An Ontology-Based Approach to Decision Support for Healthcare Workflows

Cristian Cocos*,a, Wendy MacCaula, Bryan Kramerb and Markus Latzelb

*a Centre for Logic and Information, St. Francis Xavier (StFX) University, 54B St. Mary’s Street, Antigonish NS, Canada B2G 2W5, tel.: +1 (902) 867-4931, fax: +1 (902) 867-1397
b Palomino System Innovations Inc., 533 College Street, Suite 404, Toronto ON, Canada M6G 1A8, tel.: +1 (416) 964-7333, fax: +1 (866) 256-7866

Abstract. We provide the description of a healthcare workflow management system, with emphasis on the associated knowledge bases, and their interface with the system. The two major knowledge bases are: (a) an access control ontology, which contains the access control policy, and (b) a palliative and seniors’ care ontology, which stores palliative and seniors’ care knowledge, and offers decision support for the workflow. Both these ontologies take advantage of healthcare knowledge contained in several established healthcare terminology standards (such as SNOMED-CT and ICNP), as well as information gathered from focus groups and interviews with clinicians and healthcare managers. Their development principles are expounded, their structure analyzed, and their interaction with the system elaborated by means of querying examples and interaction scenarios.

Keywords: ontology, workflow, patient-centered care, access control, SNOMED-CT

1. Introduction

Researchers at the StFX Centre for Logic and Information (CLI) are currently developing a careflow management system for palliative and seniors’ care, which is scheduled to be deployed at several hospitals in rural Nova Scotia in the near future ([18]). This work is part of a collaborative six-year R&D project involving clinicians and managers of the local health authority (Guysborough Antigonish Strait Health Authority (GASHA)) and an industry partner (Toronto-based Palomino System Innovations Inc.). This system aims to streamline the workflow by improving documentation and communication, and to promote compliance with national guidelines. It will offer several high-end features such as adaptability, flexibility, semantic web integration (in view of projected interoperability with globally-accepted biomedical controlled vocabularies), etc. In order to comply with the desideratum of patient-centered health care delivery (whereby health care data is owned by the patient), this endeavor requires a major commitment to data privacy and confidentiality. In an era marked by the advent of electronic health records (EHR), controlling and restricting access to healthcare data is of key importance. It is our belief that an ontology-structured knowledge base constitutes an ideal mechanism for the implementation/representation of a role-based access control (RBAC) policy. Here we present the design and employment of such an RBAC ontology (the Access Control Ontology (ACO)), which is intended to fulfill the commitment to privacy and confidentiality. We also introduce an ontology-based mechanism, used to offer decision support for the palliative and seniors’ care workflow. While this article may be viewed as having predominantly a practical value, several issues are of certain theoretical interest, most of which deal with our treatment of the OWL version of SNOMED-CT.

The paper is structured as follows: we first cover the background of the careflow management system, and then we present the access control side under two
implementational styles. Next we give an overview of the palliative and seniors’ care ontology (PCSO), including a brief listing of its sources and development principles, proceeded by a presentation of the workflow-ontology interaction scenario via examples. We conclude by outlining some directions for future work. Some of the material on the ACO covered in section 3 is contained in [2].

2. Background

The high-level conceptual blueprint of the careflow management system (CfMS) under development is shown in figure 1 ([18]):

![Fig. 1. A Careflow Management System ([18]).](image)

The two major components relevant from our point of view are:

- The Workflow Management System (WfMS). A WfMS [27] is a system that manages the execution of workflows through the use of software, running on one or more workflow engines. The WfMS is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of information technology tools and applications. A process definition is the representation of a process in a form which supports automated manipulation, such as modeling, or enactment by a WfMS.

- The Data Management System (DMS). A DMS manages data and knowledge during the execution of healthcare workflows. It includes the following sub-components:
  - the knowledge base—in our case, the ontologies. These provide an efficient way of managing medical, health and organizational knowledge. The CfMS allows the ontologies to guide and inform the workflow process and offer decision support using real-time access to knowledge.
  - the patient database—a.k.a. the Electronic Health Records (EHR). An EHR is an evolving concept defined as a systematic collection of electronic health information about individual patients or populations [28]. It is a record in digital format designed to be shared across different healthcare settings. Such records may include a whole range of data in comprehensive or summary form, including demographics, health history, medication and allergies, laboratory test results, radiology images, and billing information.

