Information Processing Letters 138 (2018) 81–88

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Building a business domain meta-ontology for information pre-processing

Yevgen Biletskiy *, J. Anthony Brown, Girish R. Ranganathan, Ebrahim Bagheri, Ismail Akbari

University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, E3B 5A3, Canada

ARTICLE INFO

Article history:
Received 4 September 2011
Received in revised form 12 June 2018
Accepted 20 June 2018
Available online 25 June 2018
Communicated by José Luiz Fiadeiro

Keywords:
Software engineering
Ontology
Information extraction

ABSTRACT

Business analysts, along with other business domain software application users, have created a vast amount of business documents, which often do not have any business domain ontologies in the background. This situation leads to misinterpretation of such documents, when being processed by machines, that results in inhibiting the productiveness of computer-assisted analytical work and effectiveness of business solutions due to lack of effective semantics; therefore, business analysts (especially, if rotating) can use well-designed business domain ontologies as a backbone for their official applications. The process of extracting and capturing domain ontologies from these voluminous documents requires extensive involvement of domain experts and application of methods of ontology learning that is substantially labor intensive; therefore, some intermediate solutions which would assist in capturing business domain ontologies must be developed. The present paper proposes a solution in this direction which involves building a meta-ontology as a rapid approach in conceptualizing a business domain from huge amounts of source documents. This meta-ontology can be populated by ontological concepts, attributes and relations from business documents, and then refined in order to form better business domain ontology either through automatic ontology learning methods or some other traditional ontology building approaches.

1. Introduction

Ontologies play an extremely significant role in knowledge-based software applications, primarily because they facilitate automatic semantic interpretation and interoperability of data from various information sources. Defined by Gruber as "an explicit specification of conceptualization" [1], an ontology can serve as an effective and powerful tool to capture, store, and work with domain knowledge in information systems. The definition of ontology has many variants, which can be generalized under a machine-readable, structured representation of information.

The success of business applications mainly depends on how well the business domain ontology is designed. Because the business domain ontology is intended to specify the conceptualization of a particular real-world business domain. Business domain ontology is used to define concepts and concepts' relationships in a business domain. Business domain ontology makes concepts and their relationships to have a unified and clear definition in that business domain. Constructing a business domain ontology can significantly improve knowledge sharing and interoperability between heterogeneous systems and participants in the business domain [2]. In most organizations, domain knowledge is stored across countless semi-structured

* Corresponding author.
E-mail addresses: biletski@unb.ca (Y. Biletskiy), anthony.brown@unb.ca (J.A. Brown), girish.ranganathan@unb.ca (G.R. Ranganathan), bagheri@ee.ryerson.ca (E. Bagheri), iakbari@unb.ca (I. Akbari).

https://doi.org/10.1016/j.ipl.2018.06.009
0020-0190/

and unstructured documents (i.e. MS Word, Excel, PDF, etc.), which are user friendly, but are without any type of ontology or semantics in the background. As a result, knowledge engineers (KEs) are tasked with the design, development and population of well-designed ontologies using the information from these documents. This is a time- and resource-consuming process that requires extensive involvement of domain experts and advanced ontology learning methods. If the knowledge in the source documents is first pre-processed and automatically converted into a consolidated, intermediate, ontology-mediated format, the job of the KE becomes immensely simplified.

Although there are many methodologies for building ontologies [3], the generic approach – *Skeletal Methodology*, proposed by Uschold and King [4], provides very general guidelines for developing ontologies through four stages: identification of purpose, building the ontology, evaluation, and documentation. The present paper proposes an addition to the *Skeletal Methodology*, specifically during the process of building domain ontologies. The proposed business domain ontology engineering methodology includes the additional phase of "Information Pre-Processing" prior to the "Building Ontology" phase. Pre-processing includes: defining a set of objects participating in the subsequent phases of building the ontology; identifying sets of attributes characterizing objects; identifying data sources (source documents); and defining domains. As a result, the original information is transformed into structured ontological instance data, which is machine interpretable and compatible with existing ontology learning methods. Pre-processing the source document content in this way greatly simplifies the subsequent process of producing (using ontology learning methods) a semantically rich domain ontology, which includes all necessary ontological components [3]: concepts, attributes, taxonomies, non-taxonomical relations, functional relations, constraints, axioms, and instances, which can be used by business applications to improve their performance.

