

Lessons in building OWL Ontology driven applications: OCHWIZ – an Occupational Health Application

Jay (Subbarao) Kola MBBS. (PhD), Bill Wheeldin MB ChB, Alan Rector MD, PhD.

School of Computer Science, University of Manchester, Manchester M13 9PL, UK

Abstract

Recent advances in semantic technologies have seen the Web Ontology Language(OWL)[1]take a prominent place in knowledge modelling and representation. We have attempted to combine these recent advances in semantic technologies with knowledge-based applications. In this paper, we describe how we used an OWL Ontology as a knowledge base to drive an Occupational Health Application. We take an engineering perspective on the possibility of building an OWL ontology based application and discuss the relative merits of this approach as opposed to tabular models. In particular we focus on advantages in authoring such a knowledge base and also in maintaining it. We then discuss some of the implementation challenges in engineering such applications and also possible design considerations.

Introduction

Medical Informatics has a long history of developing knowledge-based applications. In implementing knowledge based applications, it has been widely recognised that separating the knowledge content from the application logic allows reuse of the knowledge content and reduces the overhead in extending and maintaining knowledge content. As relational databases grew in popularity, much of the knowledge in medical applications was held as relational tables. More recently, knowledge tends to be captured in spreadsheets as well as databases. However, these approaches have their demerits:

- It is difficult for a clinical expert to capture knowledge.
- It is hard to maintain and extend the knowledge content.
- It is hard to predict the effect of a change in the knowledge base on the application.

We use an alternative approach to capturing clinical knowledge as an OWL ontology. We couple the ontology to a prototype medical

application, OCHWIZ (Occupational Health Wizard) to demonstrate the advantages of this approach. OWL with its formal logic underpinning offers the following advantages, which are discussed further in the paper:

- An easy and intuitive way to capture knowledge.
- The ability to maintain and extend a knowledge corpus and ability to track effects of changing a section of the knowledge base.
- The ability to infer knowledge with the help of the first order logic services built into a variant of OWL (OWL-DL).

We recognise the existence of other knowledge modelling architectures like the frames architecture and a comparison of the OWL – DL approach to the frame approach is not within the scope of this paper.

Ontologies as a Knowledge Representation Alternative

Medicine as a domain is rich in semantic relationships between entities and it often needs expressivity in modelling knowledge. This expressivity is not often easily provided by tabular models. Web Ontology Language (OWL) the W3C recommended standard for representing semantic links and knowledge is best suited to capture these complex relationships. Knowledge modelling requires specialised modelling environment and the most popular knowledge-modelling environment is Protégé [2]. Protégé makes it easy to capture the complex relations between medical entities in an intuitive way. Protégé OWL [3] is an extension to the Protégé environment that allows knowledge modellers to capture knowledge in the OWL framework, which lends a formal logical basis to the model. The particular group of first order logic that powers one variant of OWL is called Description Logic (DL)[4] and this variant of OWL is called OWL-DL. Integration of Description Logics into OWL allows encapsulation of rules (in the form of

inferences, sub-sumption, etc) in the OWL ontology which are discussed later on. At this point, it is important to understand the general architecture of a DL based OWL ontology.

Features of a OWL-DL based Ontology

A formal ontology has a vocabulary that is defined in terms of “concepts” and “roles” which define the relationships between the concepts.

- *Primitive Concepts*¹: These are the atomic building blocks of an ontology. They are often grouped together as hierarchies.
- *Properties*²: These represent roles that are then used to define relationships between primitive concepts.
- *Defined Concepts*³: These are complex descriptions built using primitive concepts and properties.
- *Restrictions*⁴: These are property-concept pairs qualified by logical attributes like “some” and “only”. E.g.: ‘has_cause’ some ‘Arsenic’
- *Axioms*: These are statements or assertions about named concepts in the ontology. For example the statement that a Concept A is related to a Concept B via a property “is_child_of” is an axiom. e.g. $A \rightarrow \text{“is_child_of”} \rightarrow B$
- *Reasoner*: A service built into DL based ontologies, which allows “reasoning” over an ontology. Reasoning usually consists of determining if a concept or description is a child of another (sub-sumption)⁵. It also checks axioms and descriptions in the ontology for logical consistency.