Some prominent terminological resources that occupy a central role in the development of the ontological knowledge bases are SNOMED-CT ([13]), and the International Classification for Nursing Practice (ICNP) ([11]). Both standards are currently used extensively with the aim of building a palliative and seniors’ care ontology (PSCO) that will guide and offer decision support for the palliative and senior’s care workflow; they also play a (slightly less emphatic) role in the development of the ACO.

The workflow management system thus interacts with (among others) both the ACO and the PSCO in order to obtain access control clearance for the system user and, respectively, to guide the workflow in deciding between alternative trajectories on the basis of palliative and seniors’ care knowledge. While both of these scenarios will be elaborated in the following, a quick illustration of the latter might be able to help ground and recall the concepts and techniques introduced below: Cancer Care Nova Scotia [38] has issued Best Practice Guidelines for the management of cancer-related pain in adults [39]. One such guideline specifies the procedures to follow as a result of a pain assessment: if the patient is not currently on a regular opioid regimen and if his/her pain intensity assessment reveals background discomfort or mild pain, then the patient is being maintained on a non-opioid regimen unless the mild pain is not stable, case in which the patient is being switched to a weak opioid regimen. This guideline is to be captured in PCSO as a class restriction/axiom, and once a patient-specific workflow reaches the point where a pain assessment is undertaken, the WfMS queries the ontology with
the patient pain intensity data, and returns the corresponding conduct to be followed by the workflow (namely ‘WeakOpioidRegimen’ or ‘NonOpioidRegimen’). More about this in section 4.

3. Access Control (ACO)

3.1. Initial Considerations

Representing the roles of a RBAC policy as classes of an access control ontology emerges as a natural choice [32], especially when the ontology in question accommodates roles natively. We have thus decided to adopt an upper-level ontology where roles figure prominently in the asserted hierarchy. Basic Formal Ontology (BFO) has enjoyed a certain success in semantic web and ontology circles, and lends itself quite naturally to the type of enterprise in which we have engaged. From our perspective, four BFO classes are of immediate interest to ACO: BFO:role, BFO:object, BFO:process, and BFO:generically_dependent_continuant. For a presentation of BFO the reader is invited to consult [25] and [26].

[22] emphasizes two ontology-based access control techniques: (a) the “Roles as Classes” technique, and the (b) “Roles as Values” technique ([22], pp. 76-77). The former represents RBAC roles as OWL classes; role hierarchies are consequently rendered as OWL class hierarchies, and role dominance is represented as the inverse of the class inheritance relation: “[A] role represented by a subclass dominates a role represented by its superclass. This corresponds to the intuition that in role hierarchies, members get more privileges as one moves up the hierarchy, while in class hierarchies, classes get more attributes as you move down” (Ibid., p. 76).

The “Roles as Values” technique models roles as “instances of the generic Role class using two properties role and active-role to link a subject to her possible and active roles, respectively” (Ibid., p. 77). While this strategy appears to be “simpler and more concise” on the surface, adding “the capability to define role hierarchies is more difficult in this representation and requires adding rules to the ontology, either in SWRL [[30]] or N3 [[31]], depending on the kind of reasoner used” (Ibid.). Moreover, this approach “can not [sic] exploit a DL reasoner’s ability to determine the subsumption relationships between a query and all of the classes in [the access control] policy” (Ibid., p. 78) hence, from this point of view, there is a definite “advantage of defining roles as classes,” in that “queries about a particular access request (Can John use printer p43?) and queries about a general class of access requests (Can every student use lab printers?) can be answered efficiently using a standard DL reasoner through subsumption reasoning” (Ibid.). Table 1 provides an overview of the differences between the two strategies.

We have, thus far, opted for an implementation along the lines of the “Roles as Classes” technique, most of all due to our familiarity with standard DL reasoners. Perhaps the most significant difference between our implementation and the one expounded in [22] comes from our reliance on the implicit BFO ontological commitment to the so-called ‘abstract particulars.’ In [22], instances of role (sub)classes such as ‘Citizen’ or ‘Resident’ are human individuals: “Suppose we want to model the US Persons hierarchy; we will have three base classes, Citizen, Resident, and Visitor, which are defined as subclasses

