Transforming Existing Knowledge Models to Information Extraction Ontologies

Marek Nekvasil Vojtěch Svátek Martin Labský

Department of Information and Knowledge Engineering, University of Economics, Prague, Winston Churchill Sq. 4, 130 67, Prague 3, Czech Republic

nekvasim@vse.cz svatek@vse.cz labsky@vse.cz

Abstract: Diverse types of structured domain models are nowadays in use in various contexts. On the one hand there are generic models, especially domain ontologies, which are typically used in applications with artificial intelligence (reasoning) flavor; on the other hand there are more specific models that only come to use in areas like software engineering or business analysis. Furthermore, the discipline of information extraction has invented very specific knowledge models called extraction ontologies, whose purpose is to help extract and semantically annotate textual data. In this paper we present a method of authoring extraction ontologies (more specifically, their abstract constituents called presentation ontologies) via reusing different types of other knowledge models, especially domain ontologies and UML models. Our priority is to maintain consistency between extracted data and those prior models.

Keywords: information extraction, ontology, UML, business models

1 Introduction

Every model can basically be looked upon as an abstraction of reality according to a certain conceptualization. If the model can be expressed as a formal specification we can call it *ontology* according to the original T. Gruber's definition [3]. Once a model is represented as a concrete artifact, it can support communication, analysis and elaboration of the relevant aspects of the underlying domain.

It is now often assumed that the use of ontologies can bring the required flexibility to many disciplines, and we believe one of them is information extraction, and *web information extraction* (WIE) in particular. In the field of WIE it is possible to distinguish several trends in the last few years. The *wrapper-based* approach is widely adopted in today's business spheres; it is based on structural information in the HTML documents. Although it is quite reliable, it is not only domain dependent but moreover document-structure dependent and thus individual extraction tasks are not very reusable. The second, *inductive,* approach is built upon statistical learning and/or language processing. The drawback of this approach is that it requires large corpora of annotated data as a base for the learning. In addition, both of these approaches usually provide the extracted data in a form that is not enough semantically structured for further use in knowledge-based systems.

Consequently, a third approach was formed, with focus on semantic annotation of extracted data, namely with a tendency for pushing structured ontologies towards the

actual extraction process, in the role of extraction models, which can be referred to as an *extraction ontology* [1] when properly formalized. It is assumed that extraction ontologies are hand-crafted based on observation of a sample of resources; however, they are required to have a clean conceptual structure, which makes them superior to ad-hoc patterns used in the early approaches to WIE.

We think that a strict single-purpose hand-crafting of such extraction ontologies is tedious because it is very demanding to author such an ontology manually (it is often done in iterations). Moreover it can introduce inconsistencies in relation to other business models and knowledge-bases but the mutual consistency is eligible in both academic spheres and enterprise environment. In this paper we hypothesize that extraction models can be crafted via reuse of existing meta-models that are already present in the company or freely available on the internet in ontology libraries. This reuse should improve further processing of any data annotated (or extracted) using the extraction ontology in terms of other knowledge models and hopefully even lower the costs of its creation because of lessening the need of thorough prior domain-analysis.

The paper is structured as follows. Section 2 explains the nature of extraction ontologies with emphasis on their relation to other kinds of knowledge models. After that follow three sections that separately discuss the possibilities of reusing various sources for the construction of extraction ontologies; first usual domain ontologies are taken into account (Section 3), after that we consider the potential of knowledge stored in UML diagrams (Section 4), and, finally, we focus on reusing other models very common in industry, namely business process models and relational models (Section 5). The last two sections are devoted to related research (Section 6) and summary conclusions with outline of future work (Section 7).

2 Presentation Ontologies

Extraction ontologies define the concepts the instances of which are to be extracted from the documents in terms of WIE, in the sense of various attributes, their allowed values as well as higher level constraints (such as e.g. cardinality). Following the terminology coined in [5], an extraction ontology can be systematically viewed both as an information ontology and a knowledge ontology, depending on its actual content. It is possible to spot at least three layers in the structure of an extraction ontology, such as each is a kind of refinement of the previous one:

1) The incorporation of class' attributes can be represented as a set of variables and can be stored along with their datatypes. From this point of view the extraction ontology can be used as a data structure, which can come in handy while for example storing the extracted data in database.
2) The extraction ontology contains concepts that are expected to be populated with lots of instances, thus it can be viewed as *information ontology*.
3) The extraction ontology can further contain additional higher-level restrictions, such as cardinality or mutual dependency, and therefore it can be looked upon as *knowledge ontology*.