This paper proposes to pre-process domain knowledge from various source documents using a general, domain-independent, background meta-ontology. A meta-ontology is a schema for other ontologies. Meta-ontology is a simple ontology whose concepts are super-classes for a further refined domain ontology. The meta-ontology in this paper will serve to develop an ontology in a way that the requirements of the ontology are met. The rationale behind these requirements is that they cover important properties of an ontology while building it [5]. The meta-ontology is domain-independent and it has general classes Domain, Document, Entity, and Attribute. The idea behind this work is to ease or accelerate the process of ontology learning when creating business domain ontologies from business documents. Pre-processing domain knowledge can aid several steps in the ontology learning process, especially concept formation, concept hierarchy induction, and ontology population [6]. The other ontology learning steps, such as relation identification and relation hierarchy induction may or may not benefit from the creation of this meta-ontology depending on the nature of the information being captured and stored because the relations used in the presented meta-ontology were created manually. But these manually created relations might serve as a starting point to extract relations and relation hierarchies from business documents. The work is based on the Business Domain Ontology Development Framework (BDODF) academia-to-industry knowledge transfer project with the substantive part to convert a large business analysis knowledge base in MS Excel format to OWL.

The paper is presented in the following structure: Section 2 presents related work in the field of business domain ontologies; Section 3 describes the goal of this work in more detail and presents the proposed business domain meta-ontology; Section 4 describes an example of customizing and populating the meta-ontology for an example source document; and finally, Section 5 concludes the work and states directions for further research.

2. Related work

A tremendous amount of research has been conducted in the past decade regarding business domain ontologies and their management. Generally, domain ontologies are intended to specify the conceptualization of a particular real-world domain. Business domain ontologies in particular, describe a set of concepts and activities related to business domains such as finance, commerce or industry involved in the production and/or delivery of goods and services.

The framework FLOPPIES [7] is a semi-automatic ontology population from documents in tabular format of web-based e-Commerce storages. Using the tabular data available on Web store product pages, the framework creates an ontology OntoProduct by employing user-defined annotations for lexical and pattern matching which facilitates product classification, property association and value extraction. The created OntoProduct ontology is mapped to GoodRelations [8] ontology for e-commerce. GoodRelations is an ontology for describing many aspects of e-commerce, such as businesses, products and services, offerings, opening hours, and prices on the Web. Since 2012, GoodRelations is integrated to schema.org vocabulary which means that schema.org can now describe more granular product information.

NOR2O [9] proposes guidelines to use of re-engineering patterns for transforming non-ontological resources such as classification schemes, thesauri, lexicons and folksonomies into ontologies. These patterns define a procedure which transforms non-ontological resources into ontologies (using TBox and ABox transformations) taking into account the resource type and its data model.

The NeOn methodology [10] is a scenario-based methodology that supports the collaborative ontology development and reuse. It proposes nine different scenarios emphasizing the reuse of ontological and non-ontological resources, merging, localizing of these resources and taking into account collaboration and dynamism. Neon methodology also includes a glossary of processes and activities involved in the development of ontologies.

In addition to ontology generation methods in business domain or other domains, various business domain ontologies have been proposed during the last few years, for instance: Enterprise Ontology [4], ontology for en-

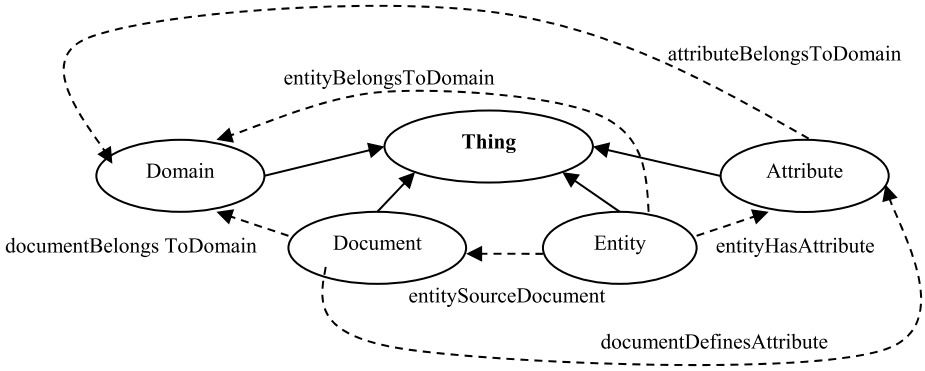

Fig. 1. The business domain meta-ontology (details are omitted).

terprise modeling [11], TOVE Ontology Project for enterprise modeling [12], Resource Event Agent Ontology [13] can be extended as a business domain ontology and its OWL-formalization [14], and many others. TOVE Ontology Project for enterprise modeling [12] develops an ontology that has the following characteristics: It provides a shared terminology for the business domain, the semantics (meanings) of each term or concept, and graphical context of each term. It also implements the semantics in Prolog that can answer commonsense questions from the business.

Although the existing approaches, methods and tools provide a solid basis for the design and re-engineering of business domain ontologies, they fail to explain a methodology through which a business analyst could capture ontologies from existing business domain documents without extensive involvement of a domain expert or knowledge engineer. This indicates the need for an automatic method to pre-process domain knowledge from existing source documents and transform them into an easily reusable ontological format.