<table>
<thead>
<tr>
<th>Property</th>
<th>Roles as Classes</th>
<th>Roles as Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining Roles</td>
<td>&lt;RoleName&gt; rdfs:subClassOf <a href="">Role:Role</a></td>
<td>&lt;RoleName&gt; a <a href="">Role:Role</a></td>
</tr>
<tr>
<td></td>
<td>&lt;ActiveRoleName&gt; rdfs:subClassOf <a href="">OfficialRole:ActiveRole</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;ActiveRoleName&gt; rdfs:subClassOf &lt;RoleName&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;ActiveRoleName&gt; bfo:activeRole</td>
<td></td>
</tr>
<tr>
<td>Role Hierarchy</td>
<td>&lt;RoleName&gt; rdfs:subClassOf &lt;SuperRoleName&gt;</td>
<td>&lt;RoleName&gt; rdfs:subClassOf &lt;SuperRoleName&gt;</td>
</tr>
<tr>
<td>Permission Association</td>
<td>OWL class expression</td>
<td>OWL class expression</td>
</tr>
<tr>
<td>Static Separation of</td>
<td>&lt;Role1&gt; owl:disjointFrom &lt;Role2&gt;</td>
<td>&lt;Role1&gt; rdfs:subClassOf &lt;Role2&gt;</td>
</tr>
<tr>
<td>Duty Constraint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic Separation of</td>
<td>&lt;ActiveRole1&gt; owl:disjointFrom &lt;ActiveRole2&gt;</td>
<td>&lt;ActiveRole1&gt; rdfs:subClassOf &lt;ActiveRole2&gt;</td>
</tr>
<tr>
<td>Duty Constraint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queries</td>
<td>role activation permitted, separation of duty, access monitoring</td>
<td>role activation permitted, separation of duty, access monitoring</td>
</tr>
<tr>
<td>Enforcing RBAC</td>
<td>Mostly using DL reasoning</td>
<td>Mostly using roles</td>
</tr>
</tbody>
</table>
of a Role class. The other classes in the domain are defined as subclasses of one of these classes. [...] [For example,] Alice is in the Citizen role [...] and Bob is in Visitor and TemporaryResident role” (Ibid., p. 76). In contrast, as it is the case in BFO-based ontologies, in ACO instances of the BFO:role class and its children are abstract individual roles (role ‘tropes’) that are being had by human individuals via some property or other (in our case PCO:hasRole and its inverse PCO:roleOf); these human individuals are not to be confused/conflated with their roles, and hence are members of a class disjoint from BFO:role.

Two major types of entities (‘resources’) are subject to access control under the ACO scenario: informational items and actions. Database fields are paradigm examples of the former category (e.g., patient ID, patient name, primary diagnosis etc.), though reports also constitute informational content that carries access restrictions (e.g., incident reports). The class of actions has been assembled from actions implemented by the workflow management system, and represents actions that the system user can (or cannot) invoke, such as form/report printing, faxing, phoning, appointment scheduling, etc. Corresponding to these two types of resources, two mechanisms have been employed (‘clearance levels’ and, respectively, ‘permission levels’), so that users’ access control credentials will be checked at login time, and consequently some of these information fields and/or actions will be cleared for access.

We will describe in the following two ways of implementing access control policies in the ACO (two ‘reasoning approaches’). Each of them has advantages and disadvantages, which will be discussed during the course of expounding the second approach, as this has been developed in response to perceived shortages of the first.

3.2. Approach 1: Hierarchical Roles

Here are the defining features of the access control policy implemented using the ‘Approach 1’ strategy:

1. Roles are organized hierarchically, hence permissions granted at higher levels of the hierarchy are inherited by roles lower in the hierarchy (“hierarchical RBAC” in [22]);
2. Resources are organized hierarchically—e.g., allowing access to a whole form means allowing access to all of its fields;
3. Access control constraints can be provided for each form field and action individually;
4. Unlike a purely linear hierarchy, we allow for disjoint roles, hence the permissions range of roles may not always intersect;
5. Database fields can be accessed as both read only and write;
6. System users may have multiple roles with regard to the same patient; in such circumstances, the clearance level assigned is a union of permissions for all roles involved.

