OntoMove: A Knowledge Based Framework For Semantic Requirement Profiling and Resource Acquisition[*]

Mehul Bhatt, Wenny Rahayu, Sury Prakash Soni and Carlo Wouters
Data Engineering and Knowledge Management Group
Department of Computer Science, La Trobe University.
{mbhatt, wenny, spsoni, cewouter}@cs.latrobe.edu.au

Abstract

The use of formal knowledge representation structures, or ontologies, has found immense applicability for the interoperability of software systems, e.g., alignment of software and business process models. Toward the management of such knowledge structures, an important foundational problem is that of ontology reuse – it is uncommon for new applications or for different components within one application to use an already available ontology in its entirety. Depending on component specific requirements, typical re-usages are restricted to refined versions of an existing ontology, with the refinement taking the form of a contraction of the knowledge contained therein. Furthermore, when the ontology is used to ascribe meaning to independently existing 'resources' (e.g., documents collections, source code, software manuals, process template repositories) by way of meta-data, there exists a direct mapping between different views/re-uses of an ontology and their respective semantic scopes within an annotated resource repository thereby leading to the concept of view/reuse dependent resource retrieval. We implement a framework that supports ontology reuse by way of a requirement driven sub-ontology extraction methodology. Additionally, based on this concept of a sub-ontology, we implement the idea of a user/component 'requirement profile' consisting of semantic descriptions of the user's interest within a 'resource repository' that has been annotated with semantic types from the ontology under consideration. A generic framework that implements these ideas and its application in the domains of Medical Information Retrieval systems and Business Process Management Systems (BPMS) is presented.

1 Introduction

The new era of semantic web has enabled users to extract semantically relevant data from the web through the use of a domain-based shared and formal knowledge representation structure, referred to as an 'Ontology'. Formal ontology-based representation schemes have found extensive applicability in domains as diverse as Information Retrieval Systems in Medicine and Bioinformatics, e-Health Information Systems, Business Process Management Systems (BPMS) etc. The common element underlying all these domains is the use of, sometimes implicitly, an ontology that defines the uniform structure of concepts and relationships in the domain in one form or another. Typical uses of an ontology in the aforementioned domains include: (a) Semantic annotation and categorisation of resources (or other semi/un-structured data sources) in a repository [13]. (b) Interoperability across systems and even within sub-systems when the overall system is distributed in nature. Here, ontologies facilitate a common, consensual understanding between distributed sub-systems and bridge the gap between businesses and software engineering process (e.g., through a business process ontology), and similarly, to facilitate the alignment of different software process models in software engineering (e.g., through a software process ontology [12]). In general, underlying the aforementioned application domains are the closely related themes of *interoperability* of large-scale (component-based) software systems and the *efficient and semantic retrieval* of their resource requirements. Within the context of the class of these applications, the following problem is identifiable at two distinct levels: (1) *Foundational (Domain Independent) Level*: It is uncommon for new applications within an existing domain to use an already available ontology in its entirety. In most cases, depending on the needs, a new application will use refined version(s) of the existing ontology, with the refinement taking the form of either an *expansion* (i.e., addition of more concepts and relationships), *contraction* (i.e., extraction of a sub-ontology that is consistent and independent in itself) or both if previously held domain knowledge is invalidated and new knowledge is added at the same time [7]. The reason this problem is foundational or general is that it can be solved on the basis of an abstract notion of an ontology, without resorting to any particular domain-specific details. (2) *Application Level*: As the size of an ontology grows bigger, so does (not necessarily as a consequence) the size of the content or data-sources that are annotated using the

[*]We acknowledge the financial and infrastructural support of the Victorian Partnership for Advanced Computing (VPAC) toward this research. VPAC, Australia – http://www.vpac.org/

concepts and relationships in the ontology. Likewise, when only partial views (sub-ontologies) of an existing ontology are used, the semantic range of the sub-ontology in the overall annotated dataset is significantly narrowed; a ramification, as will be explained shortly, which can be neatly exploited in novel ways for semantic information retrieval.

We propose OntoMove, a knowledge or ontology based approach that addresses both, the foundational and application level problems and offers a unified framework that is applicable across diverse application domains. At the foundational level, OntoMove is capable of extracting sub-ontologies from an existing domain-ontology on the basis of user-specified requirements. By using our sub-ontology extraction methodology, the user applications or software components will be able to reuse and share common concepts and properties of an existing ontology, rather than creating a brand new ontology or use the existing (base) ontology parts of which remain unutilised. At the implementation level, we propose a generic semantic information retrieval methodology that is grounded on the foundational concept of a sub-ontology. We demonstrate the manner in which a sub-ontology, obtained on the basis of user specified requirements, can be used to narrow down the scope the users/component's interest in the resource that is annotated using the given ontology. We demonstrate the usability of our proto-typical system in the context of a Medical Information Retrieval system (with every user specific need as a new component) that involves the use of a massive data-set, namely the Medical Therapeutic Guidelines (TG) [28], as the annotated resource. Another application scenario that is demonstrated is that of a Business Process Management System, focusing on aspects relevant to a process template manager (i.e., a user controlled component that is instantiated several times) and a set of process templates, which is the annotated resource that is used by the template manager. In both scenarios, proposed solutions to both aspects of the problem are illustrated. The novelty of our system is that it generates user/component/application specific tailored sub-ontologies as well as the set of data-sources that are relevant to the tailored requirement of every new component, into one encapsulated module. Furthermore, it must be noted that OntoMove utilises semantic web standards RDF(S) and OWL, as well as domain-specific standard[s] and vocabularies; this ensures maximum generality and wide acceptability of the framework within domain- specific zones.