Prototype Application: OCHWIZ

Our prototype application was designed as an aide to diagnosis in an Occupational Health Clinic setting. The application suggests possible causes and industries associated with a given clinical finding. It also tries to infer further possible diseases or findings that might be associated with the inferred causes and industries. For example, given a finding of ‘Pigmentation’, the application suggests causes of ‘Pigmentation’ and industries that might have exposure to the causes. A screenshot of OCHWIZ showing the causes of ‘Black Pigmentation’ and the industries with exposure to these causes is shown in Figure 1.

¹ Sometimes referred to as ‘Classes’.

² Also referred to as ‘slots’, ‘attributes’ or ‘roles’.

³ Also called ‘Defined Classes’.

⁴ Also called “Descriptions”.

⁵ Also test for equivalence.

To provide this functionality, the application needs knowledge about occupational diseases, their causes and industries associated with them. Entities in the occupational health setting were grouped under the following categories:

- Clinical Findings
- Industries
- Causes
- Work Roles
- Work Tasks

The relationships between these entities are shown in the table below.

Entity	Relationship	Entity
Clinical Finding	has_cause	Cause
Industry	has_exposure_to	Cause
Work_Task	is_task_in	Industry
Work_Role	is_role_in	Task

Table 1: Entity relationships

This seems simple enough to represent this in a table as shown in Table 1. However, this table does not adequately capture the complexity of the entities and relationships. A more appropriate classification of the entities is shown in Fig 2.

We present an example to highlight the advantages of the OWL-DL approach. For example, we know *Pigmentation* is caused by *Arsenic* and *Black Pigmentation* is caused by *Coal Tar*. *Coal Tar* actually consists of; *Asphalt* and *Pitch*. If we wanted to relate this in tables, we would need to create a table of the following sort.

Clinical Finding	Cause
<i>Pigmentation</i>	<i>Arsenic</i>
<i>Black Pigmentation</i>	<i>Coal Tar Constituents</i>
<i>Black Pigmentation</i>	<i>Asphalt</i>
<i>Black Pigmentation</i>	<i>Pitch</i>

Table 2 : Finding – Causes relationships

If we say *Glass product manufacturing* and *Electronic product manufacturing* have exposure to *Arsenic* and *Construction Industry* has exposure to *Coal Tar Constituents*; we need to relate all these entities in another table like:

Cause	Industry
<i>Arsenic</i>	<i>Glass product manufacturing</i>
<i>Arsenic</i>	<i>Electronic product manufacturing</i>
<i>Coal Tar Constituents</i>	<i>Construction Industry</i>
<i>Asphalt</i>	<i>Construction Industry</i>
<i>Pitch</i>	<i>Construction Industry</i>

