Performing Entity Relationship Model Extraction from Data and Schema Information as a Basis for Data Integration

Philipp Schmurr[a], Andreas Schmidt[b], Karl-Uwe Stucky[c], Wolfgang Suess[d] and Veit Hagenmeyer[e]

Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Kaiserstr. 12, Karlsruhe, Germany
{*philipp.schmurr, andreas.schmidt, karl-uwe.stucky, wolfgang.suess, veit.hagenmeyer*}*@kit.edu*

Keywords: Entity Relationship Model, Model Extraction, Data Integration, Structural Metadata, FAIR Principles.

Abstract: The goal of this work is to allow domain experts to properly perform data integration themselves and not to rely on external resources. This way the long-term data integration quality is not endangered and therefore cost for external resources can be saved. To achieve this, we propose a new approach that enables data integration based on entity-relationship (ER) models derived from arbitrary data sources. ER models are abstract and simply define all entities and relations needed for integration, which makes them easy to understand. Strategies to extract ER models from various standard data sources - relational databases, XML files and OWL data - are presented and a concept on how to extend it to arbitrary other data sources is introduced. Furthermore, the extracted models are a foundation to perform graphical data integration into an ontology based model and, thus, contribute to a harmonized knowledge management in heterogeneous data and information environments. It can be summarized as a strategy to improve the interoperability of existing data according to the FAIR principles.

1 INTRODUCTION

Whenever a new use case or project is started in industry, it usually requires gathering and integrating data from various sources to be able to achieve the intended goal. This can include internal or external data that is probably stored in multiple different information systems, file formats, and locations.

Especially with limited budget and staff members, data integration is performed manually to collect and transform the data as needed for the new project. In more advanced projects, data integration is done using available solutions that provide an automated and repeatable way for data integration. Data integration solutions often require internal data modeling experts or an outsourcing to external contractors. For cost reasons, the solutions are often simplified as much as possible, so that they hardly comply to the state of the art.

The energy domain is particularly affected by insufficient data integration. The energy transition requires more and more data-driven solutions, but especially smaller power companies do not have the necessary data experts yet. Moreover, software - similar to hardware - is mainly delivered as turnkey solutions with proprietary data models, formats and interfaces. In this context, the vendor lock-in is a considerable disadvantage, e.g. because the available interfaces to communicate with other software are hard to understand. Another challenge is that the energy system is a critical infrastructure and therefore all kinds of cloud solutions are not feasible. On the other hand, many software systems in the energy domain have originally been designed by electrical engineers as tools to support their daily work. Those tools have then evolved into software products that still have many legacy issues from a bygone era. As a result of the above reasons, the energy domain has many custom and legacy data sources and tools that are harder to integrate compared to other domains. The statements made above are based on our previous working experience in the energy industry.

Due to the mentioned conditions, we perceive it is a key characteristic of a data integration solution to be usable by energy domain experts without external assistance. As a solution we introduce FAIRlead,

[a] https://orcid.org/0009-0004-2324-7839
[b] https://orcid.org/0000-0002-9911-5881
[c] https://orcid.org/0000-0002-0065-0762
[d] https://orcid.org/0000-0003-2785-7736
[e] https://orcid.org/0000-0002-3572-9083

Schmurr, P., Schmidt, A., Stucky, K., Suess, W. and Hagenmeyer, V.
Performing Entity Relationship Model Extraction from Data and Schema Information as a Basis for Data Integration.
DOI: 10.5220/0013012600003838

In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 3: KMIS*, pages 316-322
ISBN: 978-989-758-716-0; ISSN: 2184-3228

a concept for a data integration and management system for the energy domain. It will be open source and, therefore, especially tailored for smaller budgets. The relevant characteristics of FAIRlead are the following:

- Data integration is performed with a graphical user interface that is based on conceptual entity-relationship (ER) models (Chen, 1976) of the input data sources.
- The user interface shall be easy to understand and works the same for all kinds of data sources.
- The ER models are semi automatically extracted from data sources and allow the user to improve or correct the model if necessary.
- The integration target model is based on ontologies.
- We will employ code generation methods on the created target ontology to simplify the access to the mapped data.

So in this paper, we demonstrate the first step necessary for the approach depicted above: a strategy to extract ER models from data directly, thereby using structural metadata if available. It is important to state that the user will have the final control on improving or correcting the generated ER model. To demonstrate our concept, we present the extraction process for relational databases, XML files, OWL data sets and a proprietary text-based file format from the energy domain. For the demonstration we are using the Mondial database (May, 1999) which is available in the mentioned formats, as well as a scenario file (RAW) from the Siemens PSS®E power system simulator.