The rest of the paper is organised as follow: section 2 briefly discusses the concept of an ontology as a formal knowledge representation structure. Some of the foundational issues in ontological research are highlighted from a software engineering viewpoint and related work analysed and compared in that context with our proposed framework, which is then elaborated in section 3. Section 4 presents illustrative scenarios aimed at highlighting the diversity of the potential applications of the proposed OntoMove framework. Finally, we discuss the future direction of our work in this area in section 5.

2 Knowledge-Based Approaches in Software Engineering

Ontology - A Formal Knowledge Representation Structure

Ontologies play a pivotal role by providing a source of shared and precisely defined terms that can be used as (meta-data) resource annotations in order to make those resources more accessible to automated agents. Although there are inherent distinctions between a taxonomy and a ontology, ontologies as typically used on the semantic web and software engineering applications consist of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine understanding [8]. The Web Ontology Language (OWL), is a knowledge representing scheme designed specifically for use on the semantic web; it exploits existing web standards (XML and RDF), adding the familiar ontological primitives of object and frame based systems, and the formal rigor of a very expressive Description Logic (DL) [9]. The Knowledge Base (KB) of a typical DL based system comprises of two components, the *TBox* and the *ABox*. The TBox introduces the terminology, i.e., the vocabulary of an application domain, while the ABox contains assertions about named individuals in terms of this vocabulary. The logical (DL) basis of the OWL language means that reasoning services can be provided in order to make OWL described resources more accessible to automated processes thereby allowing one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. From a formal point of view, OWL can be seen to be equivalent to a very expressive DL, with an OWL ontology corresponding to a DL 'terminology' (Tbox) whereas instance data pertaining to the ontology making up the 'assertions' or Abox.

Ontology Management

The past few years have witnessed a range of application and research oriented results in the area of ontology management and processing. Most of these results encompass areas such as ontology evolution, ontology editing and alignment, ontology merging etc. We will describe here some of the existing works and outline how our proposed OntoMove differs from those existing systems.

The work in [14] covers the area of ontology reuse and evolution in the context of ontology management within a distributed system environment. Their proposed method allows the creation of a new ontology by reusing an existing ontology, whilst taking into consideration ontology evolution and integration given the fact that the created ontologies are distributed on many different sites. Similar work by the same authors in [15] also describes the notion of 'ontology registration' in order to provide means to locate existing ontologies for reuse. While this work has addressed many important issues in a distributed ontology environment, there are a few areas that have not been fully addressed. Firstly, the proposed method lacks a proper technique to optimise the created (reused) ontologies – although an algorithm to check the validity of the ontology is proposed, the method does not include a mechanism to derive the most optimum ontology. Note that optimality involves several criteria revolving around the size of the resulting ontology, e.g., semantic simplicity and/or the minimisation of redundant content in the resulting ontology. Furthermore, the method lacks the all-important notion of a 'user/application requirement', which is indeed the essence of ontology reuse per se. Secondly, the proposed technique only focuses on the extraction of a new ontology from an existing ontology, without consideration for retrieving other artefacts or resources that might be linked or annotated against the existing ontology – any refinement of the structure or extent of the ontology bears a direct relation to its semantic scope within whatever resources have been described using that ontology. Fianlly, the work in [6] is related to the foundational problem being addressed in this research. It focuses on reasoning and improvised consistency checking for a massive ABOX (instance part) by its partitioning into a 'summary Abox' that is free of redundancies that are present in the original Abox. The main motivation behind their work is the optimisation of space and time factors involved in 'reasoning' with massive Abox or instance data, with their solution methodology using a logical (i.e., DL) approach for the manipulation of Abox data. The main motivation behind the 'foundational problem' being addressed in OntoMove is to produce a 'new view' of an existing TBOX (and hence the corresponding Abox). MOVE achieves this by proving the semantic correctness and well-formed'ness of the new ontology, which is derived by the application of 'heuristics' or 'optimisation schemes'. The heuristics are extra-logical in nature and are based on a very general characterisation of the notion of an 'Ontology'.

Within the context of direct applications of ontologies in software engineering, there have been some works in using an ontological approach for domain modelling and engineering in software engineering area [5], [4]. The work in [5] proposes a method to restrict a modelling language such as UML to enable certain specification descriptions only based on the domain semantics as mentioned in the ontology. The motivation behind this work is to avoid the inconsistencies that often occur between the analysis (domain model) and the design models. In [4] an approach to build a domain ontology and to map the domain ontology into object-oriented structures is introduced. The argument behind this is the fact that such domain model will not be directly useful to operational reuse unless it is adapted to the common technology within the domain, which is the object-oriented technology within the software engineering domain. Recently, there have also been a number of ontologies introduced in business process model [1], ontologies to model software patterns [26] and the use of semantic web technologies in general toward the development of software maintenance methodologies [10]. All these works highlight the increase of interest in software engineering field to utilise ontology as a mean to standardise processes or tasks. However, despite these recent efforts, there has been no real focus on tailoring the ontologies to meet user specific needs as well as to integrate them with the extraction of ontology-annotated data sets or resources. We believe this focus will play an important role in applying the notion of reuse in ontology engineering.

Finally, it is important to differentiate the system proposed in this paper with the existing tools for ontology editing and alignment such as Protégé [23, 17], or OntoEdit [18]. Whilst the above tools provide efficient techniques to view, visualise, edit and align ontologies, they do not particularly address the issues of (i) user-driven automatic extraction of valid sub-ontologies, and (ii) semantic and structural optimisation of the resulting ontologies. In addition, these tools do not address the application level annotation and linking.