Table 3: Causes – Industry relationships

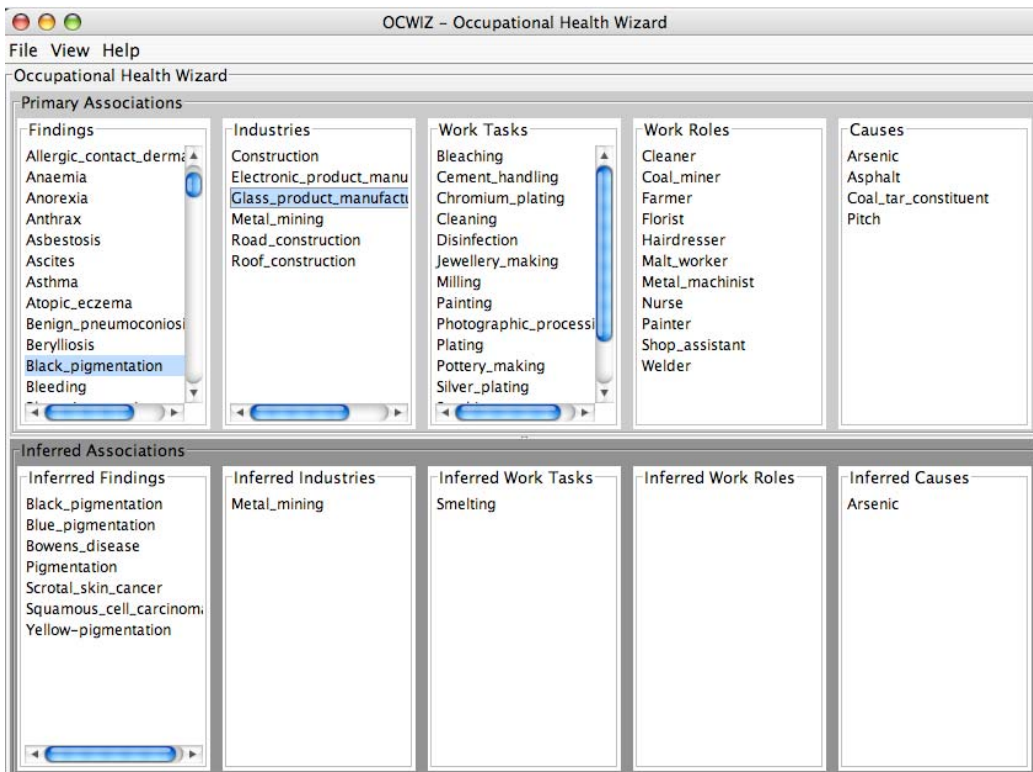


Fig 1 : Screen capture of OCHWIZ application

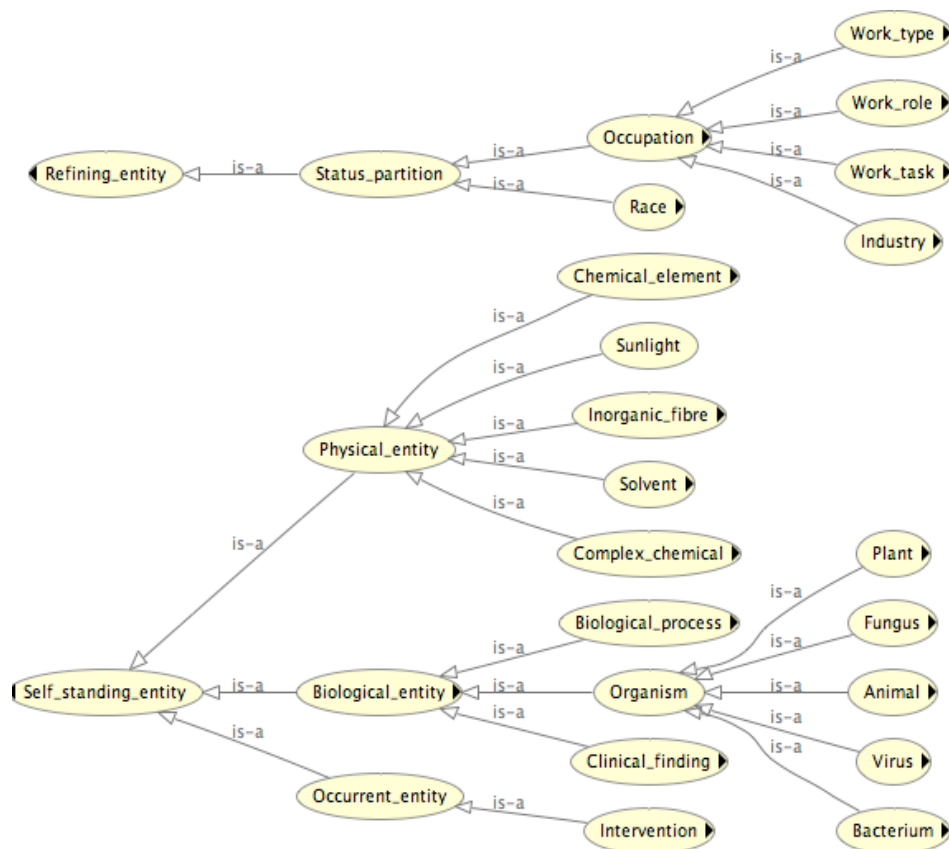


Fig 2 : Entity hierarchy

Table 3 repeats information that *Construction Industry* has exposure to *Asphalt* and *Pitch*, which is already implied by saying *Construction Industry* has exposure to *Coal Tar Constituents*. We can already see that there is needless repetition of information in the tables. In reality we might want to say that *Pigmentation* can also be of other types: *Blue Pigmentation* and *Yellow Pigmentation*. We then say *Blue pigmentation* has specific cause *Silver Salts* and *Yellow Pigmentation* has specific cause *Dinitrophenol* in addition to the general causes of *Pigmentation*. This clearly means extending the existing table to include these new cases as shown in Table 4.