The work described in the present paper proposes to use a business domain meta-ontology for pre-processing domain knowledge. In this work, the meta-ontology has been used in the business domain with sub-domain of contracts. Hence, the business domain meta-ontology simply refers to the meta-ontology that is being used in the business domain (with the contracts sub-domain). The meta-ontology here has for general classes Domain, Document, Entity, and Attribute which take business domain source documents as input and pre-process them for the ontology building step. Analogous to our work, Herre and Heller [15] have described a relevant approach of semantic modeling using top-level ontology concepts in the field of medical information systems. In the enterprise domain, Lee et al. [16] propose the layered ontology representation model as a foundation for the product ontology architecture for collaborative enterprises. Also in order to solve some problems from the lack of semantics of the Enterprise Architecture, an ontology-based Enterprise Architecture has been suggested by Kang et al. [17], where the enterprise is modeled according to the Zachman Enterprise Architecture framework [18]. The authors have developed a meta-model in order to model relationships of Enterprise Architecture components. Besides domain ontologies, meta-modeling is also used to develop process-centered enterprise ontologies. They use of an ontology-based retrieval process, documents about a product, produced by a given project produces or a customer buying a specific product can be retrieved [19].

3. The business domain meta-ontology

The overall goal of this work is to create a general business domain meta-ontology as well as to deploy an extendable ontology population tool capable of populating the background meta-ontology with instances extracted from structured and semi-structured business domain source documents. The resulting populated meta-ontology can then be reused (possibly involving ontology learning methods such as OnToKnowledge [20] and DILIGENT [21], as well as ontology evaluation techniques such as OntoClean [22], and/or ontology building techniques such as METHONTOLOGY [23]) to form a normal, semantically rich business domain ontology. As discussed above, the need of meta-ontology is required to identify candidates for classes/entities of a regular ontology. Instances of entity in the meta-ontology become entities in the regular ontology.

The meta-ontology in Fig. 1 is proposed to model the information from business domain source documents (solid line – subclassOf relations, dashed line – non-taxonomical relations):

- Thing – an upper-level concept conceptualizing the entire enterprise information system (or whole business domain) such as “Business Domain” or “Enterprise”. In order to keep the name of the upper-level concept open to modifications, the default name “Thing” is used;
- Document – specifies a source document of any type used in the business domain;
- Domain – specifies business sub-domains under the main domain for example the insurance domain;
- Entity – specifies a particular concept in the business domain;
- Attribute – specifies a property (or characterization) of an entity.

The structure of the meta-ontology is generally independent from the domain knowledge (here business domain). Fig. 1 shows the meta-ontology in general. Business domain meta-ontology is the meta-ontology (Fig. 1) populated with instances from business domain (here insurance subdomain). The instances are extracted from the source documents in the insurance domain. However, the instances are omitted in the Fig. 1 for simplicity. In summary, an *Entity* is a concept or term defined in a *Document* that is important in a particular business *Domain* and an *Attribute* describes an aspect of a corresponding Entity. For example, a "Database Mapping" Document in the "Insurance" Domain could contain a "Sale" Entity, which would have a corresponding "Active" Attribute, which is set to either true or false depending on the status of a sale. Additionally, both Entities and Attributes have "facets" (not shown in Fig. 1), which are essentially string literals describing them (i.e. datatype properties). For example, an Entity could have a "name" facet and a "code" facet or an Attribute could have "default value" or "sample value" facets.

This meta-ontology also includes non-taxonomical (association) relations (Fig. 1); for example, an Entity could have the following relationships to other entities:

- EntityInstanceX entityHasAttribtues AttributeInstanceA;
- EntityInstanceX entityBelongsToDomain DomainInstanceA;
- EntityInstanceX entitySourceDocument DocumentInstanceA.

Where EntityInstanceX is an instance of the class Entity (of the meta- ontology) and can be a class candidate for the regular ontology that is obtained from this meta-ontology. It is similar for the AttributeInstanceA. Similar inter-concept relationships (i.e. object-type properties) can be found between all concepts in the ontology.

The structure of the meta-ontology is generally independent from the domain knowledge. Hence the instance data for the specific business domain meta-ontology can be extracted from the source documents using simple, automatic information extraction techniques. The instance data can be effectively pre-processed and stored in a general, platform-independent, ontological format. It should be noted that the focus of this paper is on building the meta-ontology itself as "Information Pre-Processing" prior to the "Building Ontology" phase and not the process of extracting instance information or the process of reusing the meta-ontology into a true ontology.