3 OntoMove - Knowledge-Based Profiling and Acquisition

3.1 Overview of the Framework

OntoMove stands for 'Ontologies on the *MOVE*'. The term 'MOVE' itself is an acronym for 'Materialised Ontology View Extraction', which is the title of the research project that is foundational to the work reported herein. A brief overview of the OntoMove framework illustrated in Fig. 1 follows: We use the general term 'component' to refer to an application or user that can supply a set of requirements that is expressed using a pre-defined scheme involving the concepts and relationships from a base ontology. Henceforth, this will be referred to as a 'component requirement specification'. Specifically, this requirements specification is essentially an approximate or incomplete specification of the needs of a requesting *component* and takes the form of a partial labelling of the domain ontology.

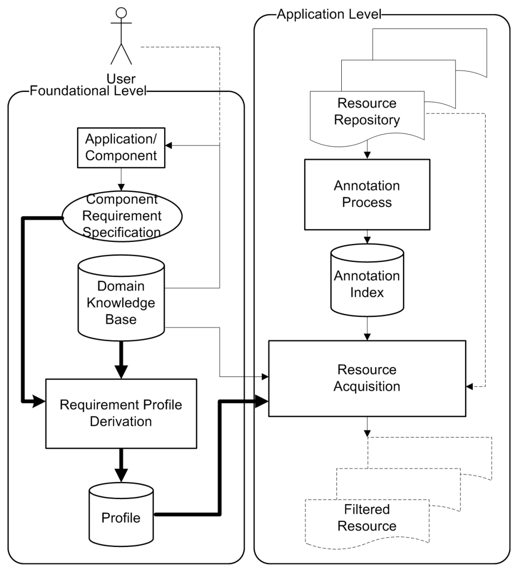


Figure 1. OntoMove Framework

Details notwithstanding, the labelling is the component's (potentially conflicting) expression of interest and consists of semantic information defined in terms of concepts and relationships that constitute the resource domain.

On the basis of the initial requirement specification, a profile of the requesting component is derived (see 'requirement profile derivation' in Fig. 1) based on the concept of a sub-ontology. A component profile is a *complete*[1] requirement specification that is derived using the partial/initial requirement specification that is provided by the component. Between the initial requirements specification for a component and the derivation of its complete requirement profile lies a complex process (as detailed in section 3.2) that is representative of our solution to the foundational-level problem involving the *contraction* of an ontology.

At the application level, we subscribe to a general notion of a resource since the actual type of a resource is irrelevant to our resource acquisition framework. However, in our illustrations of the proposed methodology in section 4, we focus on resources as being functional process templates and unstructured or semi-structured data sets that are of interest in their respective domains being analysed. The assumption that is applicable in this context being that irrespective of the precise type of a resource, the resource under consideration should be (semantically) categorised using a well-defined ontology that is comprehensively representative of the resource domain. Resource acquisition refers to the process of selectively acquiring (i.e., filtering) resources from a resource repository. Also, the concept of 'resource annotation' is a well-defined topic in the community, and its usage in this work consistent with the commonly held interpretation – the categorisation or classification of resources based on a some domain specific criteria such as functionality, its semantic type and similarity to others resources. Although shown as a part of the framework, 'resource annotation' is presently semi-automatically performed using a external tool (see section 3.4). Finally, by a 'component requirement manager', we refer to that part of the framework which facilitates the creation and maintenance of component requirement specifications, associated requirement profiles, resource repositories and the mapping between a requirement profile and the corresponding resource repository.

3.2 Component Requirement Profiling Using Sub-Ontologies

It is uncommon for new applications within an existing domain to use an already available ontology in its entirety. In most cases, depending on the needs, a new application will use refined version(s) of the existing ontology. This concept of utilising sub-ontologies has an interesting application: when only partial views (sub-ontologies) of an existing ontology are used, the semantic range of the sub-ontology in the overall annotated dataset is significantly narrowed; something that leads to the idea of a 'requirement specification' based 'component/application requirement' profile derivation. This problem has stimulated our work in the area of sub-ontology extraction[2]. The extraction process, referred to as Materialised Ontology View Extraction (MOVE) [31, 3], is capable of deriving materialised views, also referred to as sub-ontologies, from a massively sized base ontology. Note that the resulting ontology is semantically complete [2] and a valid ontology independent of the base ontology. This is achieved in MOVE through the enforcement of relevant constraints that take the form of various optimization schemes such as requirements consistency (RCOS), semantic completeness (SCOS), well-formedness (WFOS), and total simplicity (TSOS) that ensure the quality and the optimality of the resulting view. For example, RCOS checks for the consistency of the user specified requirements for the target ontology in the form of the labelling. SCOS considers the completeness of the concepts, i.e. if one concept is defined in terms of an another concept, the latter cannot be omitted from the sub-ontology without loss of semantic meaning of the former concept. Furthermore, it might be possible that the user requirements (labelling) is consistent, but there might be statements that inevitably lead to a solution that is not a valid ontology. WFOS contains the proper rules to prevent this from happening. Finally, applying TSOS to an existing solution (along with its requirements specification) will

[1]The concept of *completeness* is non-trivial and involves qualitative benchmarks along several dimensions. This is elaborated in section 3.2.

[2]Also investigated is its optimization using distributed methods [3]. This aspect is however not relevant in this paper.

result in the smallest possible solution that is still a valid ontology. Fig. 2 shows a schematic of the sequential extraction process that we call *MOVE*. The process begins with the import of the ontology externally represented using an OWL ontology. The actual extraction process/execution of optimization schemes is initiated by way of *requirements specification* by a user or another application and the execution of the other optimization schemes.