Clinical Finding	Cause
<i>Pigmentation</i>	<i>Arsenic</i>
<i>Black Pigmentation</i>	<i>Coal Tar Constituents</i>
<i>Black Pigmentation</i>	<i>Asphalt</i>
<i>Black Pigmentation</i>	<i>Pitch</i>
<i>Black Pigmentation</i>	<i>Arsenic</i>
<i>Blue Pigmentation</i>	<i>Silver Salts</i>
<i>Blue Pigmentation</i>	<i>Arsenic</i>
<i>Yellow Pigmentation</i>	<i>Dinitrophenol</i>

Table 4: Pigmentation types - causes

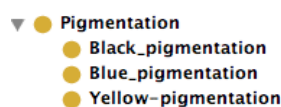
We might now also want to say *Construction Industry* can be of different types; *Roof Construction* and *Road Construction*. This means rewriting all the tables that we have created to include these additional industries and the process is laborious and cumbersome.

OWL Modelling

We now take the same the same information and represent it in an OWL ontology that drives the OCHWIZ application. In implementation, we use some of the built in features of OWL-DL as described earlier and discuss each under the following sections; primitive hierarchies, defined concepts and reasoner inferences. We do not discuss other features of OWL like transitivity, disjoints, etc in this paper and we refer the reader to general literature on OWL DL technologies [5,6].

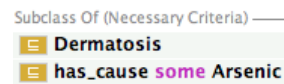
Concept Hierarchies

The information presented in in the tables above can be represented as primitive concept hierarchies in an OWL Ontology as follows.



If causes of *Pigmentation* were listed on *Pigmentation* they would get inherited down to

Black, Blue and Yellow Pigmentations.



The specific causes of *Black, Blue and Yellow Pigmentations* can be asserted on the entities separately. The description of *Black Pigmentation* in the ontology would look like:



Note that *Black Pigmentation* has inherited causes of *Pigmentation* as shown under the “Inherited descriptions” in the figure above. Also if *Asphalt* and *Pitch* are listed under *Coal Tar Constituents*, an axiom that *Coal Tar Constituents* cause *Black Pigmentation* will automatically infer *Asphalt* and *Pitch* as causes of *Black Pigmentation*.



We represent types of *Construction Industry* in a similar fashion. Hierarchies are intuitive to model as opposed to repetitive information in tables. An alternative approach using tables will be to capture the hierarchy information in a separate table as ‘Parent – Child’ relationship and access this information at run time. However this is neither as elegant nor as clear as creating hierarchies in an OWL ontology. OCHWIZ uses this hierarchy to infer causes of *Pigmentation* and *Black Pigmentation* as shown in Figures 3 and 4 respectively.

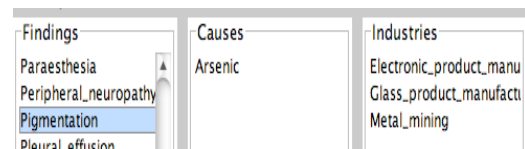


Fig 3: Inferences for Pigmentation

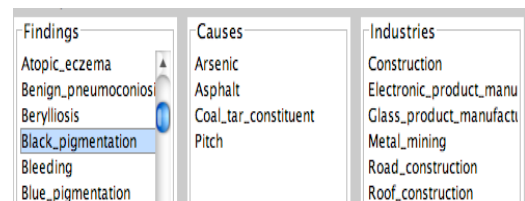


Fig 4: Inferences for Black Pigmentation

Defined concepts

As mentioned above, OWL *defined concepts* use combinations of properties and primitive concepts to create complex entities. We can pass a description of a *defined concept* to the reasoner and then ask interesting questions of it like; is this *defined concept* more general or more specific than some concepts in the ontology. A use case for this in our application was to infer industries with exposure to a given chemical substance. In the above example, we pass all substances that can cause *Black Pigmentation* to the reasoner and ask for industries with exposure to these causes. The result of the query gives us industries associated with *Black Pigmentation* and is shown in Figure 4.