4. Customizing and implementing the meta-ontology

There are countless types and formats of business documents, almost any of which could theoretically be used to populate the proposed meta-ontology; however, the prototype presented as part of this work uses four types of semi-structured business domain documents as examples: Domain Analysis Entity Tables (ET); Database Mapping (DB); Report Analysis (RPT); and User Interface Analysis (UI) documents, all of which are in either MS Word or MS Excel format. These document types may not be common across all business domains, but serve as a functional, real-world example of the type of documents that the system is capable of dealing with. While the current research focuses on these four document types, the population tool provides a simple mechanism to add new document types as ontology population sources by customizing the meta-ontology with additional classes and developing a new set of extraction rules to be associated with the new document type.

In general, business domain documents are (semi) structured for human readability, without any semantics to support machine processing. As a result, the process of populating the meta-ontology in the prototype is more dependent on the structure and format of the documents than the information itself. The most common mechanism for presenting domain information is via a table in MS Word or Excel, wherein the headers and structure of the table can be used to identify the relevant information. Different types of source documents will define entities and attributes in different ways, so the proposed meta-ontology must be customized to support each type of source document. For example, if the proposed work is going to support Database Mapping (DB) source documents, then a subclass to the Document, Entity and Attribute classes will be added to the meta-ontology to support this type of document. Customization entails creating a subclass of the Document, Entity and Attribute classes corresponding to each document type, where each of these subclasses is further specialized with additional object- and data-type properties.

Meta-ontology (Fig. 1) is populated with Instances for its entities (Domain, Document, Entity, Attribute). For example, any concept or term in a business domain document that is important to that business domain is extracted and will be added to the meta-ontology as an instance of its Entity. In a similar way, any attribute that describes an aspect of an entity is added to the meta-ontology as an instance to the Attribute entity. This process builds the business domain meta-ontology from the meta-ontology. As the next step, ontology learning or ontology building methods such as OnToKnowledge [20] or METHONTOLOGY [23] can utilize this business domain meta-ontology to build the domain ontology. For example, instances of the 'Entity' or 'Attribute' entities from business domain meta-ontology can be used as classes and attributes to build the domain ontology.

An example of the meta-ontology, customized to work with the four source document types described above, is presented in Fig. 2, where:

- ET Doc – Entity Table Document;
- DB Doc – Database Mapping Document;
- X Doc – Document of any type;
- DA Ent – Domain Analysis Entity from Entity tables;
- DB Ent – DB Mapping Entity from DB Mapping Documents;
- X Ent – X-type Entity from X-type Documents;
- DA Att – Domain Analysis Attribute from Entity tables;

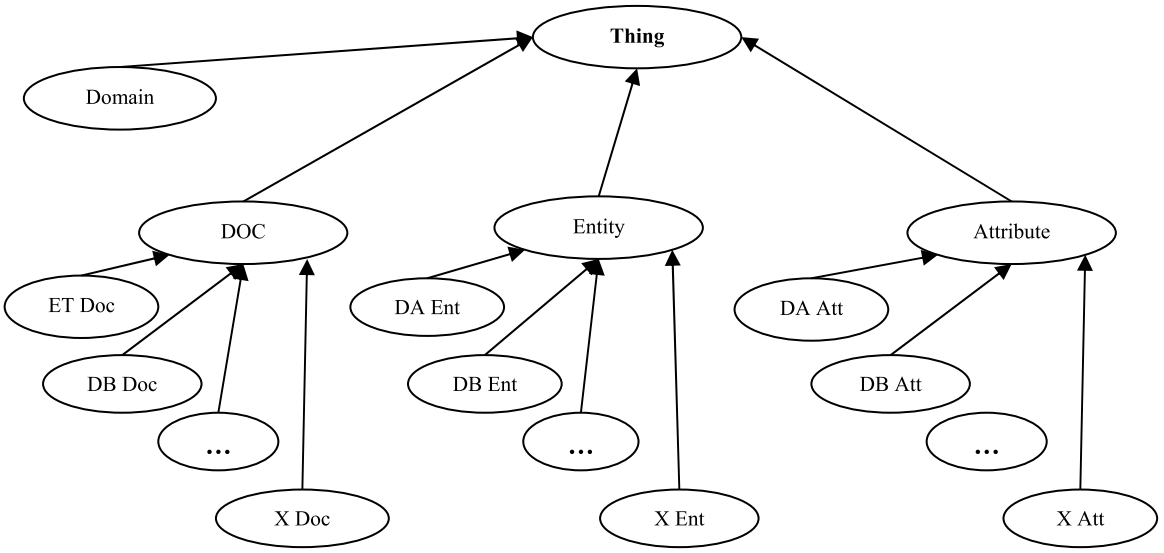

Fig. 2. The document type specific business domain meta-ontology.