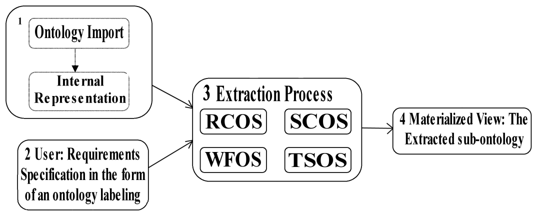

1
Ontology Import
Internal Representation
2 User: Requirements Specification in the form of an ontology labeling
3 Extraction Process
RCOS
SCOS
WFOS
TSOS
4 Materialized View: The Extracted sub-ontology

Figure 2. MOVE: Requirement Profiling Using Sub-Ontologies

Whereas the details of the relevant extraction mechanism being employed are beyond the scope of our current discussion, it is important to note that an externally provided *requirements specification* is used as the basis of the derivation/extraction process. The sub-ontologies derived by MOVE are valid independent ontologies, known as Materialized Ontologies, that are specifically extracted to meet certain component needs. In the extraction process, no new information is introduced (e.g., adding a new concept). However, it is possible that existing semantics are represented in a slightly different manner (i.e., a different *view* is established). Intuitively, the definition states that - starting from a base ontology, elements may be left out and / or combined, as long as the result is a valid ontology, i.e. should be a valid ontology even if the base ontology is taken away. In the process, no new elements should be introduced (unless the new element is a combination of a number of original elements, i.e., the compression of other elements).

3.3 Dynamic Profile Driven Resource Acquisition

We incorporate the idea of a sub-ontology based user profile that contextualises an application's or component's requirements on the basis of the semantic information (i.e., concepts and relationships among them) that is present in the corresponding sub-ontology. The profile is derived by applying the Materialised Ontology View Extraction (*MOVE*) algorithm on a incompletely specified *component requirements specification* that is supplied by the requesting component. The main steps of the overall process are as follows (see section 4 practical illustations):

1. **Component Requirement Specification**: The requesting component selects semantic types and properties from the base ontology (knowledge base) on the basis of the area of its requirements (i.e., interest/specialisation). Note that this specification is rather crude and does not constitute a valid sub-ontology. In fact, the specification itself might be inconsistent in which case a concrete profile may not be derivable. This and other aspects, as previously discussed in section 3.2, are handled by various optimisation scheme present that constitute the sub-ontology derivation algorithm.

2. **Requirement Profile Derivation**: A consistent requirements specification is the input for our sub-ontology extraction (referred to as profile derivation) algorithm, namely *MOVE*. A sub-ontology essentially establishes a context on the basis of the user requirements specification and is indicative of the requesting component's preferences in terms of the semantic types, properties (and their dependencies as derived by *MOVE*) that were either explicitly specified in the requirements specification or were implicitly considered essential for inclusion by *MOVE*. This process (see section 3.2), is more intricate than has been made out to be; however, details not being relevant here, we direct interested reader to [31] or [3] for an in-depth illustration.

3. **Profile Driven Resource Retrieval**: Once a context has been established in the form of a requirement profile, resource retrieval performed within the context of the profile narrows the range of the resources (in the resource repository) by including only those resources that are representative of the semantic types that are present in the context. Note that the retrieval stage is also dependent on the resource annotation index (see section 3.4), which is basically a mapping of the resources within the resource repository to the semantic types within the domain ontology on the basis of their utility, functionality, placement within the ontological hierarchy etc.

The Semantic Scope of a Requirement Profile: For every semantic type within a presently active 'requirement profile' (or multiple profiles), the steps illustrated in Listing 1 are performed in order to derive the semantic scope of the profile within a resource repository. In the following, we briefly explain the algorithm in Listing 1 for establishing the semantic scope of semantic type from the profile within a annotated resource repository: In step 2 of the algorithm, if a primitive concept is selected ', then it is simply added to the established semantic scope denoted by $ResultSet$. In step 3, if a property is selected instead, then find the

Domain ($domainConcept$) and *Range* ($rangeConcept$) of the selected property and perform this algorithm for each of them. Finally, in step 5, the $ResultSet$ that is obtained is essentially a set of concepts from the profile. Complement this set by including all concepts that have been defined to be $equivalent$ (using $OWL : sameAs$ property) to any concept present in the original result set. This step is likely to widen the semantic scope of the results by including the synonym concepts. Once the set of concepts resulting from the application of above mentioned steps is obtained, the *annotation index* is queried to retrieve the list of *annotation objects* that exist for each of the concept included in the result set. Finally, the information in each annotation object is used to generate a (presentable) list of resources that lie within the scope of the requirement profile under consideration.

```
INPUT : A semantic type (concept or property)
        from the 'requirement profile' (R)

OUTPUT: A set of concepts that are either directly or
        indirectly related to a named concept/property
        (ResultSet).
BEGIN
1. ResultSet = NULL;
2. if ( R is Concept )
    {
      if (R is Primitive )
      {
        ResultSet = ResultSet U R;
      }
      else if (R is union or R is intersection )
      {
        Loop through each Concept in R
        {
          C = next(R);
          ResultSet = ResultSet U C;
        }
      }
    }
3.  else if ( R is Property )
    {
      domainConcept =  domain(R);
      do step 2 for domainConcept;

      rangeConcept    =  range(R);
      do step 2 for rangeConcept;
    }
4.  tempSet = NULL;

5.  Loop through each Concept in ResultSet
    {
      C = next(ResultSet);
      tempSet = tempSet U getEquivalentConcepts(C);
    }
6.  ResultSet = tempSet U ResultSet

7.  Return ResultSet;
END
```

Listing 1. Establishing Semantic Scope

3.4 Semantic Resource Categorisation

Semantic resource categorisation by annotation is a well-researched and understood topic; several tools exist for the categorisation of resources based on their semantic content in semi-automate ways [20, 21]. In this work, we are not interested in the categorisation per se. Presently, the resource retrieval module within OntoMove presumes the existence of an annotation index that provides the mapping between resources and ontological descriptions. Toward the application scenario illustrated in section 4, we are using OntoMat [20], which is a publicly available semi-automatic annotation tool.