This corresponds with the information we started with except for *Metal Mining*, which we did not directly associate with *Black Pigmentation*. However, the reasoner inferred that *Metal Mining* is associated with *Black Pigmentation* because we stated elsewhere that *Smelting* is a *Work Task* which has exposure to *Arsenic*. We also asserted that *Metal Mining* has a *work task* that is *Smelting*. This causes the reasoner to infer that *Metal Mining* is in fact associated with *Black Pigmentation*. The description of the defined class “*Probe_industry_has_exposure_to_Coal_tar_constituent_or_Arsenic*” looks like this :

Equivalent Class (Necessary & Sufficient Criteria)

```
Industry
has_exposure_to some (Coal_tar_constituent or Arsenic)
```

Equivalent Class (Necessary & Sufficient Criteria)

```
Industry
has_work_task some (has_exposure_to some (Coal_tar_constituent or Arsenic))
```

Though this at first strikes as a round about way of capturing information in the ontology, it in fact ensures that knowledge goes in the right place without being repeating at various levels as in the case of tabular architectures. For example a *Cleaner* might have a work task *Cleaning* might have exposure to *Cleaning Agents* but *Cleaning* is a task common to many industries. So the substances a *Cleaner* gets exposed to depends on the industry s/he works in and also to *Cleaning Agents*. In the OCHWIZ application, we in fact use a generic reasoner query to infer all roles, tasks and industries with exposure to a substance. From the roles and tasks, we infer the Industry involved by a further query to the reasoner. We display these inferred work roles tasks separately under the primary associations in Figure 1. Note that *Black Pigmentation* is

inferred to have possible associated findings like *Bowen's disease* and cancer.

Knowledge base extension and maintenance

The reasoner also allows us to group unrelated primitive concepts by using a concept description. For example, we might want to extend our pigmentation example to say that *Black Pigmentation* was caused by any substances that has a constituent ‘*Polycyclic aromatic hydrocarbon*’ (PAH). We then create a defined class “*Polycyclic_aromatic_hydrocarbon_containing_substance*” with the following definition.

Equivalent Class (Necessary & Sufficient Criteria)

```
Complex_chemical
has_constituent some Polycyclic_aromatic_hydrocarbon
```

We also add another axiom on *Coal_Tar_Constituents* that it has constituent “*Polycyclic aromatic hydrocarbon*”(PAH).

Subclass Of (Necessary Criteria)

```
Complex_chemical
has_constituent some Polycyclic_aromatic_hydrocarbon
```

If we query the reasoner for causes of *Black Pigmentation* using the new definition and it will return :

```
Polycyclic_aromatic_hydrocarbon_containing_substance
└─ Coal_tar_constituent
   └─ Asphalt
   └─ Pitch
   └─ Tobacco_smoke
```

Notice that *Tobacco Smoke* has been inferred to be a cause of *Black Pigmentation* with our new definition. This has been inferred because *Tobacco Smoke* has been asserted to contain *PAHs* elsewhere in the ontology. The clinical modeller now has a choice to accept or reject this inference. If this inference is wrong, then it shows the statement *PAHs* cause *Black Pigmentation* is clearly wrong and this needs to be changed. The reasoner can be used in this fashion to maintain consistency and accuracy of the OWL knowledge base. On the other hand, this new knowledge can be used to infer all diseases that are caused by “*Polycyclic aromatic hydrocarbons*”. A OWL description of the form :

Equivalent Class (Necessary & Sufficient Criteria)

```
Biological_entity
has_cause some (has_constituent some Polycyclic_aromatic_hydrocarbon)
```

returns the following disease caused :

- ▼ ☰ Probe_disease_caused_by_polycyclic_aromatic_hydrocarbons
 - Black_pigmentation
 - Lung_cancer
 - ▼ ● Squamous_cell_carcinoma
 - Bowens_disease
 - Scrotal_skin_cancer

A new addition to this list is *Squamous Cell Carcinoma*. This is explained by the fact that *Coal_tar_constituent* is one of the causes of *Squamous Cell Carcinoma* asserted in the ontology. The reasoner then infers the relationship between *Coal Tar Constituents* and *PAHs* and classifies *Squamous Cell Carcinoma* under Disease caused by *PAHs*. *Lung cancer* was inferred cause *Tobacco Smoke* was asserted to be cause of *Lung Cancer*. The ability to create groups of concepts dynamically based on an arbitrary OWL description without actually creating a named concept in the OWL ontology ensures that the application is adaptable to new use cases.

The inference power of the reasoner bundled in OWL DL is extremely powerful and is used in the OCHWIZ application to infer *Findings* from *Causes*, *Industries* from *Causes*, *Industries* from *Work Roles*, etc. The general mechanism is similar to how causes and industries associated with *Pigmentation* and *Black Pigmentation* were derived above.