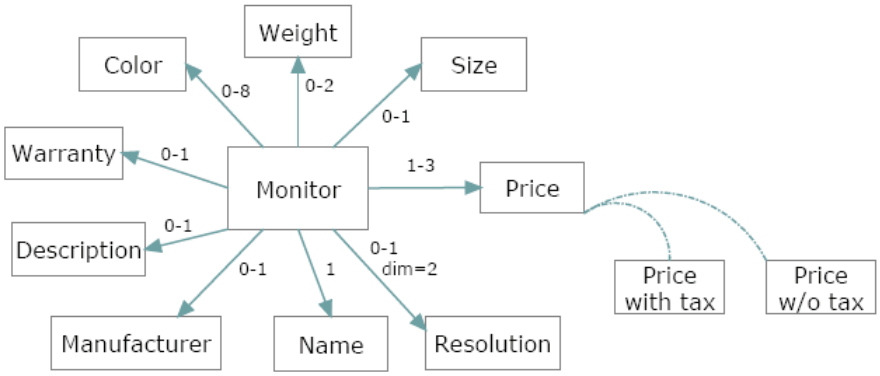

Fig. 1: High-level structure of presentation ontology for computer monitors

As such ontologies are meant to describe the *presentation* of objects within some media (on the web pages, in the notion of WIE) instead of real-world objects, it is natural to speak about *presentation ontologies.* A presentation ontology represents the fundamental part of an extraction ontology: it is the abstract part that captures the logical structure of the presentation; together with some additional low-level patterns (that enable information extraction) it forms the extraction ontology. In Fig. 1 we see the graphical depiction of a presentation ontology for the computer monitor (product catalogue) domain.

Because of different modeling principles applied while authoring presentation ontologies in contrast to other conceptual models they have a slightly different nature. Most often a presentation ontology contains a single class, referred to as the *core class* (multi-class presentation ontologies are also possible, however they are not so convenient for computational processing). The core class is then supplemented with its attributes and additional constrains. Due to this difference from other knowledge models, a transformation process is needed for their meaningful reuse. Each kind of model has its own specifics; in the remaining sections of this work we will walk through them. However generally the transformation process will consist of a few steps that are common regardless of the source of underlying knowledge. These are:

1) choose the core class C and add it to the presentation ontology
2) create its attributes in the presentation ontology
3) formulate ontological constraints (data type, cardinality) over attributes
4) create additional "WIE hooks" for each attribute: in addition to simple datatype restrictions over attributes, more extraction knowledge (e.g. regular patterns) can be added based on the content or context of known or estimated instances.

Regretfully, as the particular structure of the domain models can be very variable and the expressiveness of the source models is often high. The models thus cannot be transformed deterministically, as there are many ways of reusing a single model. Therefore the outcomes of all the rules presented below should rather be interpreted

as recommendations for an expert designer to help him/her author a suitable presentation ontology.

3 Reuse of Domain Ontologies

While the inclusion of extraction patterns is specific for the WIE setting, the abstract conceptual structure is analogous to that of domain ontologies. As the number of domain ontologies available on the semantic web increases, their reuse would be quite beneficial.

Transformation of a domain ontology expressed in the standard semantic web ontology language OWL[1] (or other high-level ontology language) into a presentation ontology will mainly amount to the transformation steps mentioned above.

The first step is to choose the core class. We so far formulated four rules that can help choose the core class:

a1) Class C that has individuals directly asserted in the domain ontology should probably not become the core class in the presentation ontology.
a2) If some property D does not have an inverse property explicitly declared, a class C in the domain of this property is more likely to become the core class than any class C_1 that figures in its range.
a3) If a class C has a minimum cardinality restriction on property D whose range is class C_1, such that C_1 does not have any restrictions on the inverse property D, then C_1 should not become the core class.
a4) If there is a chain of object properties (O_1,O_2, ...,O_n), where O_k is object property of C_k, and for every k, $1 \leq k \leq n - 1$ the range of O_k is C_{k+1}, then the classes at the ends of such a chain (i.e. C1 and Cn) are more likely to form the core class. If a class C is at the end of more such chains, it is even more suitable for becoming the core class.