The BFO:role branch contains the main mechanism of our RBAC implementation, whereby roles relevant from the point of view of access control in a palliative/seniors care setting have been grouped on categories representing clearance and permission levels for informational items and actions respectively. As mentioned above, the ‘role’ branch is intended to accommodate roles such as ‘Patient,’ ‘Caregiver,’ ‘Medical Doctor’ etc. Most of the classes that populate this branch have been imported straight from SNOMED-CT: we thus saw fit to import SNOMED’s ‘Occupation (occupation)’ (SCTID_14679004) branch in its entirety, as it contains a significant portion of relevant roles. Interestingly enough, SNOMED does not comprise a properly titled ‘role’ class, nor does it contain any class to this effect, hence most of the classes that we have assembled under the ‘role’ branch have been culled from various (and quite heterogeneous) places in SNOMED—mostly ‘occupation’ and ‘person.’ As such, our endeavor can also be viewed as a SNOMED-CT restructuring enterprise by bringing it in line with BFO—an undertaking that, as of this writing, is being contemplated under the auspices of the IHTSDO (the SNOMED-CT custodian). HealthCareRole (see figure 2) is another ACO-specific class that groups several SNOMED-CT classes such as ‘Caregiver (person),’ ‘Clinical trial participant (person)’ etc. The SNOMED ‘Relative (person)’ class (SCTID_125677006) comprises the usual relative roles, as well as some more or less exotic degrees of human inter-relation, such as ‘Great-great grandparent (person),’ ‘Fraternal twin brother (person),’ etc.

Fig. 2: Roles in ACO.
The core ACO mechanisms reside under the Clearance-Level0Role and PermissionLevel0Role branches. These defined classes contain respectively five and two defined (sub)classes labeled ClearanceLevelsRole, ClearanceLevelsRole and, respectively, PermissionLevel0Role (figure 3), where \( x \) takes values from 1 to 4, and \( y \) and \( w \), 1 and 2. (The number of clearance and permission levels is subject to future amendments.) Due to space limitations we will refrain from elaborating the labeling scheme, though the labeling principles should emerge clearly from figure 3. We will detail the content of the clearance level hierarchy, and occasionally point out the differences between it and the permission level hierarchy.

Fig. 3: ACO clearance and permission levels

Each of the ClearanceLevel\( x \)Role classes is defined as a union of roles that have a certain security clearance level, which is to be dictated by real-life security and privacy considerations. ClearanceLevel2Role, for example, is a union of the following SNOMED CT classes: ‘Physiotherapist/occupational therapist (occupation),’ ‘Social worker (occupation),’ ‘Community nurse (occupation),’ ‘Pain management specialist (occupation)’ and ‘Pharmacist (occupation)’ (see the Manchester syntax rendering in figure 4). Similarly, PermissionLevel\( y \)Role classes are unions of roles with the same permission level; e.g., PermissionLevel2Role is a union of ‘Worker in religion (occupation),’ ‘Interpreter (occupation),’ ‘Pharmacist (occupation)” and ‘Translator (occupation).’ Each of these two mechanisms governs access to each of the two types of resources that are subject to access control: clearance level classes govern access to informational items, while permission level classes to actions (instances of SystemProcedure (fig. 6)).

Clearance level classes have been construed in a slightly non-orthodox manner as classes of non-access (see below figures 7 and 8). In view of this, a corollary of the cascading structure of clearance levels is that all ClearanceLevel\( y \)Role roles also have clearance level \( x \) for \( x \geq y \), hence all information that is allowed to, say, a community nurse (a ClearanceLevel2Role), is also allowed to a clinical oncologist (a ClearanceLevel0Role role), but not vice versa. In general, a parent inherits the clearance level of its children. This behavior might look striking at first glance, since it might be seen as contrary to the spirit of the subsumption hierarchy; we will discuss additional aspects of this arrangement at the beginning of the next (sub)section (3.3), where the motivation for such a procedure will become apparent.

Figure 3 shows that the ClearanceLevel2Role class has two children: ClearanceLevel32Role and ClearanceLevel31Role, which have been stipulated as disjoint, so that the two actually constitute a partition of ClearanceLevel2Role. A similar account holds for permission levels (see also figure 3), and while we have not designed any disjoint permission levels, these can be captured in a manner completely analogous to clearance levels.

We also added classes that represent actual information and actions that are subject to access control, and devised a method to tie clearance and permission level roles with that information. We have thus opted to add six relations (‘object properties’) together with reciprocal (‘inverse’) relations:

Fig. 4: ClearanceLevel2Role definition.
1. `hasWriteAccessTo` and `hasReadAccessTo` are relations from ‘role’ to ‘DatabaseField’ (see below), and, respectively, to ‘Report (record artifact)’ or DatabaseField.’ `hasWriteAccessTo` is a child of `hasReadAccessTo`, hence whoever has write access to some resource obviously has read access to it. Their reciprocals are `writeAccessibleTo` and `readAccessibleTo`, with the former being a child of the latter;

2. `invokableBy` is a relation from ACO:SystemProcedure to BFO:role, and plays the same role as either of `read/writeAccessibleTo` vis-à-vis action permissions;

3. `hasRole` (adapted from ICNP) is a relation whose range is BFO:role, and is intended to tie roles with their bearers; its reciprocal is `roleOf`;

4. `hasClearanceLevel` ties the SNOMED-CT class of ‘Homo sapiens (organism)’—a sub-class of BFO:object—with a clearance level. Its reciprocal relation is `clearanceLevelOf`;

5. similarly, `permissionLevelOf` connects permission levels with elements of ‘Homo sapiens (organism).’