3.5 Design and Implementation

OntoMove has been designed to work with ontologies represented in the OWL language. This is driven by the fact that OWL is the emerging industry standard and is recommended by the W3C [30] for the representation of ontologies. Furthermore, numerous semantic web tools (e.g., Protégé OWL Plugin [24], OntoMat [20]) supporting OWL have been already developed in the open-source community. In addition, tool builders have developed powerful reasoning systems that support reasoning with ontologies represented in the OWL language (e.g., RACER [25]). This is useful for potential users of OntoMove who want to extend the framework in order to utilise the reasoning capability that is inherent with an OWL based representation scheme. The entire OntoMove framework illustrated in Fig. 1 has been implemented in Java and the JENA ontology API has been utilised to create and manipulate OWL ontology models. As long as domain ontologies are represented in OWL and the resource annotations follow the presently used (OntoMat) annotation vocabulary, the system is usable with any ontology and an arbitrary set of resources.

As mentioned previously, although the annotation phase is shown as a part of the overall OntoMove framework, that functionality is presently being utilised via OntoMat [20], which is a publicly available tool for resource annotation. An important feature of our system is that new annotations may be performed and integrated within the system dynamically; this is absolutely essential since annotation is not a one time job and is best performed incrementally because of the qualitative nature of the task and the fact that resource repositories are frequently updated and/or extended. A built-in 'annotation-indexer' maintains and builds the relationships, namely *index entries*, between the existing semantic types and their instances in the medical TG. Additionally, the annotation-indexer also computes important statistics relevant to the existing annotations, which is useful in determining the quantity and quality of the annotations being performed. These statistics can be dynamically obtained from within the system. For purposes of efficiency, the index generated by the annotation-indexer is serializable-deserializable; as such, generation of the index is a one-time task and user may choose to reload an existing index or re-generate one if there has been some change in the information sources. For maximum flexibility, the annotation-indexer and its associated statistics have been implemented to be usable either with or independent of the main *OntoMove* application.

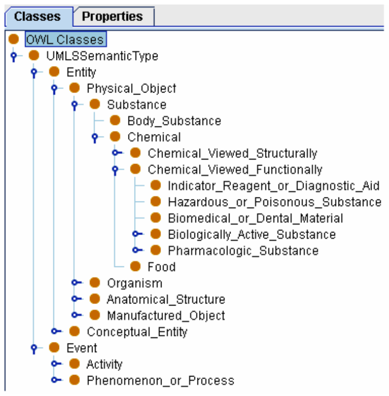
4 Applications

4.1 Medical Resource Retrieval

We demonstrate the application of our OntoMove framework for improving the efficacy of information retrieval in the Medical Information Systems (MIS) domain. Here, the main objective is to produce semantically correct results whilst retrieving information from the vast knowledge sources (i.e., the resource repository) available in the form of the Medical Therapeutic Guidelines (TG) [28]. Moreover, since the domain knowledge captured in the ontology is based on the Unified Medical Language System (UMLS) [29] knowledge sources, namely the UMLS Semantic Network (SN) and UMLS Metathesaurus®, the approach is compatible with controlled vocabularies and classifications used in patient records, administrative health data, bibliographic & full-text databases, and expert systems. The resource retrieval phase is based on the previously (section 3.3) elaborated concept of a 'requirement profile', which is a key aspect within the OntoMove framework. The overall set of semantic types and their instances contained in medical domain vocabularies is massive, for e.g., Gene Ontology [27], GALEN [19] and the UMLS [29] sources used in this work. As such, semantic profiling of a user's field of specialisation or interest is necessary functionality in any medical domain information retrieval system, given the structural and semantic extent of any medical domain information source. Since an individual user is unlikely to be interested in every conceivable category of medical data that is present in the information sources or the vocabularies. It is envisaged that OntoMove's capability to produce semantically correct sub-ontologies based on user preferences will significantly improve the quality of information retrieval in the medical TG domain.

4.1.1 UMLS Knowledge Source Server (UMLSKS)

The *UMLSKS* has been used to gain access to the vast amount of knowledge contained in the UMLS by way of its two main components, viz - the UMLS Metathesaurus® and the UMLS Semantic Network. Fig. 3 is a extremely narrow view of the semantic network and is indicative of the sort of semantic types present in it. The UMLSKS allows the user to (manually or programmatically) request information about particular Metathesaurus concepts, including attributes such as the concept's definition, its semantic types, concepts that are related to it, hierarchical context details etc, all of which can be restricted to source specific details. The UMLSKS also allows the user to request information about the attributes themselves, for example, by querying for all concepts that have been assigned to a particular semantic type, or by querying for all the terms that have a particular lexical tag. The semantic network also contains information about semantic types and their relationships.