When a core class is chosen, its attributes have to be created in the presentation ontology. Again these attributes can be based solely on the needs of the presentation ontology creator and on its purpose (in praxis they are either chosen ad-hoc or statistically learned from a corpus of sample data), however to maintain the semantic soundness of the resulting data, even the choice of attributes should be based on an existing knowledge model. We thus formulated another set of rules that support the population of the core class C with attributes:

b1) A datatype property may directly yield an attribute. Furthermore a datatype property D of some class C_1, together with a chain of object properties (typically *part-of* properties) (O_1,O_2, ...,O_n), where O_1 is object property of C, O_n is object property of C_1, and for every k, $1 \leq k \leq n - 1$, there is a class having both O_k and O_{k+1} as its properties, may yield an attribute. For example in a weather forecast domain for C=`WeatherForecast`,

[1] `http://www.w3.org/2004/OWL/`

C_1=Weather and D=hasTemperature, and O_1=forecastWeather having C as its domain and C_1 as its range, can yield an attribute such as 'forecastTemperature'.

b2) A set of mutually disjoint subclasses of C may yield an attribute even without a property counterpart in the source ontology. An example from a weather forecast domain would be C=Precipitation having mutually disjoint subclasses {Rain, Snow, Hailstorm, ..} can yield the attribute 'Precipitation'.

b3) A set of mutually disjoint subclasses of some C_1 such that exists a chain of object properties between C and C_1 (in the same sense as in the first rule), may yield an attribute.

b4) An object property O of a class C which object is a class C_1 that has some individuals asserted in the ontology may yield an attribute. Some possible values of such an attribute could be directly given by the asserted individuals. For example in another ontology from weather forecast domain C=Weather, C_1=Precipitation and O=hasPrecipitation with C_1 having asserted individual such as {rain, snow, hailstorm, ..} can again yield an attribute 'Precipitation'.

To support these rules we performed tests on freely available domain ontologies with varying domains, size, purpose and structure. Results are shown in Table 1. The table shows how many times each rule could be used while transforming the given ontology into a presentation ontology.

Some of these numbers can be of particular interest, for example it is not hard to spot the correlation between the use of rules b4) and a1), which is most likely due to the fact that both are based on the presence of instances. Further there are three ontologies that were only suitable for rule b3); this can be explained by the fact that they are bare taxonomies. Note that, even if there are no properties present in such taxonomies, it is still possible to derive a few attributes of the core class from them.

In summary, by these tests we verified the possibility of reusing knowledge stored in existing domain ontologies. Next we will focus on other knowledge models that are commonly present in the corporate environment.

4 Reuse of UML Models

Being the standard modeling language in software engineering, UML[2] has received wide attention not only in academic spheres, but also in industrial software development. As a consequence, UML is much better supported in terms of tools and available expertise than the semantic web languages such as OWL. The wide acceptance of UML makes it an ideal candidate for the search for existing knowledge and we think it may prove useful for authoring a presentation ontology. The drawback of trying to reuse knowledge stored in UML models is however their public unavailability, because UML diagrams are often considered the company's precious