The present paper is organized as follows: in Section 2 we present related work on extracting models from metadata and data. Then, in Section 3, we present the data sources on which our experiments are based. The basic concepts of extracting the components of an ER model from different data sources are explained in Section 4. In Section 5, we then present initial results of the model extraction tools we have developed so far. In Section 6 we summarize our results and formulate an outlook for future research.

2 RELATED WORK FOR MODEL EXTRACTION

Extraction or reverse engineering of models has already been done on several types of data:

For spreadsheet data there exist several publications around the extraction of ClassSheet models (Cunha et al., 2010). Those models can be integrated into a spreadsheet file directly (Cunha et al., 2012) and can also be compared against relational schemas (Cunha et al., 2016).

The detection of what table headers are, as well as the recognition of relations between the tables is a key benefit of the proposed ClassSheet implementations. We have attempted to integrate those features from the available open source code, but struggled with the more than a decade old code base. Moreover, the implementation only focuses on Open Office spreadsheets which is a harsh limitation.

Relational databases also have been target of model extraction work before. Chiang et al. have presented an approach to extract an extended entity relationship (EER) model from a given database (Chiang et al., 1994b) and later also investigated the performance of their approach (Chiang et al., 1994a). Later Alalfi et al. have published a solution to extract the EER model and to export it to a UML diagram in the XML meta interchange format (XMI) (Alalfi et al., 2008).

The logic to extract the concepts of an ER model (entities, attributes, relations, cardinalities) proposed in those works, is reused in a similar form in our solution.

However, we have focused on a more lightweight semantic representation of the ER model compared to those papers. So for us, the diagram can be generated from our ER representation, but it is not the primary representation.

The next big group of model extraction papers targets the extraction of data from XML files as well as their existing schema representations (for example Document Type Definition (DTD)). One approach to extract a DTD from XML data is presented by Siau et al. (Siau et al., 2011). This approach even creates a new graph format they call Extended DTD Graph. Moreover, the approach employs techniques to find out relations between elements that are based on ID/IDREF(s) relationships and not just the hierarchical structure of the document. Klímek et al. created a survey of approaches to extract schema information from XML data also including further schema formats as XML schema (XSD), RelaxNG and schematron (Klímek and Nečaský, 2010). Finally, the extraction of ER models from DTDs is presented by both Yang (Yang et al., 2004) and Mello (Mello and Heuser, 2001). The first aims to improve existing DTDs since they are hard to read and understand. The ER representation is therefore used as a means to improve understandability. This follows the same basic idea as our approach to make data integration more approachable for domain experts. The approach from Mello employs a rule set to extract a canonical conceptual

model in an ontological representation and then also utilizes it for a semantic integration solution that focuses on XML data. The basic idea of conceptual model extraction for integration purposes is similar in our approach and we consider the idea to represent the conceptual model with an ontology for the future. We apply the concepts presented in these papers to extract the ER models from a DTD source. A newer approach by Della Penna et al. was proposed to extract an ER model also from XML schema (XSD) (Della Penna et al., 2006). Our implementation does follow a similar approach when working with a dataset that provides XSD information. In general, our solution can utilize existing software for either DTD or XSD extraction as a pre-processing step to utilize the benefits of an available DTD or XSD schema compared to directly extracting an ER model from XML data alone.

Lastly, the extraction of ER models from data represented in the Resource Description Framework (RDF) is an area of interest. However, RDF based data usually has an ontology (mostly in Web Ontology Language (OWL) format) as a model which is already an integration ready way of representing a data structure. The terms conceptual model (ER models are conceptual models) and ontology often appear together, both as a means to understand the domain before creating an ontology (Gomez-Perez et al., 2000) and also to use ontologies (e.g. the Unified Foundational Ontology (UFO)) to properly define the semantics of a conceptual model that is built for a new information system (Guizzardi, 2005). The latter approach is called ontology-based conceptual modelling.

To actually extract conceptual models from ontologies El-Ghalayini et al. proposed a rule based approach that was intended to later merge multiple conceptual models into one overarching conceptual model of a target domain (El-Ghalayini et al., 2005). A similar rule based approach was presented by Han et al. that directly focused on ER models (Han et al., 2010) and also used OWL while El-Ghalayini did still work with the OWL predecessors. Our presented solution is similar to those two, but puts more detail into also understanding the restrictions and cardinalities of the input ontology.