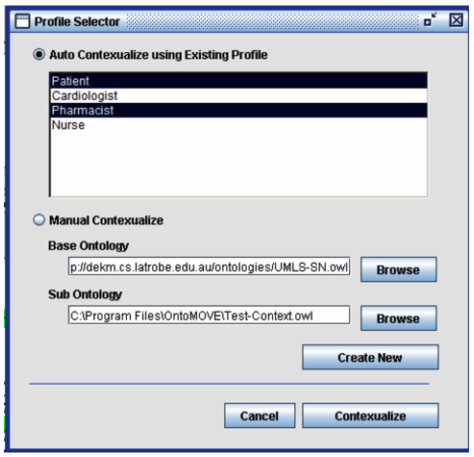
Classes | Properties

- OWL Classes
  - UMLSSemanticType
    - Entity
      - Physical_Object
        - Substance
          - Body_Substance
          - Chemical
            - Chemical_Viewed_Structurally
            - Chemical_Viewed_Functionally
              - Indicator_Reagent_or_Diagnostic_Aid
              - Hazardous_or_Poisonous_Substance
              - Biomedical_or_Dental_Material
              - Biologically_Active_Substance
              - Pharmacologic_Substance
            - Food
        - Organism
        - Anatomical_Structure
        - Manufactured_Object
      - Conceptual_Entity
    - Event
      - Activity
      - Phenomenon_or_Process

Figure 3. OntoMove - UMLS-SN Ontology Class View

4.1.2 Domain Knowledge - UMLS Semantic Network Ontology

The UMLS Semantic Network (UMLS-SN) represents the meta-level of the medical ontology that we develop for purposes of annotation. This is because the UMLS Metathesaurus uses UMLS-SN as its meta level to define medical domain concepts from various medical vocabularies. We represent the entire UMLS-SN taxonomy in the form of a subsumption hierarchy in the OWL language, i.e., as an OWL ontology (referred to as the UMLS-SN ontology henceforth). Although all semantic types have been mapped, we have not mapped every property or relationship that exists between the semantic types; this is because the set of relationships between the semantic types is too massive to be used in its entirety and also because it is not our objective to develop a comprehensive mapping of the entire UMLS-SN in the OWL language (furthermore, the ontological or knowledge engineering perspective to be applied whilst mapping the UMLS semantic network to the OWL language is problematic [11]). Obviously, the ontology that results is basically a very small view of the base UMLSKS – one that concerns our use of UMLSKS. The resulting ontology is imported in *OntoMat* [20] in synchrony with the medical information sources, namely the Medical Therapeutic Guidelines (TG) [28]. A subset of the guidelines are then annotated using the ontology that we imported into OntoMat following which the modified (annotated) re-



[✓ Auto Contexualize using Existing Profile]

Patient
Cardiologist
Pharmacist
Nurse

Manual Contexualize

Base Ontology: p://dekm.cs.latrobe.edu.au/ontologies/UMLS-SN.owl

Sub Ontology: C:\Program Files\OntoMOVE\Test-Context.owl

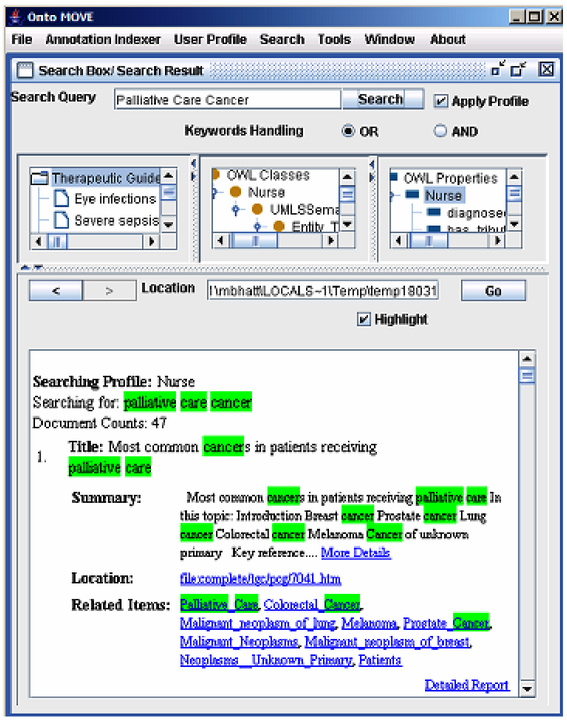
Figure 4. OntoMove - Profile Selector

sources are serialized as a different version.

4.1.3 Medical (TG) Resource Retrieval Interface

As elaborated in section 3.2, we incorporate the idea of a sub-ontology based user profile that contextualises the respective user's area of interest and/or specialisation on the basis of the semantic types (& relationships among them) that are present in the corresponding sub-ontology. The profile itself is derived by applying the profile derivation algorithm (i.e., MOVE) on a incompletely specified 'Requirement Specification' supplied by the end-user.

For the resource retrieval after the derivation of a requirement profile, the user performs a search query (see Fig. 5 consisting of arbitrary keywords (along with the usual means to combine keywords). Prior to the search, the user applies a *profile* (See Fig. 4) in the context of which the search is to be performed. Note that it is also possible to apply more than one profile at the same time, for example, applying a *patient* and *pharmacist* profiles in conjunction. As discussed previously, the effect of applying a profile (or a combination of them) is to establish context based on semantic information about the users area of interest that is contained in the profile. As can be seen in Fig. 5, the retrieved search results are sequentially listed in the lower part of the interface in a sequential manner – the precise order of the results is based on the conceptual similarity/distance between the identifiable semantic type of the search keywords with the annotations that are present in the retrieved documents. In addition to a brief summary of every document that is retrieved, the user also has access to a detailed *Semantic Report* (i.e., information about other Metathesaurus items that are present in the same document and might possibly be of interest to the user) and an utility-oriented *Annotation Viewer* (i.e., explicitly querying the semantic extent



Search Query: Palliative Care Cancer [✓ Apply Profile]

Keywords Handling [✓ OR] AND

Location: I\mbhatt\LOCALS~1\Temp\temp18031

[✓ Highlight]

Searching Profile: Nurse
Searching for: palliative care cancer
Document Counts: 47

1. **Title:** Most common cancers in patients receiving palliative care

Summary: Most common cancers in patients receiving palliative care In this topic: Introduction Breast cancer Prostate cancer Lung cancer Colorectal cancer Melanoma Cancer of unknown primary Key reference.... More Details

Location: file:complete/tgc/pcg/7041.htm

Related Items: Palliative_Care, Colorectal_Cancer, Malignant_neoplasm_of_lung, Melanoma, Prostate_Cancer, Malignant_Neoplasms, Malignant_neoplasm_of_breast, Neoplasms__Unknown_Primary, Patients

Figure 5. OntoMove - TG Resource Retrieval

of a concept/property within the resource repository).