[2] http://www.omg.org/technology/documents/formal/uml.htm

domain, ontology uri	number of classes	rules							
		a1)	a2)	a3)	a4)	b1)	b2)	b3)	b4)
weather:									
weather-ont (from Semwebcentral)	9	1	4	0	0	4	7	9	1
WeatherConcepts (from LSDIS)	19	0	9	0	6	8	8	3	0
weather-ont3 (from AgentCities)	96	7	38	7	11	0	6	21	6
publication:									
http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl	43	0	3	0	3	3	11	4	0
http://sib.deri.ie/fileadmin/documents/swportal.owl	70	0	10	0	4	6	19	3	0
http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl	61	3	4	4	12	11	25	6	1
http://ebiquity.umbc.edu/ontology/publication.owl	15	0	5	2	0	15	7	1	0
http://alignapi.gforge.inria.fr/tutorial/myOnto.owl	40	1	21	3	4	10	19	10	0
conference:									
http://sib.deri.ie/fileadmin/documents/swportal.owl	70	0	4	0	6	1	22	3	0
http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_2.owl	43	0	1	0	2	6	9	2	0
http://zeitkunst.org/bibtex/0.1/bibtex.owl	15	0	0	4	0	40	0	1	0
computer:									
http://semweb.mcdonaldbradley.com/OWL/jiva.owl	259	0	2	0	2	2	8	3	0
http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20021212 (rdfs)	7	0	20	0	0	20	0	0	0
http://morpheus.cs.umbc.edu/aks1/ontosem.owl	7596	0	0	0	0	0	0	5	0
event:									
http://www.ontotext.com/kim/kimo.rdfs	322	0	11	0	4	8	10	4	0
http://rhizomik.net/ontologies/2005/03/TVAnytimeContent.owl	376	0	0	0	0	0	0	32	0
http://smartweb.dfki.de/ontology/swinto0.3.1.rdfs	2006	0	100+	0	44	85	42	14	0
http://www2.sims.berkeley.edu/academics/courses/is202/f04/phone_project/Group8/group78.owl	1650	0	0	0	0	0	0	15	0

Table 1: Transformation rules test results

property. However, for a company such as e-shop wishing to apply information extraction so as to analyze its competitors' offers, it will always be possible to reuse its own models.

According to the Object Management Group specifications the UML diagrams can be divided into several quite different groups: structure diagrams, behavior diagrams and others (such as interaction diagrams). The UML language has been devised in order to integrate competing proposals for modeling languages in the area of software engineering. This integration effort was undertaken in order to push object-oriented design methods into industrial practice. Object oriented design is, in a way, similar to ontological engineering; in some diagrams this is quite obvious. Ontological foundations of UML diagrams are however non-trivial, and worth exploring as e.g. in [4].

There are some ongoing projects that aim to come with a standardized approach of transforming UML diagrams to common ontology languages (for example [2]), however they are often only concerned with class diagrams and do not make the best of other parts of UML (which can of course be useful too).

There are different possibilities of deriving a presentation ontology from diagrams of every group; some guidelines follow.

4.1 Structural Diagrams

The various structural diagrams such as class diagrams, component diagrams, and deployment diagrams describe static, structural constructs (e.g., classes, components or nodes artifacts).

The most common of these diagrams is the *class diagram* because it is very valuable in software engineering tasks. Luckily the concept of class in UML is very similar to the meaning of what is a class in an ontology (and yet it is not the same, for details see [4]).

For our purpose of deriving the presentation ontology we can work with the UML class in the same way as we did with the classes in domain ontologies. Some rules remain intact, but some work differently:

- A class can still directly yield a class in the presentation ontology, and a property can still directly yield an attribute.
- As there are no inverse properties in class diagrams, rule a2) can therefore be seen as even stronger.
- The multiplicity of a relation in a class model can be, with a certain degree of tolerance, translated to a cardinality restriction, and therefore used in rule a3)
- The generalization in class models is nearly equivalent to is-a hierarchy relation, and can therefore serve as support for all chain rules (a4), b1), b3)).
- As multiple inheritance is not commonly allowed in the class model, an individual cannot be instance of more than one bottom-level class and thus all subclasses of a class can be considered as mutually disjoint. This comes in handy in the rules b2) and b3).

We see that the class diagram can be used quite extensively. Similarly, the object diagram fulfils all properties of the class diagram and moreover can incorporate instances. We can use these in a similar way as we did in the case of domain ontologies:

- The instances can be used as individuals in rule a1) for rejecting a core class.
- The instances can be used as individuals in rule b4) for populating the core class with attributes.