CTR55	Insurance Contract
Description	An Insurance contract determines the legal framework under which the features of an insurance policy are enforced.
Definitions	Define important terms used in the policy language.
Insuring Agreement	Describes the covered perils, or risks assumed, or nature of coverage, or makes some reference to the contractual agreement between insurer and insured. It summarizes the major promises of the insurance company, as well as stating what is covered.
Declarations	Identifies who is an insured, the insured's address, the insuring company, what risks or property are covered, the policy limits (amount of insurance), any applicable deductibles, the policy period and premium amount.
Exclusions	Take coverage away from the Insuring Agreement by describing property, perils, hazards or losses arising from specific causes which are not covered by the policy.
Conditions	Provisions, rules of conduct, duties and obligations required for coverage. If policy conditions are not met, the insurer can deny the claim.

Fig. 3. An example of source document from the insurance sub-domain.

- DB Att – DB Mapping Attribute from DB Mapping Documents;
- X Att – X-type Attribute from X-type Documents.

As a proof of concept, an OWL (Web Ontology Language) [24] model of the above meta-ontology was created using Protégé [25] as the ontology editor in the present work. Any other ontology language [26] or ontology editor can be used for this purpose. Within the background meta-ontology, there are no instance or property values, just concepts and properties, which allow the knowledge engineer to specify precisely what information they wish to capture, independent of the information extracted from the source documents. OWL was chosen based on several reasons, specifically because it is portable, widely used, supports declaration of properties at the object level (i.e. properties are part of the background ontology, not extracted with the instances), and stores validation information in the ontology as OWL constraints, which simplifies any subsequent validation processes.

An example of an MS Excel Entity Table, a source document from the sub-domain of *Contracts* is presented in Fig. 3. In this example, it is clear to a human reader that the first row defines an entity and the proceeding rows describe the defined entity; however, a computer is not able to automatically make this distinction without additional semantic information, which can be obtained through pre-processing. The customized background meta-ontology used to capture information from Entity Table source documents is presented as a Protégé snapshot in Fig. 4, with the taxonomy shown in the left frame and the properties of the ContractEntity class in the lower right frame. Note the additional domain-specific ontological properties that have been added to capture the

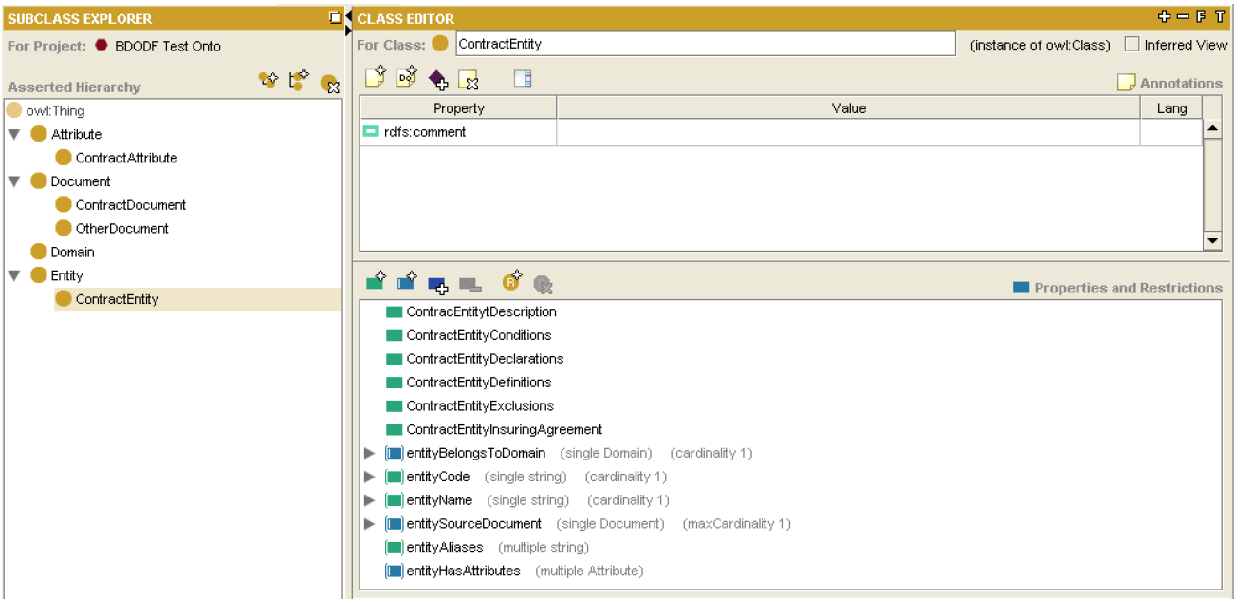

Fig. 4. The background ontology to capture information from documents in the insurance sub-domain.