The classes that represent controlled information have been grouped under the BFO:generically_dependent_continuant branch, which is the BFO entity designed to account for informational and other abstract entities. ACO imports the Informational Artifact Ontology (IAO) ([7]) under the BFO:generically_dependent_continuant class, and entities of the IAO are used to define ACO’s metadata fields. The root of the IAO is ‘information content entity’ (IAO_0000030). We have added the following chain of ACO-specific subclasses as a child of IAO_0000030: ‘Proposition,’ ‘Declarative-Proposition,’ and ‘DatabaseField’ (see figure 5). While the latter inclusion might, at first glance, strike some as strange, we are confident that this choice will emerge as ontologically compelling once one contemplates the concrete case of a numerical value (126, say), housed in the ‘Blood Pressure’ field of a medical form—hence the ‘BloodPressureField’ contains a proposition about (the value of) blood pressure, namely ‘This patient’s blood pressure has value 126 mmHg.’

The type of information that makes the subject of access control policies has been, at this point, limited to community palliative and seniors’ care records. Our focus has been to extract information fields from GASHA’s palliative and seniors’ care programs. The GASHA workflow collects patient information using numerous forms. Form fields have been represented as children of form classes.

Fig. 5: Database fields.

Similarly, classes that represent controlled actions comprise the ACO:SystemProcedure class, which is a child of BFO:process (figure 6).

Fig. 6: System procedures.

Finally, all the classes that make the subject of access control policies have been endowed with restrictions that outline their corresponding clearance/permission level. Here are a few examples:

The HospitalizationHistoryField (ACO_0000101), a child of the F1.8Field class (i.e., a field on GASHA form F1.8) has been designed so as to not be accessible to a clearance level 2 role—which would obviously make it inaccessible to any role in a higher clearance level 3 role.

2 At present there is no standard nomenclature for GASHA forms and fields. Some labels for both forms and fields have thus been conceived by ontology developers on a more-or-less whimsical basis, therefore we caution readers to not be tempted to read too much in the labeling of some of the forms. The workflow system, however, will use internally only ontology URIs to refer to ACO entities.
clearance level category (i.e., 31, 32 and 4) as well. Figure 7 shows this:

![Figure 7: Hospitalization History field restrictions.](image)

F1.10 form will not be accessible to a clearance level 3 role, which means that all fields on this form will bear the same restriction. This is a case where the restriction has been implemented at the parent class level (F1.10Field (ACO_0000129)), instead of adding it to all component fields individually (see figure 8).

![Figure 8: Form F1.10 restrictions.](image)

PhoningJob will only be invokable by a PermissionLevel2Role:

![Figure 9: PhoningJob restrictions.](image)

The last of the relevant BFO classes is BFO:object, whose only direct child is the ‘Organism (organism)’ SNOMED class (SCTID_410607006). This contains ‘Homo sapiens (organism)’ as an only subclass (SCTID_337915000), and represents the main ACO role-bearer.

ACO currently contains approximately 10,000 classes, and its level of DL expressivity is SROLF, which is N2ExpTime-hard ([24]). Reasoning and querying the ACO using widely available free or open source OWL-DL reasoners (Pellet [37], FaCT++ [36], etc.) currently poses no tractability problems, nor do we envisage any in the future.

According to the basic interaction scenario, access control clearance is checked at login time, by querying ACO upon user login. ACO provides in return a list of GASHA form fields and reports, and a list of system actions forbidden to the user. The workflow system—that, by design, is in possession of the list of all actions and informational items—acts accordingly, by blocking access to the requisite actions and information entities.

An example of a DL query is the following: say Individual, is an instance of the ‘Homo sapiens (organism)’ class that has been defined as a psychotherapist—i.e., is an instance of ‘hasRole some ‘Psychotherapist (occupation)’.’ A query such as ‘not (accessibleTo some (roleOf value Individual,))’ will reveal all the form fields that are not accessible to Individual,.. Another way of defining an individual is to assign it a clearance level outright, without going through the detour of specifying its role, thus sparing the reasoner the effort of figuring out the individual’s clearance level. If Individual2 is defined as ‘hasClearanceLevel some ClearanceLevel0Role,’ the query ‘not (accessibleTo some (roleOf value Individual,))’ will return all the form fields not accessible to Individual,.