Last but not least, there is the Conceptual Model Ontology (CMO) (McCusker et al., 2011) that provides the possibility to annotate other ontologies with conceptual model concepts. One of the goals of the CMO is also to allow integration of different data sources, but rather by allowing to use a common natural language terminology to query different data sources. This only works if the concepts have already been annotated with the additional triples, while our approach tries to aid the process of creating the integration mapping instead.

3 DEMONSTRATION DATA SETS

The Mondial database[1] is a collection of data about the world - more precisely countries, cities and geographic features like mountains, lakes or rivers, as well as some demographic features like economy, religions and ethnic groups.

We selected it as one of the data sets for this publication due to the availability in several data formats and because it is more than just a trivial example, containing more than 15 entities and more than 20 individual relations. Also, it includes different concepts that are relevant for ER models like the differentiation between weak and strong entities, key attributes and all sorts of different relation cardinalities. Moreover, it provides a reference ER model that we can compare our results against.

Since we focus our work on the energy domain and also claim to support arbitrary data sources, we also include the SAVNW example power grid model from the Siemens PSS®E power system simulator in RAW format. This model represents the topology and component attributes of an electrical power grid. It is serialized as a text file that can be interpreted as a set of different tables, one for each power grid element type like busbars, transformers and generators. The serialization may contain comments for the column labels, but this is not mandatory. So without any particular inputs about the structure of the file format the ER model will not be able to extract anything else than a set of enumerated tables and a set of enumerated attributes. This scenario is a good example that needs to leverage additional user inputs about the ER model.

4 ER MODEL EXTRACTION

This Section will give a brief overview on the components of an ER model and how they can be extracted from various data source types. This is mostly limited to the important keywords to look for with a certain data type. The works referenced in the related work section can give more detailed instructions on how to perform the necessary extraction steps.

ER models are used to create an abstract representation of the most important concepts and relationships while building a database model. It helps

[1] https://www.dbis.informatik.uni-goettingen.de/Mondial/

to clarify the required capabilities of a model before transforming it into a physical set of tables and constraints in a database system. ER models basically consist of entity types, relation types, their cardinalities and attribute types.

There are several common notations to visualize ER models as diagrams. The notation used in the this paper is the Chen notation (Chen, 1976). Example diagrams can be seen in Figure 1 which will be explained in more detail in Section 5. In general, entity types are depicted as rectangular boxes, while relation types use a diamond shape. Attribute types use an oval shape and cardinalities are applied as labels to the edges between entity types and relation types.

Double-lined rectangles or diamonds indicate either so-called weak entity types or identifying relationship types respectively. This is a special way of indicating that the weak entity can not exist without the entity connected via the identifying relationship.

The usage of the term entity is sometimes not precisely clear. In order to be precise the following rules apply: An entity is the instance of an entity type and all entity instances form an entity set. An ER diagram does only depict entity types, so in natural language it can happen that the words ”type” and ”set” are omitted when talking about ER models. This effectively means that the word ”entity” is used synonymously for all of them.

4.1 Entities

Entities are often the most simple component to extract from any data source. For any tabular data like a set of CSV files or a relational database, the entities are usually reflected by the tables. However, tables can also represent $n : m$ relations, which requires some rules to check both the table's foreign and primary keys (in a relational database). In CSV based data there is not direct clue that can be used to determine if a table is a relation or not. For OWL data the entity types are usually represented by the *owl:Class* type, so the entities are the instances of that class respectively. Depending on the used OWL model it is important to use a reasoner that fills in missing class definitions from *rdfs:subClassOf* predicates.

In XML an entity is represented by an element that has attributes or does contain child elements (called *complexType* in XSD). Lastly, in hierarchical object notations like JSON all objects are considered an entity. Therefore, the ER diagram does contain one rectangle per entity set respectively. The decision what type an entity belongs to can be difficult (e.g. if there can be optional values that are not present in every object). If there is schema information available, it becomes much easier to assign objects to their respective entity type.

4.2 Relations

Whenever an entity refers to one or more other entities this is usually done with a relation.