Table 1
General view of the extraction rules.

Rule returnDatatype methodName(param1, param2)			
Cases	Condition1	Condition2	A1/RET1
	Boolean statement(s) that use cases data – java code	Boolean statement(s) that use cases data – java code	Java code that uses the case result data (action or return)
Case 1	C1Data1	C1Data2	Result Data
Case 2	C2Data1	C2Data2	Result Data
Case 3	C3Data1	C3Data2	Result Data

domain-specific structure of this particular type of document (e.g., ContractEntityDescription).

The position-based information extraction process from source documents is performed by OpenL Tablets [21], a tool mainly devoted to externalization and processing data presented in tabular format, and a set of extraction rules corresponding to the particular type of the source document (Fig. 5). The extraction process is based on the table structure and patterns, not the semantics of the data, making it useful for any document using this particular table structure, and easily customizable to fit other types of source documents. As long as the source document has a tabular format such as a Microsoft Excel table or a table in a Word document, position-based extraction process can be used to extract data from that source document.

Generally, the extraction process can be presented as follows. The rules are created using OpenL, Tablets [21] meaning they are a combination of Java code and decision tables. OpenL allows java code to be externalized from the system and re-compiled at runtime. This means you can add, remove, or modify parsing logic without affecting the application code. Some of the OpenL parsing rules are simply actions necessary for parsing, with no decision logic involved. For example, there is a ruleOpenL “rule” that simply iterates through each row of a given excel table and calls the appropriate rule associated with that row. Rules containing decision tables have the structure as demonstrated in Table 1.

The decision table works as follows: If the given input parameters cause all the boolean condition statements to be true for one of the cases (based on the corresponding case data), the action/return code for the decision table is triggered, using the case result data. For example, the rule in Table 1 contains 2 conditions and 3 cases. This rule accepts a row from an Excel table as a parameter and classifies the row based on the width (# cells) and the pattern in the left-most cell. The return value of the method (*val*) is either null or one of the three strings in the RET1 column. A similar type of rule could be made for any document type, such as MS Word, where the rule would accept a row of a MS Word table, or HTML, where the rule would accept a row from an HTML table. Essentially, the rules prepared in iteration 1 can be re-used as type of rule template for extraction. No matter what type of source document is used, very similar parsing rules have to be created, the only difference is the internals of exactly how that parsing occurs.

In the example in Fig. 5, the extraction process works as follows. The first row is so-called Decision Table Header. It consists of the keyword “Rules” and table signature which includes return value type, table name and input parameters in quotes. The 2nd row contains condition headers

Rules String classifyEntityTableRow(ILogicalTable lrow, DocumentInfo doc, String uri)							
C1	C2		C3		C4	C5	RET1
int rowWidth=lrow.getLogicalWidth(); if(rowWidth>=minWidth)return true; else{ //System.out.println("C1 Failed"); return false;}	String content=lrow.getLogicalColumn(index).getGridTable().getStringValue(0, 0); if(content!=null){ content = content.trim(); Pattern p = Pattern.compile(pat); Matcher m = p.matcher(content); if(m.matches()) return true; else{ //System.out.println("C2a Failed"); return false;}		String content=lrow.getLogicalColumn(index).getGridTable().getStringValue(0, 0); if(content!=null) { content = content.trim(); Pattern p = Pattern.compile(pat); Matcher m = p.matcher(content); if(m.matches()) return true; else{		if(check){ if(lrow.getLogicalColumn(0).getGridTable().getCellInfo(0,0).getFont().isStrikeout()& & lrow.getLogicalColumn(1).getGridTable().getCellInfo(0,0).getFont(	if(check){ String content=lrow.getLogicalColumn(0).getGridTable().getStringValue(0, 0); if(content!=null){ content =	return val;
int minWidth	int index	String pat	int index	String pat	boolean check	boolean check	String val
Minimum Row Width (# Cells)	**Column A Index**	**Pattern A**	**Column B Index**	**Pattern B**	**eck Strikethrou**	**Document Dor**	**Return Value**
2	0	\p{Alpha}+\d+	1	\p{ASCII}+|.+	FALSE	FALSE	entity
2	0	\p{Alpha}+ \d+	1	\p{ASCII}+|.+	FALSE	FALSE	entity
2	0	\p{Alpha}+ \d+-\d+	1	\p{ASCII}+|.+	FALSE	FALSE	entity
5	0	\d+|\p{Digit}+	1	\p{ASCII}+|.+	TRUE	FALSE	attribute
2	0	\D+	1	\p{ASCII}+|.+	FALSE	FALSE	entity facet

Fig. 5. The set of extraction rules corresponding to each source document type.