Other diagrams in UML that can be considered as structural do not provide such extensive sources for building presentation ontologies, however some of their features may still be useful:

- *Composite structure diagrams* can, along with the information about classes and instances, provide some limited restrictions that can in very specific cases yield an axiomatic rule about existence of some attribute value.
- *Component diagrams* depict the structure of individual components, and therefore the inclusion of a class in some component can be vaguely translated as part-of relation. This relation can again serve as a basis for the chain rules (a4), b1), b3)) or simply yield an attribute as in a1).
- The *package diagram* is used to provide some logical wholes to other diagrams. The existence of a package of some entities that can be mapped to classes can again yield a part-of relation, or sometimes even an attribute. If the package contains some entities that can be mapped to attribute, it could serve some examples of that attribute's values (as a part of the additional extraction knowledge).

However not all kinds of models provide useful information in the means of authoring an extraction ontology. For example the *deployment diagram* contains detailed implementation details, which could prove useful while populating an ontology with individuals, but it is not very helpful for our purpose.

4.2 Behavioral Diagrams

The behavioral diagrams specify the dynamic, behavioral constructs such as activities, interactions, and states. The use of these constructs is not so straightforward as in the case of structural elements, yet they still can be used. One of the most used behavioral diagrams in praxis is the *state machine diagram*. The use of the state machine diagram can be supported by the fact that it describes the possible states of every object of a particular class and therefore it can tell something about the class itself:

- The described set of possible states of an entity and transitions between them can yield an attribute in presentation ontology, and moreover it can yield example values of this attribute provided by the individual states.

- Ideally the set of states of an entity described by a state machine diagram is complete, so it should provide a complete enumeration of values of an attribute (provided an attribute is yielded).
- The presence of a choice point in the state machine diagram can be contingent on the existence of a relation to some other entity. This relation can yield an attribute directly or can be used with other rules.

In UML there is also the *activity diagram*, which is basically an extended version of the state machine diagram. The nature of this extension lies in the fact that it can describe dependencies between states of different entities and thus entail some relation/s between them, which can again be further used with the former rules.

Beyond these two diagrams we can place *use case diagrams* in the behavioral group. Use cases are a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do. The key concepts associated with use cases are actors, use cases, and the subject. The subject is the system under consideration to which the use cases apply, and is often described by the class diagram. The users and any other systems that may interact with the subject are represented as actors. The use case diagrams can also be a source for our approach:

- Actors always model entities that are outside the system and therefore the actor should not yield a core class.
- Although the actor should not yield a core class, it can yield an ordinary class that can further be transformed.
- In the case of actor generalization it is possible to use rules that work with ordinary subclasses.
- In a perfect case the presence of a use case could lead to the existence of a possible state of some entity. Then we could work with the state just like when concerning state machine diagrams.

4.3 Interaction Diagrams and UML Supplements

Interactions are used in a number of different situations. They are used to get a better grip of an interaction situation for an individual designer or for a group that needs to achieve a common understanding of the situation.

Interactions are (according to OMG recommendations) also used during the more detailed design phase, where the precise inter-process communication must be set up according to formal protocols. However, as a source for authoring a presentation ontology the interaction diagrams are very limited, hence the rules for their use are vague:

- Generally the set of interactions between two entities should lead to the existence of at least one relation between them.
- The elements of interaction in individual diagrams can yield possible values of attributes, however these attributes should have been specified elsewhere.

The supplements of UML are only interesting to the extent that they can provide additional information to the extraction part of the final ontology, such as data types of values.

The above is not a complete list of models that exist in UML, however, we see little use of the others at the moment.

5 Suitability of Other Commonly Used Metamodels

UML is not the only framework of metamodels used in today's industry. There are many other various ways of formalizing specific knowledge. Amongst others, two are of particular interest: relational database models and business process models.

The *relational model* for database management is a database model based on predicate logic and set theory, so it also has many things in common with other means of specification of a domain (e.g. abundant literature about translating database content into ontology instances exists). The reuse of a relational model for building a presentation ontology can also be driven by some non-deterministic rules:

- An entity (i.e. a table) can directly yield a class and its fields (i.e. columns) can directly yield attributes.
- Foreign key references can be used as a general type of concept relation, i.e. a property of a class, and can be used in the chaining rules.
- In contrast to the reuse of domain ontologies it is necessary to distinguish the supporting tables that are incorporating the *m:n* relations. These auxiliary tables should not yield a core class (or any other class), despite they would be rated high by the original (domain-ontology) transformation rules based on property chains.
- Due to the explicit specification of primary and secondary keys it is easy to recognize an inverse property and use the rule a2) if it is not present (and it is not present commonly).
- As inverse properties are not common in the relational model, the chaining rules should be even more effective than in the reuse of domain ontologies.