In object notations this can either be a property that has another object as its value or its value is a reference to some kind of ID attribute. In OWL we have the dedicated *owl:ObjectProperty* concept that specifies a relation between entities. There it is necessary to correctly track the domains and ranges of the respective property to see what entities can actually be connected with this relation. For tabular data and relational databases it is harder to recognize relations. Generally a foreign key constraint in a relational database represents a relation. However, it was mentioned above that sometimes a table can also represent a relation. So to make a decision it is required to examine the foreign and private key constraints. For bare table data (e.g. CSV) without any modeled key constraints, a strategy to find potential relation types is to check for common naming patterns that involve for example ID columns in each table and a combination with the table name in others (e.g. EntityB has a column EntityA_ID). However, in these cases it is often better to just have the user correct the extracted ER model to contain proper relations.

4.3 Cardinalities

In order to carry the intended meaning, relations require quantity constraints that apply between the connected entities. For plain table data this is almost impossible to tell without any additional schema information. So, again this is one of the points that require the user to be able to correct the model with their knowledge.

Relational database schemas allow to infer cardinalities through their modeling patterns and constraints. In OWL there is the *owl:Restriction* concept as well as the exact, max and min cardinality predicates that can apply to an *ObjectProperty* and the domain (and sometimes even range) in question. However, if there is no inverse property defined there is no clear indication for the second half of the relation cardinality, as one property only defines the cardinality on the domain side. For hierarchical object notations there is no precise solution to extract cardinalities without an additional schema that specifies them. For XSD these are the *minOccurs* and *maxOccurs* attributes on the element. In JSON schema, cardinalities can be inferred from arrays if there are allowed

element counts and for an object property it can be specified if it is required or not.

4.4 Attributes

In relational databases, all attributes that are not used in the definition of foreign key constraints can be considered as attributes in the context of an ER model. In simple table based data, the attributes are all the columns that do not qualify to be part of a relation. In OWL there is the dedicated *owl:DatatypeProperty* concept for the purpose of encoding attributes. For object notations all primitive properties can be considered attributes, if they are not a reference to another entity's ID. In XML, elements without XML attributes and only a primitive content as well as XML attributes are typically regarded as attributes of the ER model.

5 EXAMPLES FROM THE FAIRlead ER MODEL EXTRACTION

Figure 1 shows the ER diagram of the province entity in three versions. It uses the mondial XML data set. The upper version is extracted without including any additional schema information. The second version does then include information from the official mondial DTD file. And the last version is generated using the official XSD file.

Several differences can be observed between the three strategies. First, the version without any schema information does use the entity name *mondial/country/province* which is our solution's way of compound naming the province entity type to indicate the hierarchical position of the entity set within the XML file. This is the case, because without schema information it might be possible to encounter different province instances at different positions in the XML tree and could not guarantee that they are equal. The *mondial/country/province/city* entity type is an example of this - also occurring as *mondial/country/city* if it does not belong to any province. This one or none relation can even be seen in the third diagram at the *citytoprov* relation that is derived from a *xsd:keyref* between the two entities.

In the lower two diagrams we see *province* to be an independent entity, which is guaranteed by either a *xsd:key* in XSD or an *ID* type in DTD. Additionally, the relations in the lower diagrams can have different names, while without a schema the only relation is *IsChild*, which reflects the hierarchical structure of the

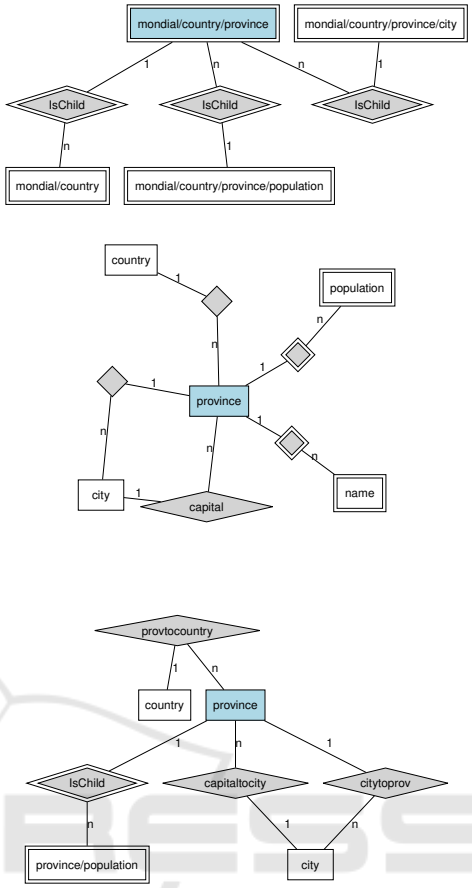

Figure 1: ER extraction of the province entity from Mondial XML data in three processing types.