4.2 Business Process Management Systems

The application of our framework in this domain is similar in nature to the one discussed for the medical domain in section 4.1. The methodology being the same, only the domain specific knowledge (i.e., the ontology) and the type of resource varies.

The Knowledge Base For exemplary purposes, we utilise the MIT process handbook, which is a comprehensive framework for organizing large amounts of useful knowledge about business processes [16]. The handbook primarily consists of 3 types of knowledge: (1) Generic models of typical business activities (e.g., buying, making, and selling) that are universal traits of any business activity, (2) Specific case examples of interesting activities that particular businesses have undertaken and (3) Frameworks for classifying all this knowledge [16, chapter 8, pg. 221]. The knowledge contained in the handbook (presently 8000 business processes) has been represented in the OWL language and is available for public use [22]. The hierarchy illustrated in Fig. 6, an extremely small view the overall knowledge base, is illustrative of the types present in the process ontology.

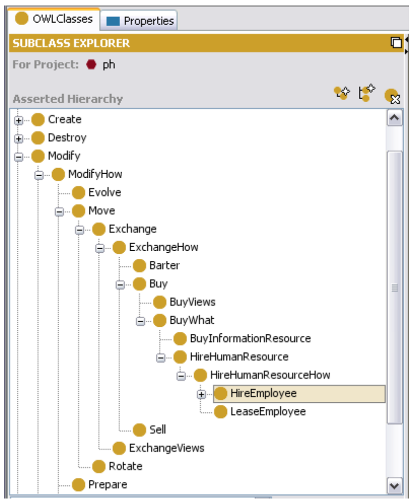


Figure 6. Business Process Ontology

```
<ProcessTemplate>
  <Meta>
    <Office>La Trobe University, Melbourne</Office>
    <ProcessName>Hire Employee</ProcessName>
    <Description>
       Hiring an employee is the usual method of
       filling a long-term need for labour.
    </Description>
  </Meta>
  <ParentProcess>Hire Human Resource</ParentProcess>
  <SubProcess />
  <PrimaryProduct>Employee</PrimaryProduct>
  <Triggers>
    <Trigger>
      Create account for newly appointed
      Employee in Accounts Department.
    </Trigger>
    .
    .
  </Triggs>
  <Tasks>
    <Task>
      <name>Select human resources</name>
      <details filename="SelectHumanResource.xml" />
    </Task>
    .
  </Tasks>
</ProcessTemplate>
```

Listing 2. Process Template

The Resource Process templates are reusable process model structures that can be instantiated and tailored to specific modelling requirements. They serve as knowledge and resources of legacy system for further reuse. In general, a process template (for business, software or other processes) consists of meta-level information including a systematic description of items such as inputs to the process, anticipated outcomes, products involved, actors, triggers, sub-processes and so forth. We adopt a simple XML/RDF based representation scheme for the process template (see Listing 2). The semantic types present in the process handbook ontology can then be used to annotate a set of business process templates (i.e., the resource). Listing 3 is exemplary of an annotation object that utilises the concept 'HireEmployee' from the process handbook ontology in Fig. 6 to annotate the process template in Listing 2. Note that the precise nature of a process template is not relevant in so far as the application of our framework is concerned. In fact, as long as the annotation vocabulary in Listing 3 is being adhered to, the resource repository can exist independent of the framework.

```
<Annotation>
<rdfs:label
  xmlns:rdfs="http://www.w3.org/2001/XMLSchema">
    Hiring Employee Process Template
  </rdfs:label>

  <rdfs:Resource
  xmlns:rdfs="http://www.w3.org/2001/XMLSchema">
    <processHandbook>HireEmployee</processHandbook>
  </rdfs:Resource>
</Annotation>
```

Listing 3. Annotation Object

The Use Case - Semantic Retrieval of Process Templates Within a BPMS, it is common to have a process template retrieval component that is instantiated several times whenever new projects (or business processes) are initiated. The main objective of this component is to facilitate the reusability of existing process templates contained within (and continuously to) a repository. In fact, such a component (and use-case) is typical of other process-modelling systems too, including those modelling software-development processes[3]. Restricting ourselves to a BPMS, our framework can be directly applied for the selective filtering of business process templates (during the retrieval phase) on the basis of user specified requirements. Assuming that the process templates (Listing 2) have been annotated (in the vocabulary in Listing 3) using the knowledge contained in a process ontology (see Fig. 6), the framework can be directly applied to derive a requirement profile that is representative of the requirement specifications of a new project (see 3.2). Once this is achieved, the process templates that lie within the semantic scope of the requirement profile are easily derivable in the manner illustrated in section 3.3.