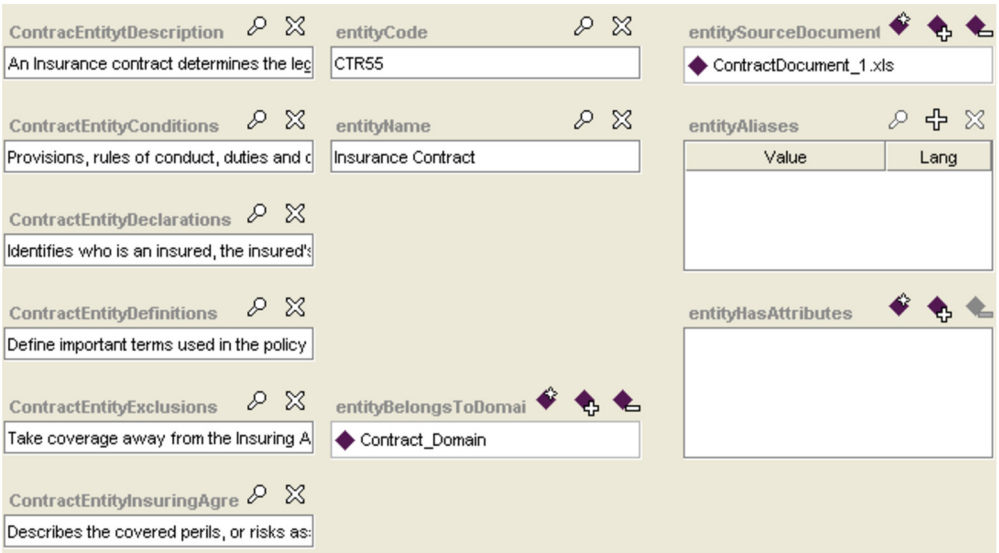

Fig. 6. The Protégé snapshot of the meta-ontology populated by information from source documents from the Contracts sub-domain.

(C1, C2, HC1, etc.), action and return headers (A1, RET1, RET2, etc.). These headers must follow the exact syntax. In other words, condition headers must start with "C", actions with "A" and return values with "RET", along with a number. The 3th row contains condition/action/return statements. OpenL Tablets supports Java grammar enhanced with OpenL Business Expression (BEX) grammar features [27]. The 4th row shows the parameter definition cells. The 5th row is business display names for parameters. Then at the last, the table will be filled with data (concrete parameter values of rules). A rule is executed only when all its conditions are true. Absence of a parameter in a condition cell is interpreted as a true value. Blank action or return value cells are ignored. Decision table returns the first not blank action/return value whose conditions are true.

JENA, the Semantic Web framework for Java, Application Program Interface (JENA API) [28] was used to populate the meta-ontology with the extracted information (as instances of subclasses of the Entity and Attribute classes). The populated meta-ontology is presented as a Protégé snapshot in Fig. 6, with the same information from the source document. The meta-ontology can also be automatically populated by other ontology population methods [6]. The populated meta-ontology has thus automatically captured the domain information in a semantically-rich, machine-readable format that can be refined to a true business domain ontology.

5. Conclusion

The work presented in this paper was devoted to the development of a business domain meta-ontology, which can be populated with information from various business domain source documents, by effectively pre-processing the necessary information and preparing the knowledge base. The proposed meta-ontology, implemented in OWL, served as a background ontology for a set of OpenL extrac-

tion rules to automatically extract and capture ontological information (in particular – instances, attributes and relations) from semi-structured documents (e.g. Excel Entity Tables). The populated meta-ontology can serve as an input to various ontology learning methods to build a true business domain ontology. Future research includes re-use and development of ontology learning methods, which allow the populated meta-ontology to be used to build a business domain ontology; as well as re-use and development of more sophisticated methods of information extraction for the meta-ontology population. The developed meta-ontology and extraction/population system have the potential to significantly reduce the time and resources required to form a business domain ontology from semi-structured source documents.

Acknowledgements

The research being presented is inspired by a prototype developed by Exigen, Inc. The work is based on the Business Domain Ontology Development Framework (BDODF) academia-to-industry knowledge transfer project supported by the Atlantic Canada Opportunities Agency, Exigen Canada Inc., and the University of New Brunswick, Canada.