Advantages of the OWL DL approach

In the above sections, we have demonstrated that the use of OWL DL's inbuilt features, provides a substantial advantage in implementing and maintaining Knowledge based applications. The general design principles used in the ontology that drives OCHWIZ are applicable to any other medical or non-medical applications. In fact, the implementation architecture can be made as generic as possible since most of the power of the application comes from the ontology design and the DL reasoner built into OWL.

- *Hierarchies*: It is worth highlighting that the hierarchies of concepts offer a clear advantage over a flat list of concepts offered by tabular models.
- *Defined concepts*: The use of OWL descriptions and class axioms allows one to capture of information in an ontology in the same way that alternative architectures allow.
- *DL Reasoner*: However, it is the logic-based services provided under the hood by OWL DL that lends power to the knowledge back end. As shown above, the

use of defined classes, allows the reasoner to infer implicit knowledge in the ontology without it having to be explicitly stated at each stage.

- *Modelling*: Capturing knowledge in the form of an ontology is a very intuitive process and can be performed by domain experts without knowledge of technologies unlike databases.
- *Reuse*: OWL is the W3C standard for representing knowledge and semantics and allows reuse of clinical content captured in ontology. There is a growing number of standard biomedical resources like the *NCI-Metathesaurus* [7], the *Gene Ontology* [8], etc available in OWL format and these many of these can be reused in modelling applications.

Limitations of the OWL DL approach

Though OWL DL based ontologies offer an elegant and powerful alternative to other knowledge modelling architectures, they are not entirely without problems. In the implementation of OCHWIZ application, we encountered some issues with the DL reasoner bundled with OWL-DL. These issues are primarily related to classification times. Classification time in simple words is the time taken by the DL reasoner to reason over an ontology. The following factors were primarily seen to influence classification times.

- *Use of reciprocal relationships*: Asserting that *Black Pigmentation* has_cause *Coal_tar* and *Coal_tar* is_cause_of *Black Pigmentation* increases the classification time. This is increase in classification time is directly proportional to the number of reciprocal relationships used in the ontology. In a well-designed ontology, often there is no need to assert reciprocal relationships and often the inferences of the reasoner in one direction can be programmatically reversed at application run time. The use of *OWL Individuals* instead of *OWL Concepts* to infer reciprocal relationships is another approach to addressing this issue.
- *Use of disjunctions*: The use of a large chain of concepts in the form *Disease* has_cause (*A or B or C or D or E*) slows down the reasoner. The use of an OWL description of this kind is very infrequent in a well-designed ontology and can be easily avoided.
- *Scaling issues*: Often classification times of ontologies are related to the complexity of the OWL-DL constructs used in the

ontology rather than the size of the ontology.

Many of these issues are areas of active research [4,9,10]. In our experience, the gaps in the knowledge model that arise from such modelling compromises can easily be plugged by providing supplemental software methods between the OWL ontology and the application.

Software Bridges :

In building the OCHWIZ application, we had to implement some software methods to address some of the above-mentioned issues with the DL reasoner. We believe that there is always more than one way to address these issues and so we only present the approach we adopted. In particular to address the limitation imposed by OWL in creating “*reciprocal relationships*”, we created two versions of the ontology. These ontologies were ‘*mirror images*’ of each other in the sense that the directions of restrictions in the ontologies were in exactly opposite directions. For example, in one ontology we would have the following restriction on “*Black Pigmentation*”;

Subclass Of (Necessary Criteria) —————
 [E] Pigmentation
 [E] has_cause some Coal_tar_constituent

and the mirror ontology would have the following restriction on “*Coal Tar Constituent*”;

Subclass Of (Necessary Criteria) —————
 [E] Complex_chemical
 [E] is_cause_of some Black_pigmentation

This helps us ask two questions of the reasoner ; ‘*What are the causes of Black Pigmentation?*’ and ‘*What are the diseases caused by ‘Coal Tar Constituents?’*. This is done only because OCHWIZ infers associations of a Clinical Finding. For example, when a user selects “Black Pigmentation”, the application generates the associated findings as shown in Figure 5.

