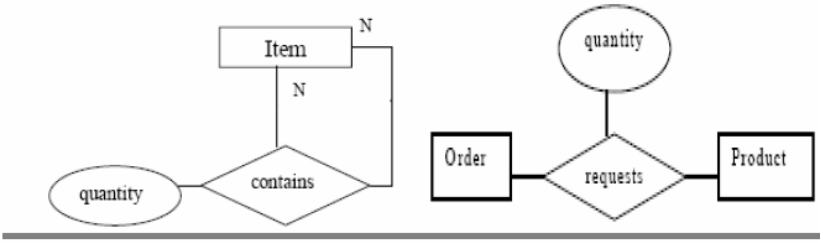
Fig. 3. Bi-Directional Relationship to Object Property Mapping

Bi-Directional Relationship. Every relationship between entities is mapped onto object property between classes. But in ERD it is bi-directional and object

property in OWL is uni-directional, so *two* object properties should be generated between those entities having bi-directioanl relationship. One corresponding to the relationship as represented in the ERD**,** and second as an inverse property of the original object property. For example if a relationship *Work* exists between Employee and Department as shown in Fig. 3, then in OWL two object properties are generated with names Work(domain:Employee and range:Department) and other Has_Work (by default it takes the opposite domain and range values of Work property) as the inverse property of *Work*. Note that the name of second property is generated by preceding the word '*has*' of the actual property and later on can be changed to give meaningful name to it.

Unary (recursive) 1:N Relationship. Unary recursive relation between a person Employee and other worker Employees can exists when one Employee handles/assists many Employees. In Fig. 3, Employee entity with *Manages* relationship is transformed into Employee OWL class and Manages as an Object property with *same* domain and range as employee.

Unary M:N Relationship. When unary relationship exists between M object to N objects then that it is transformed into two OWL classes. For example Many *Items* contain many item-components as shown in Fig. 4. In this case, we transform Item entity to Item OWL class and build another OWL *Contains* class that has some values from item class.




```
<owl:Class rdf:ID="Contains">
   <owl:Restriction><owl:someValuesFrom>
     <owl:Class rdf:ID="Item"/>
    </owl:someValuesFrom><owl:onProperty>
    <owl:ObjectProperty rdf:resourse="#has"/>
    </owl:onProperty></owl:Restriction>
 </owl:Class>
<owl:ObjectProperty rdf:ID="has">
  <rdfs:domain rdf:resource="#Contains"/>
  <rdfs:range rdf:resource="#Item"/>
 </owl:ObjectProperty>
```

```
<owl:Class rdf:ID="Requests">

<owl:DatatypeProperty rdf:ID="quantity">
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#Request"/>
 </owl:DatatypeProperty>
```

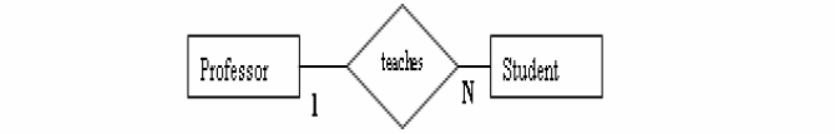
Fig. 4. Associative Entity and Unary Recursive Mappings

Associative Entity or Relationship having attribute. Associative Entity or Relationship having attribute is also mapped onto OWL class as shown in Fig. 4. Attribute of Relationship or Associative Entity is mapped into datatype property of corresponding OWL class.

1 to Many Relationship (mandatory). Cardinalities and Modalities are transformed into OWL restrictions within corresponding OWL classes. 1 to many relationship is transformed into restrictions as shown in Fig. 5.

1 to Many Relationship (optional). In case of optional, we do not put restrictions in the corresponding class.

Many to Many Relationship. This type of relationship in ERD is transformed into restrictions in OWL classes, and corresponding cardinalities and modalities are split up into two indicating Many to 1 and 1 to Many relationships, their implementation rules are applied accordingly to generate OWL code.




```
<owl:Class rdf:ID="Student">
  <rdfs:subClassOf><owl:Restriction>
    <owl:valuesFrom><owl:Class rdf:ID="Professor"/>
    </owl:valuesFrom><owl:onProperty>
      <owl:ObjectProperty rdf:ID="isTaughtBy"/>
    </owl:onProperty><owl:cardinality
rdf:datatype="#int"> 1</owl:cardinality>
    </owl:Restriction></rdfs:subClassOf>
 </owl:Class>

<owl:ObjectProperty rdf:about="#isTaughtBy">
   <owl:inverseOf rdf:resource="#teaches"/>
 </owl:ObjectProperty>
```

```
<owl:Class rdf:about="#Professor">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:someValuesFrom rdf:resource="#Student"/>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#teaches"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
<owl:ObjectProperty rdf:about="#teaches">
  <rdfs:domain rdf:resource="#Professor"/>
  <rdfs:range rdf:resource="#Student"/>
 </owl:ObjectProperty>
```