3.3. Approach 2: Property Chains

The main reason that led to the search for an alternative way of encoding the access control policy in the ontology is the slight ontological awkwardness of the first approach. Note that the tree of clearance levels outlined in the previous subsection postulates some ClearanceLevelxRole (resp. PermissionLevelxRole) classes as subclasses of some other such ClearanceLevelyRole (resp. PermissionLeveluRole) class; all of them are, in the ultimate analysis, subclasses of ClearanceLevel0Role (resp. PermissionLevel0Role) (see figure 3). Given the peculiar manner in which BFO models role classes as classes of individual abstract roles (as explained above (section 3.1)), the asserted hierarchy of clearance/permission levels has the problematic outcome that a role class such as ‘Translator (occupation)’—which is a component/subclass of ClearanceLevel4Role)—simply is a ClearanceLevel0Role (i.e. a ‘Medical doctor (occupation)’ or ‘Clinical oncologist’ (occupation)); this obviously says (much) more than that whatever is prohibited to a Medical doctor is prohibited to a Translator. Nevertheless, in the strict context of the ACO and of what ACO is supposed to achieve, this ontological quirk is of little consequence, as querying and reasoning certainly give the desired results; for all intents and purposes, then, this reasoning style remains a viable solution. Problems may arise if ACO is integrated in a larger ontology, or reused in a
different scenario, where this particular ontological compromise may become unaffordable.

The second aim targeted by the revamping of the ACO reasoning mechanism is compliance with one of the core tenets of access control theory, according to which everything that is not explicitly permitted should be forbidden. Approach 1 did precisely the opposite: by default, the workflow system had been instructed to allow access to an informational entity (form field or report), unless explicitly forbidden by the ontology. This conduct was adopted at the time approach 1 was designed due to incomplete information about the GASHA form fields: as the activity of requirements gathering unfolded, the list of all form fields subject to access control became available, hence making possible an approach where, upon querying, the ontology furnishes the list of accessible fields. Queries can now be formulated in straightforward positive fashion, unlike in the previous case where one was supposed to ask, in effect, what form fields are forbidden to a role and/or clearance level.

With the exception of the first and fourth features, the defining features of the access control scenario under this implementation remain the same as in the case of approach 1 (see feature list in the first paragraph of section 3.2). Relevant roles are still grouped in clearance levels via ClearanceLevelRole, resp. PermissionLevelRole classes, though the group of such classes exhibits no tree-like hierarchy: no clearance/permission level class is a subclass of another such class, thus upholding ontological accuracy (figure 10). Approach 1 object properties (hasReadAccessTo, hasRole, invookableBy, etc.—see the list in the previous subsection) have also been maintained, though, as we will see, the main reasoning mechanism has now been relegated to restrictions imposed on some of them.

The relevant modifications of the ACO taxonomy under approach 2 are:

1. Given that we do not envisage dealing with subclasses of individual form fields, they have now been modeled as individuals; e.g., where MedicationNameField was a class in approach 1, in approach 2 it has been modeled as an individual;
2. The same goes for the SystemProcedure class (see figure 6): where, e.g., DocumentSigningJob was a class in approach 1, it now is an individual (figure 11);
3. In addition to grouping form fields in forms (see figure 5), they have now been grouped on seven categories of uniform access as in figure 12; all form fields in each category thus have the same access control policy attached. Furthermore, these seven categories have been clustered into (a) one class that contains them all (AllGASHAForms&Fields), and (b) classes that correspond to each clearance level (classes with the FormCL prefix, figure 13).