The *business process model* is designed to describe a collection of activities needed to produce a specific output for a particular customer or market. It is the basic tool of the discipline of business process engineering. A process in this context is a specific ordering of work activities across time and space and it is related to the change of a state of an entity. We can again spot some useful rules:

- Every process depicts a change of a state of some entity and therefore after the fashion of state machine diagrams it should yield a possible value of an attribute, or it can even yield the attribute itself.
- The event element and the choice element should express a relation to some other entity and therefore could yield an attribute given by this relation.

- A set of processes delimited by interactions with external systems (and ideally even enclosed in a pool) should describe an entity, which can then possibly lead to a class in presentation ontology.

6 Related Work

One of the most similar projects to ours was presented in [2]; it is concerned with the transformation of UML class diagrams and relational models to domain ontologies (i.e. across the types of models we consider as prior). The relation of ontologies and UML models is also important in [4].

On the other hand, the idea of extraction ontologies has first been coined by D. Embley's group, e.g. in [1]. As far as the relation between domain and extraction ontologies is concerned, they assume the opposite direction: transformation of extraction ontologies (along with extracted instances) aiming at complying with existing domain ontologies. The problem of authoring the extraction ontology itself is therefore not addressed.

For completeness, let us also mention our ongoing Ex project [7], running in parallel, and focused on endowing information extraction technology with synergistic exploitation of multiple resources, extraction ontology having the prominent role.

Our work has close affinity to projects addressing ontology selection, such as OntoSelect[3] or Watson[4]; as successful search for ontologies is a necessary prerequisite for our approach, we plan to extensively use their functionality.

7 Conclusion and Future Work

We have shown how various knowledge models can be reused in favor of the semantically driven approach to web information extraction. The pros of this reuse are in the fact that the final knowledge model for information extraction is semantically well structured and that the annotation of extracted data is consistent with existing models as much as possible. The cons on the other hand are that the business models are not commonly freely available and so the enterprise has to rely on its own models. Furthermore, there is no standard language for specifying the presentation and extraction ontologies; we use a proprietary format at the moment.

Finally, although it is possible to reuse knowledge stored in any one of the different models, we think it would also be possible to reuse knowledge that is dispersed across multiple models. For example, the role of the state entity in different models could be of particular interest. This is one of interesting directions for further work.

[3] http://olp.dfki.de/ontoselect
[4] http://watson.kmi.open.ac.uk

Acknowledgement
The research leading to this paper was supported by the European Commission under contract FP6-027026, Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content - K-Space.

Reference

1. Embley, D. W., Tao, C., and Liddle, S. W.: Automatically extracting ontologically specified data from HTML tables of unknown structure. In Proc. ER ’02, pages 322–337, 2002
2. Falkovych, K., Sabou, M., and Stuckenschmidt, H.: UML for the Semantic Web: Transformation-Based Approaches. In Knowledge Transformation for the Semantic Web, p. 92-106. IOS Press, 2003
3. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993
4. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, Telematica Instituut Fundamental Research Series No. 15, 2005, ISBN 90-75176-81-3
5. Heijst, van G., Schreiber, G., Wielinga, B.: Using Explicit Ontologies in KBS development, Int. J. Human-Computer Studies, Volume 46, 1997, 183-292.
6. Labský, M., Nekvasil, M., Svátek, V.: Towards Web Information Extraction using Extraction Ontologies and (Indirectly) Domain Ontologies. Whistler 18.10.2007 – 21.10.2007. In: K-CAP'07. New York : ACM, 2007, s. 201–202. ISBN 978-1-59593-643-1.
7. Labský, M., Svátek, V., Nekvasil, M., and Rak, D.: Information extraction using extraction ontologies. In Proc. PriCKL’07, ECML/PKDD Workshop on Prior Conceptual Knowledge in Machine Learning and Knowledge Discovery, Warsaw, Poland, 2007