XML document. Overall, it seems the XSD version is the most accurate one - but for many other entities the quality of the model lacks behind the schema information available from a relational or ontological model.

An example for the lacking quality that is based on the modeling decisions in the XSD file can be seen in Figure 2. It shows, a pattern that can often be seen in the model extracted from the XSD. The highlighted entity *river/located* can also be seen as a relation itself. For this kind of scenario it might be possible to contract the resulting ER diagram and produce an *n to n* relation *located*. This is something that we might consider in the future to optimize certain patterns in the resulting ER diagram independent of the original data source.

Looking at Figure 3 is a nice example of how cardinalities can be extracted from OWL data. The *isBorderOf* relation is correctly reflected as a *2 to n* relationship. Moreover, the *locatedIn* relation has no restrictions in OWL, so it correctly is depicted as a many-to-many relationship.

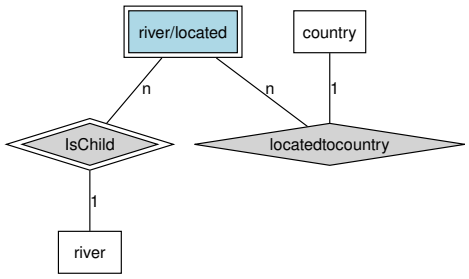

Figure 2: Extracted ER model from the XSD schema that could be contracted with a post-processing step.

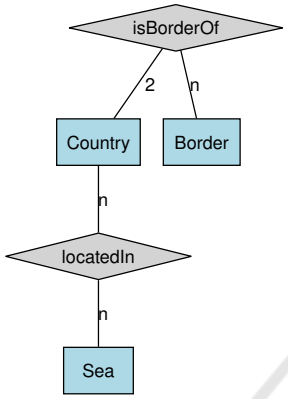

Figure 3: Extracted ER model of the OWL data set that showcases the cardinality possibilities in an ontology.

Considering the PSS®E data set, the matter is more complicated. The file theoretically can include structural metadata in the form of comments, but that is not mandatory to be a valid file. Depending on the source the file was received from, these comments might not be available. So in the worst case scenario, an extracted ER model might contain no information on table and column names. So after some quick pre-processing to split the RAW file into a set of CSV files, the only information for the ER model available is a set of entities named *Table1* to *TableN*. Each of those entities will have a set of attributes named *Column1* to *ColumnN* respectively. At that point it is mandatory, to allow the user to edit this extracted ER model. It might be enough to just correct the bits and pieces that are later relevant for the data integration step. An example of such a user corrected ER model can be seen in Figure 4.

Finally, it must be noted that none of the extracted models exactly matches the original ER diagram of the mondial database. A reason for this are the specific modeling decisions that have been made to create the respective physical models, that do not allow for an exact reconstruction. In general, this is not a problem for the target purpose of performing data integration. In our approach the domain expert that knows the data sources is the same person that shall perform the visual data integration. This means as long as the ER representation is reflecting the data source more or less accurately the user will be able to make sense of it.

However, there are extreme cases like the XML file with no schema information or the PSS®E data set as well as the cases where relations have been detected as entities. In those examples, it may not be possible to infer any useful ER model. In that regard the presented approach is semi automatic and the user needs to introduce corrections to the ER model. This has been shown as an example with the PSS®E data set and the ER model in Figure 4.

6 CONCLUSION AND FUTURE WORK

With our data integration and management solution FAIRlead, we want to enable domain experts to perform data integration in a way that improves the FAIRness of their existing data. As a first step of this solution, we have presented an open-source tool that allows the extraction of ER models from various data sources that can be reused for future data integration efforts. Moreover, additional data sources can be integrated in the model extraction either by implementing a new converter or by converting the input data to an already supported format. Many potential data sources can likely be transformed into one of the presented ER model extraction solutions with a small pre-processing step. With the example of the PSS®E RAW file, it has become clear that the user must be able to improve the extracted ER model, because the file can technically come without any structural metadata.

Finding an appropriate solution for the model corrections and also to allow visual editing of the extracted schema is one of our future steps. Moreover, we will continuously improve the current implementation, to produce accurate ER models according to the given input data. This for example includes the integration of existing schema generation tools into the process (e.g. to generate a DTD out of XML data before processing it).

Based upon the FAIRlead ER model extraction strategies presented in this paper, we will create a GUI that allows to perform data integration from multiple heterogeneous data sources using conceptual models. This user interface will use a flow-based programming approach to visually show the link between original data source entities and the resulting ontological concepts.