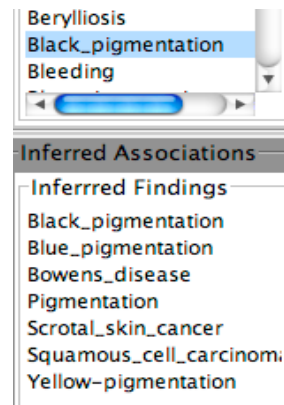


Figure 5 : Inferred associated findings of Black Pigmentation

These inferred findings alert the clinician to possible clinical findings that might be associated with ‘*Black Pigmentation*’ and are caused by agents causing *Black Pigmentation*. These inferred findings are generated by asking the reasoner the question:

Subclass Of (Necessary Criteria) —————
 [E] Clinical_finding
 [E] has_cause some (Causative_agent that (is_cause_of some Black_pigmentation))

To answer this question, the ontology needs to have restrictions in the form “*has_cause some Causative_agent*” and “*is_cause_of some Clinical_finding*”. This is a ‘*reciprocal relationship*’ as described above and it effectively stops the reasoner when used in more than a few cases. By using ‘*mirror ontologies*’ we ask these questions of separate ontologies and combine these answers on the application side. A cleaner option is to use “*Individuals*” and assert relationships between the individuals. For example assertions on “*Coal_tar_constituent*” individual;

Asserted relationships —————
 [E] is_exposed_by Construction_ind
 [E] is_cause_of Squamous_cell_carcinoma_ind
 [E] is_cause_of Black_pigmentation_ind

will allow the us to ask of the reasoner, queries in the opposite direction. So a “*Black Pigmentation Individual*” is inferred to “*has_cause some Coal_tar_constituent Individual*”. This avoids the problem of creating ‘*mirror ontologies*’ but adds the step of creating individuals for each class in the ontology at application run time. Using either of these approaches did not have significant differences on the performance of the application.

Conclusion

We have demonstrated that OWL-DL ontologies offer an intuitive and powerful architecture to capture knowledge in knowledge based systems. The flexibility and richness of OWL is especially suited to capturing the rich relationships between concepts in medicine. Use of concept hierarchies offers significant advantage over tabular knowledge models. DL based OWL descriptions in an ontology allows dynamic creation of hierarchies making applications adaptable. The DL reasoner that is part of OWL-DL allows various forms of inferences over the knowledge captured in the ontology. These inferences allow an OWL-DL based application to access implicit knowledge in a knowledge base even when it has not been stated explicitly. The DL reasoner built into OWL-DL also allows intelligent applications to be built on top of an OWL-DL ontology with minimal programming effort and extends the power of such applications. Performance issues with the DL reasoner are a matter of concern but in our experience good ontology design and application implementation results in performances comparable to other application architectures. A well-designed OWL-DL ontology as a knowledge back-end for an application has great potential as an alternative to traditional knowledge driven application architectures especially in the field of medicine.

References

1. W3C Consortium OWL Specification (<http://www.w3.org/2004/OWL/>)
2. Protégé Ontology Editor (<http://protege.stanford.edu/>)
3. Horridge, M., N. Drummond, et al. (2004, 2004). "Building Ontologies with Protege-OWL Plugin." 2004, from <http://www.co-ode.org/resources/>.
4. Baader F, Calvanese D, McGuinness DL, Nardi D, and Patel-Schneider PF; The Description Logic Handbook: Theory, Implementation and Application; Cambridge University Press, 2002.
5. Knublauch, H., O. Dameron, et al. (2004). Weaving the biomedical semantic web with the protege owl plugin. Proceedings of KR-Med-2004: International Workshop on Formal Biomedical Knowledge Representation.
6. Horridge, M., H. Knublauch, et al. (2004). A practical guide to building OWL ontologies using the Protege-OWL plugin and CO-ODE tools, University of Manchester, CO-ODE project: 118.
7. NCI Center for Bioinformatics caCORE: <http://ncicb.nci.nih.gov/NCICB/core>
8. The Gene Ontology Consortium (2000). "Gene Ontology: tool for the unification of biology." *Nature Genetics* 25: 25-29.
9. Parsia P, Halaschek-Wiener C and Evren Sirin Towards incremental reasoning through updates in owl-dl in Reasoning on the web (row); 2006 (<http://www.mindswap.org/papers/2006/inclass.pdf>).
10. Parsia B, and Evren Sirin and Kalyanpur A: Debugging owl ontologies in The 14th international world wide web conference (www2005), Chiba, Japan, May 2005.