References

[1] T.R. Gruber, A translation approach to portable ontologies, Knowl. Acquis. 5 (2) (1993) 199–220.
[2] Guoqiang Zhang, Suling Jia, Qiang Wang, Qi Liu, REA-based Enterprise Business Domain Ontology Construction, J. Softw. 5 (5) (2010) 522.
[3] A. Gomez-Perez, O. Corcho, M. Fernandez-Lopez, Ontological Engineering: With Examples from the Areas of Knowledge Management e-Commerce and the Semantic Web, Advanced Information and Knowledge Processing, Springer, ISBN 1-85233-551-3, 2004, 415 pp.
[4] M. Uschold, M. King, S. Moralee, Y. Zorgios, The enterprise ontology, in: Special Issue on Putting Ontologies to Use, Knowl. Eng. Rev. 13 (1998) 31–89.
[5] J.L.G. Dietz, N. Habing, A meta Ontology for Organizations, On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops, Springer, Berlin Heidelberg, 2004.
[6] P. Buitelaar, P. Cimiano, A. Frank, M. Hartung, S. Racioppa, Ontology-based information extraction and integration from heterogeneous data sources, Int. J. Hum.-Comput. Stud. 66 (2008) 759–788.
[7] L.J. Nederstigt, S.S. Aanen, D. Vandic, F. Frasincar, FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores, Decis. Support Syst. 59 (2014) 296–311.
[8] M. Hepp, Goodrelations: an ontology for describing products and services offers on the web, in: Knowledge Engineering: Practice and Patterns, Springer, Berlin Heidelberg, 2008, pp. 329–346.
[9] B.M. Villazón-Terrazas, A Method for Reusing and Re-Engineering Non-Ontological Resources for Building Ontologies, vol. 12, IOS Press, 2012.
[10] M.C. Suarez-Figueroa, G.A. de Cea, C. Buil, K. Dellschaft, M. Fernandez-Lopez, A. Garcia, A. Gomez-Perez, et al., NeOn methodology for building contextualized ontology networks, NeOn Deliverable D5.4.1, 2008.
[11] M.S. Fox, M. Barbuceanu, M. Gruninger, An organization ontology for enterprise modeling: preliminary concepts for linking structure and behaviour, Comput. Ind. 29 (1–2) (1996) 123–134.
[12] M.S. Fox, M. Gruninger, Enterprise modeling, AI Mag. 19 (1998) 109–121.
[13] G.L. Geerts, W.E. McCarthy, An ontological analysis of the economic primitives of the extended-REA enterprise information architecture, Int. J. Account. Inf. Syst. 3 (2002) 1–16.
[14] F. Gailly, G. Poels, in: Towards Ontology-Driven Information Systems: Redesign and Formalization of the REA Ontology, in: LNCS, vol. 4439, Springer-Verlag, 2007, pp. 245–259.
[15] H. Herre, B. Heller, Semantic foundations of medical information systems based on top-level ontologies, Knowl.-Based Syst. 19 (2) (2006) 107–115.
[16] J. Lee, H. Chae, C-H. Kim, K. Kim, Design of product ontology architecture for collaborative enterprises, Expert Syst. Appl. 36 (2 Part 1) (2009) 2300–2309.
[17] D. Kang, J. Lee, S. Choi, K. Kim, An ontology-based enterprise architecture, Expert Syst. Appl. 37 (2) (2010) 1456–1464.
[18] J.A. Zachman, A framework for information systems architecture, IBM Syst. J. 26 (3) (1987) 276–292.
[19] K.H. Han, J.W. Park, Process-centered knowledge model and enterprise ontology for the development of knowledge management system, Expert Syst. Appl. 36 (4) (2009) 7441–7447.
[20] Sure, S. Staab, R. Studer, On-to-knowledge methodology (OTKM), in: Handbook on Ontologies, Springer, Berlin Heidelberg, 2004, pp. 117–132.
[21] H.S. Pinto, S. Staab, C. Tempich, DILIGENT: towards a fine-grained methodology for distributed, loosely-controlled and evolving, in: In-Ecai 2004: Proceedings of the 16th European Conference on Artificial Intelligence, vol. 110, IOS Press, 2004, p. 393.
[22] C. Welty, N. Guarino, Supporting ontological analysis of taxonomic relationships, Data Knowl. Eng. 39 (1) (2001) 51–74.
[23] M. Fernández-López, A. Gómez-Pérez, N. Juristo, METHONTOLOGY: from ontological art towards ontological engineering, in: Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, 1997, pp. 33–40.
[24] OWL: Web Ontology Language, 2004. Available December 1, 2009, http://www.w3.org/2004/OWL.
[25] Protégé, Available September 1, 2011, http://protege.stanford.edu/, 2000.
[26] J.R.G. Pulido, M.A.G. Ruiz, R. Herrera, E. Cabello, S. Legrand, D. Elliman, Ontology languages for the semantic web: a never completely updated review, Knowl.-Based Syst. 19 (7) (2006) 489–497.
[27] OpenL Tablets, Available September 1, 2011, http://openl-tablets.sourceforge.net/, 2006.
[28] JENA API, Available September 1, 2011, http://jena.sourceforge.net/, 2006.