---

**Fig. 10: Clearance and permission level roles.**

---

**Fig. 11: System procedures.**

---

**Fig. 12: New field categories: PatientMedication.**

---

3 ACO under approach 2 reflects some differences from the access control policy at the time approach 1 was implemented (essentially fewer clearance and permission levels); these differences can be seen by comparing figures 3 and 10, though a complete listing is not essential for the understanding of the approach 2 reasoning mechanism.
Given the taxonomy depicted above, capturing the access control policy comes down to writing a series of axioms of the form ‘All humans love all cats.’ It turns out that expressing such a mundane first order logic (FOL) sentence in OWL 2 DL (\(\mathbb{SROIQ}\)) is far from straightforward, and, in fact, requires excruciatingly complicated workarounds stemming, essentially, from the lack of means to express object property unions (‘role unions’ in DL lingo). In designing the requisite access control policy restrictions, we have taken inspiration from [33]. We will illustrate this design in the following by means of an example involving concrete ACO data. Say we need to capture in ACO the following sentence (henceforth, the ‘target axiom’): ‘All instances of ClearanceLevel1Role have read access to all form fields in FormsCL1,’ where FormsCL1 is the union of PatientDemographicData, PatientStatusIndicators and CareTeamComposition, and ClearanceLevel1Role is ‘Worker in religion (occupation)’ (see figure 10). The first step is to represent all classes involved as object properties in the following manner:

i. ClearanceLevel1Role EquivalentTo pClearanceLevel1Role value AnIndividual1,

ii. FormsCL1 EquivalentTo pFormsCL1 value AnIndividual1,

where AnIndividual1 is a dummy individual that will only be used for this purpose. With this, the target axiom can be written as a property chain restriction to the hasReadAccessTo object property: pClearanceLevel1Role \(\circ\) inverse(pFormsCL1) SubPropertyOf hasReadAccessTo. The final move is to make sure that the ClearanceLevel1Role class is not empty, otherwise the query ‘readAccessibleBy’ some ClearanceLevel1Role’ will return no results. Another example of a query is ‘readAccessibleBy some roleOf value UserName,’ which will return the list of GASHA from fields read-accessible by UserName.

This whole reasoning machinery will have to be replicated for all ClearanceLevelxRole and PermissionLevelyRole classes, which makes the development of the ACO ontology under this approach a very cumbersome affair; this, as a matter of fact, constitutes the main disadvantage of this strategy. Fortunately, as explained in fn. 3, the particular access control policy we had to implement via approach 2 is simpler than the one captured via the first approach, hence switching to the ‘property chains’ reasoning strategy amounted to no more than a minor inconvenience. Implementing more complicated policies however, may become prohibitive under this approach. Furthermore, extra care must be taken in the presence of object property chains, as they can quickly lead to undecidability complications if used in several mundane circumstances, such as in conjunction with disjoint classes etc.

ACO’s level of DL expressivity under the second approach is \(\mathbb{SROIQ}\); querying times are comparable with the first approach, meaning under 10 seconds using Pellet’s SPARQL engine on a run-of-the mill PC, such as the one described in fn. 4.

4. Palliative and Seniors’ Care Ontology (PCSO)

The main requirements of the knowledge base associated with the workflow management system under development by our group is to represent palliative and seniors’ care knowledge as exhaustively and accurately as possible, and in a way that lends itself to querying in real time. More specifically, the project’s intent is to pair the palliative and seniors’ care workflow with a knowledge base. This will interface with the workflow and provide domain-knowledge-based decision support; the ontology/knowledge base can thus be regarded as guiding and informing the workflow process using real-time access to dynamic knowledge. There is a trade-off between the requirements of exhaustiveness and real-time tractability, though a discussion of this matter will be postponed for later. In view of this conflict, we have chosen to construe and develop the workflow-associated palliative and seniors’ care knowledge base as a general theory of the palliative and seniors’ care domain, therefore avoiding any reference to individual patients, and individual patient records. It is not only that a lumping together of general and particular knowledge would have considerably enlarged the ontology—thus undermining prospects of real-time reasoning—but ontologies as knowledge representation tools do not particularly lend themselves to a dynamic treatment such as the one imposed by the fluid character of patient data. Thus, by itself, the
ontology will be unable to interact in any meaningful way with the workflow; in order to offer guidance to, and direct the workflow, the ontology will have to be supplemented by a proper patient database, containing the electronic health records (EHR) for every patient subject to the workflow.

The ontology must employ a rich vocabulary, covering a wide spectrum of terms, from human anatomy and clinical practice to nursing routines and administrative aspects, etc. Other constraints that have been imposed are:

- The ontology should be either developed natively in OWL DL, or should be easily translatable into OWL DL;
- The ontology should comply with the norms of ontological decorum established by the OBO Foundry ([14]);
- The ontology should conform with the guidelines established by Canadian healthcare governing bodies and regulatory agencies.