Fig. 5. 1 to Many Relationship Mappings

3.2 Comparison with ERONTO

We have implemented the above defined rules into a prototype, ER2OWL. It facilitates a quick transformation, and reduces the cost and time for building OWL ontologies with the reuse of existing entity relationship models. As comparison

with ERONTO, the generated ontology is more complete and does not have incompleteness errors.

The limitations of ERONTO are addressed by ER2OWL and produce ontologies that serve best when spread out to be used in real world applications. Unlike ERONTO, the system attach the "functional" tag with datatype properties with single valued property, moreover does not create overheads by transforming composite attributes to owl classes and let the application safe from incompleteness and consistency errors [11,12,13]. (Details about these errors are found in individual paper and are out of scope of this paper). Our system asks the user to suggest the direction of relationship between entities, so that valid domain and range of object properties should be generated against each object property.

4 Conclusions and Future Work

This paper presents a framework for transforming the structured analysis and design artifact, ERD, into the OWL ontology. We have provided rules to transform ERD concepts into equivalent OWL ontology for semantic web. The set of mapping rules has been demonstrated with the diagrams to promote understanding. The framework provides OWL ontology for semantic web component from old legacy systems and enable them to upgrade and become a part of emerging semantic web. This proposed framework helps software engineers in upgrading the structured analysis and design artifact ERD, to components of semantic web. Our ongoing research on this topic is to handle other cases of relationships that are not binary, which require reification.

Reference

[1] T. P. Fries. A Framework for Transforming Structured Analysis and Design Artifacts to UML. SIGDOC'06, Myrtle Beach, South Carolina, USA, October 18-20, 2006,

[2] R. M. Colomb, A. Gerber, M. Lawley. Issues in Mapping Metamodels in the Ontology Development Metamodel Using QVT.

[3] Z. Xu, S. Zhang, Y. Dong. Mapping between Relational Database Schema and OWL ontology for Deep Annotation. Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06), 2006 IEEE

[4] A. Kupfer, S. Eckstein, K. Neumann and B. Mathiak. A Coevolution Approach for Database Schemas and related Ontologies. Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06), 2006 IEEE

[5] Ontology Definition Metamodel, second Revised Submission to OMG/RDF ad/2006-04-13

[6] O. Vasilecas, D. Bugaite, J. Trinkunas. On Approach for Enterprise Ontology Transformation into Conceptual Model. International Conference on Computer Systems and Technologies, CompSysTech'06

[7] D. Dou, P. LePendu. Ontology based Integration for Relational Databases. *SAC'06,* April 2327, 2006, Dijon, France.

[8] S. R. Upadhyaya and P. S. Kumar. ERONTO: A Tool for Extracting Ontologies from Extended E/R Diagrams, ACM Symposium on Applied Computing 2005.
[9] OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/
[10] P. Chen. The Entity Relationship model towards a unified view of data, ACM Transactions Database Systems., I,1(March 1976),9-36.
[11]. M.A. Qadir, M. Fahad, S.A. Hussain-Shah, Incompleteness Errors in Ontologies. InProc. of International Conference on Granular Computing, Silicon Valley, USA, IEEE Computer Society. pp 279-282
[12]. W. Noshairwan, M.A. Qadir, M. Fahad. Sufficient Knowledge Omission error and Redundant Disjoint Relation in Ontology. InProc. 5th Atlantic Web Intelligence Conference, Fontainebleau, France (June 25-27, 2007)
[13]. M. Fahad, M.A. Qadir, W. Noshairwan. Semantic Inconsistency Errors in Ontologies. In Proc. of InternaQtional Conference on Granular Computing, Silicon Valley, USA, IEEE Computer Society. pp 283-286
[14]. G. Antoniou, and F.V. Harmelen, A Semantic Web Primer. MIT Press Cambridge, ISBN 0-262-01210-3, 2004.