5 Conclusion and Future Work

We proposed and implemented a framework for the refinement of ontologies for purposes of their reuse. The mechanism to reuse ontologies, coupled with the profiling of an application's interest in a subset of the ontology is used to implement the idea of semantic profiling and acquisition a component's resource requirements. The following extensions to the existing framework are in progress: Requirement profile derivation in the present system is performed in a built-in sequential manner. We are working

[3]For instance, the Microsoft® Visual Studio Team System has a process template manager module that simply present the user with the list of all process templates from the Team Foundation server.

toward utilizing the distributed architecture for the profile derivation algorithm that has been designed for use in a distributed cluster environment by porting an existing platform-dependent version (see [3]) to the present Java-based framework. The distributed architecture is essential in order to make the profile derivation process computationally optimal in a WWW environment by leveraging a cluster setup that is not uncommon for a business organisation. We are achieving this goal via the medium of web-services, i.e., a web-service performing as a mediator between the distributed profile derivation framework operating in a Linux cluster (supplied VPAC, Australia) and the Java-based *OntoMove* application. For consistency of representation across the system, all transfer of information is being designed to be entirely based on the OWL serialisation syntax (i.e., XML/RDF). Furthermore, work is in progress to develop built-in functionality, similar to that utilized via OntoMat, to create manual annotations using our custom annotation vocabulary. Although essentially similar to OntoMat in terms of the overall results obtained, this built-in functionality will differ in two regards: (a) Instead of an RDF-based annotation schema, we develop a annotation ontology in the OWL language, (b) The annotation schema itself will not be static as is the case with OntoMat thereby allowing users to specify their own annotation types. Most importantly, these extensions also facilitate the provision of the entire requirement driven resource retrieval methodology within one framework or application.

References

[1] A. Bernstein. Process recombination: An ontology based approach for business process re-design. *SAP Design Guild*, 7, October 2003.

[2] M. Bhatt, A. Flahive, C. Wouters, W. Rahayu, and D. Taniar. Semantic completeness in sub-ontology extraction using distributed methods. In *Lecture Notes in Computer Science*, volume 3045, pages 508–517. Springer Verlag, 2004.

[3] M. Bhatt, A. Flahive, C. Wouters, W. Rahayu, and D. Taniar. Move: A distributed framework for materialized ontology view extraction. *Algorithmica*, 45(3):457–481, 2006.

[4] R. de Almeida Falbo, G. Guizzardi, and K. C. Duarte. An ontological approach to domain engineering. In *The international Conference on Software Engineering and Knowledge Engineering (SEKE'02), Italy*, pages 351–358, 2002.

[5] J. Evermann and Y. Wand. Toward formalizing domain modeling semantics in language syntax. *IEEE Transactions on Software Engineering*, 31(1):21–37, January 2005.

[6] A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas. The summary abox: Cutting ontologies down to size. In *The Semantic Web - ISWC 2006*, volume 4273 of *LNCS*, pages 343–356. Springer, 2006.

[7] N. Foo. Ontology revision. *Lecture Notes in Computer Science*, 954:16–31, 1995.

[8] I. Horrocks. Daml+Oil: A reasonable ontology language. In *Proceedings of EDBT-02*, pages 2–13. Volume 2287, LNCS, Springer Verlag, 2002.

[9] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[10] D. Hyland-Wood, D. Carrington, and S. Kaplan. Toward a software maintenance methodology using semantic web techniques. In *Proceedings of the Second International IEEE Workshop on Software Evolvability at ICSM-06*, 2006.

[11] V. Kashyap and A. Borgida. Representing the umls semantic network using owl: (or "what's in a semantic web link?"). In *International Semantic Web Conference*, pages 1–16, 2003.

[12] Y. Lin and H. Ding. Ontology-based semantic annotation for semantic interoperability of process models. In *CIMCA '05 and IAWTIC, Vol 1*, pages 162–167, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Y. Lin and D. Straunskas. Ontology-based semantic annotation of process templates for reuse. In *Proc. of 10th CAiSE/IFIP8.1/EUNO, International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD05)*, pages 31–37, Porta, Portugal, 2005.

[14] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12:286–302, 2003.

[15] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. An infrastructure for searching, reusing, and evolving distributed ontologies. In *Proceedings of the twelfth international conference on World Wide Web 2003*, pages 439–448, 2003.

[16] T. W. Malone, K. Crowston, and G. A. Herman, editors. The MIT Press, 1st edition, September.

[17] N. F. Noy, M. Sintex, S. Decker, M. Crubzy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems 16*, pages 60–71, 2001.

[18] OntoEdit. OntoEdit - The OTK Tool Repository. Karlsruhe University, Germany.

[19] Ontology GALEN, 2001. http://www.opengalen.org/.

[20] OntoMat. OntoMat - An Interactive Annotation Tool.

[21] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. Kim - semantic annotation platform. In *International Semantic Web Conference*, pages 834–849, 2003.

[22] Process Handbook OWL-Version. The mit process handbook in owl. University of Zurich.

[23] Protege. An Ontology and Knowledge Base Editor. http://protege.stanford.edu.

[24] Protege OWL Plugin. Ontology editor for the semantic web. http://protege.stanford.edu/plugins/owl/index.html.

[25] RACER. Renamed ABox and Concept Expression Reasoner. http://www.sts.tu-harburg.de/ r.f.moeller/racer/.

[26] J.-M. Rosengard and M. F. Ursu. Ontological representations of software patterns. In *KES*, pages 31–37, 2004.

[27] The Gene Ontology Consortium. Gene Ontology: A Tool for the Unification of Biology. Natural Genetics, 25:2530, 2000.

[28] Therapeutic Guidelines Limited. Medical Therapeutic Guidelines. http://www.tg.com.au/.

[29] U.S. National Library of Medicine. Unified Medical Language System (UMLS). http://www.nlm.nih.gov/research/umls/.

[30] W3C. OWL Web Ontology Language: A W3C Recommendation, February, 2004. http://www.w3.org/News/2004.

[31] C. Wouters, T. S. Dillon, J. W. Rahayu, and E. Chang. A practical walkthrough of the ontology derivation rules. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 259–268, London, UK, 2002. Springer-Verlag.