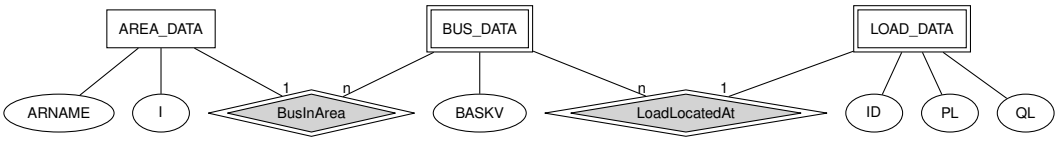

Figure 4: User corrected ER model of the PSS®E data set.

ACKNOWLEDGEMENTS

The authors would like to thank the German Federal Government, the German State Governments, and the Joint Science Conference (GWK) for their funding and support as part of the NFDI4Energy consortium. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 501865131 within the German National Research Data Infrastructure (NFDI, https://www.nfdi.de/).

This publication was also supported by the Helmholtz Metadata Collaboration (HMC, https://www.helmholtz-metadata.de/), an incubator-platform of the Helmholtz Association within the framework of the Information and Data Science strategic initiative.

REFERENCES

Alalfi, M. H., Cordy, J. R., and Dean, T. R. (2008). SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas. In *2008 15th Working Conference on Reverse Engineering*, pages 187–191.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

Chiang, R. H. L., Barron, T. M., and Storey, V. C. (1994a). Performance evaluation of reverse engineering relational databases into extended Entity-Relationship models. In Elmasri, R. A., Kouramajian, V., and Thalheim, B., editors, *Entity-Relationship Approach — ER '93*, pages 352–363, Berlin, Heidelberg. Springer.

Chiang, R. H. L., Barron, T. M., and Storey, V. C. (1994b). Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Data & Knowledge Engineering*, 12(2):107–142.

Cunha, J., Erwig, M., Mendes, J., and Saraiva, J. (2016). Model inference for spreadsheets. *Automated Software Engineering*, 23(3):361–392.

Cunha, J., Erwig, M., and Saraiva, J. (2010). Automatically Inferring ClassSheet Models from Spreadsheets. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 93–100.

Cunha, J., Fernandes, J. P., Mendes, J., and Saraiva, J. (2012). Extension and implementation of ClassSheet models. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 19–22.

Della Penna, G., Marco, A. D., Intrigila, B., Melatti, I., and Pierantonio, A. (2006). Interoperability mapping from XML schemas to ER diagrams. *Data & Knowledge Engineering*, 59(1):166–188.

El-Ghalayini, H., Odeh, M., Mcclatchey, R., and Solomonides, T. (2005). Reverse Engineering Ontology to Conceptual Data Models.

Gomez-Perez, A., Fernández-López, M., and Vicente, A. (2000). Towards a Method to Conceptualize Domain Ontologies.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. PhD thesis.

Han, L., Xu, J., and Yao, Q. (2010). Entity-Relationship semantic meta-model based on ontology. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, volume 11, pages V11–219–V11–222.

Klímek, J. and Nečaský, M. (2010). Reverse-engineering of XML Schemas: A Survey. In *CEUR Workshop Proceedings*, volume 567, pages 96–107.

May, W. (1999). Information Extraction and Integration with Florid: The Mondial Case Study. Technical Report 131, Universität Freiburg, Institut für Informatik.

McCusker, J., Luciano, J., and Mcguinness, D. (2011). Towards an Ontology for Conceptual Modeling. *CEUR Workshop Proceedings*, 833.

Mello, R. d. S. and Heuser, C. A. (2001). A Rule-Based Conversion of a DTD to a Conceptual Schema. In S.Kunii, H., Jajodia, S., and Sølvberg, A., editors, *Conceptual Modeling — ER 2001*, pages 133–148, Berlin, Heidelberg. Springer.

Siau, K., Shiu, H., and Fong, J. (2011). Reverse Engineering from an XML Document into an Extended DTD Graph. pages 101–119.

Yang, W., Zhan, M., Wang, Q., and Shi, B. (2004). A conversion of a DTD to conceptual model by using UML. In *The Fourth International Conference onComputer and Information Technology, 2004. CIT '04.*, pages 303–308.

APPENDIX

The FAIRlead code used to perform the ER model extraction can be found on github:
https://github.com/Cpprentice/FAIRlead-model-extraction