The venture of building such a controlled vocabulary not only that cannot ignore the previous work undertaken by biomedical ontologists and standardization bodies worldwide, but, due to the sheer size of the domain to be covered, cannot realistically be carried out without appeal to such work. The ideal scenario would be to import (via the owl:imports mechanism) a relatively stable, and sufficiently rich biomedical ontology (the ‘background’ ontology), and then add specific palliative and seniors’ care terms.

In light of this, for PCSO purposes we have decided to use SNOMED-CT almost entirely, as this is currently the only serious option as far as a medical terminology is concerned. For ontology development, the OBI MIREOT (“minimal information to reference external ontology terms”) import mechanism will also be employed ([9]), via its web-based Onto-Fox implementation ([10]).

As in the case of ACO, we have adopted BFO as upper-level ontology. While this is not the proper venue to debate the merits and drawbacks of embracing an upper-level ontology in general, a few words about our particular choice (BFO) are in order. As indicated at the beginning of section 3, BFO accommodates roles natively, which is a big plus at least from the point of view of ACO. One of the major reasons for the adoption of BFO is that, as mentioned in passing above, BFO has gained traction in IHTSDO/SNOMED-CT circles, based, in part, on the reliance of Gene Ontology (GO) on BFO. As such, founding the SNOMED-CT-based ACO and PCSO on BFO hardly deserves further motivation. Nevertheless, BFO also offers other advantages, such as a well-motivated philosophical background, a sound ontological structure, minimum interference with domain-level ('content') matter—i.e. genuine ‘upper-level behavior’—etc. Furthermore, besides SNOMED-CT, several other domain ontologies have made resolute steps towards integration with BFO—see, e.g., FMA [34], ChEBI [35], etc. Another example of ontology reuse is the import of the widely used BFO-based Relation Ontology (RO) [29], which will provide a kernel of relational terms for PCSO.

Besides using the terms collected manually by our researchers from various palliative and seniors’ care resources and clinical documents (primarily GASHA palliative and seniors’ care forms), components of the term pool used to construct PCSO (some of which have also been mentioned above) include:

1. The Ontology for Biomedical Investigations (OBI) ([8]): The OBI project is developing an integrated ontology for the description of biological and clinical investigations. The ontology targets the design of an investigation, the protocols, instrumentation and material used, the data generated and the type analysis performed on it. Currently OBI is being built under the Basic Formal Ontology (BFO), and is available in OWL format;

2. The International Classification for Nursing Practice (ICNP®), which is a unified nursing language system. It is a compositional terminology for nursing practice that facilitates the development of and the cross-mapping among local terms and existing terminologies. ICNP version 2 consists of only one root axis – Nursing Phenomena, which in turn has seven axes – Client, Focus, Location, Judgment, Means, Time, and Action. ICNP can be provided, upon request, in OWL format;

3. ACGT Master Ontology (ACGT MO) ([12]): The ACGT MO is an ontology that aims to represent the domain of cancer research and management, with special emphasis on mammary carcinoma (‘breast cancer’), Wilms’ tumor (nephroblastoma) and rhabdoid tumor; available in OWL format;

4. C. Kuziemsky’s Environmental Scan on the Minimum Data Set (MDS) for Palliative Care ([3]);

5. The Canadian Hospice Palliative Care Association’s Model to Guide Hospice Palliative Care: Based on National Principles and
Norms of Practice (‘the National Model’) [5];
6. The Canadian Council on Health Services Accreditation Hospice Palliative Care Standards (‘the Standards’) [4].

These components of the term pool were carefully inspected, and terms extracted and examined with an eye to their proper place in a taxonomy of the palliative and seniors’ care domain. Work in analyzing these data and knowledge bases is ongoing, and terms are extracted and processed in accordance with OBO Foundry and other general ontological principles, in order to secure a clean taxonomy.

Ontology development proceeds along two major development paths: terminology and reasoning, corresponding to two potential uses of the PCSO: (a) terminology alignment and standardization, and (b) workflow guidance. The former treats the PCSO as a controlled vocabulary, while the latter employs its full resources via the assertive knowledge (axioms, restrictions, taxonomy) contained within.

The desiderata of terminology alignment and biomedical knowledge wealth have led to the adoption of SNOMED-CT as a centerpiece of PCSO, and, to a lower extent, of ACO. In order to ensure a proper level of integration of SNOMED, we have initiated a re-grouping of SNOMED terms and branches under the BFO upper-level categories. The process of building the PCSO has thus placed a major emphasis on the (ongoing) enterprise of re-shaping and re-arranging of SNOMED-CT. To our knowledge, the only previous attempt in this regard is [23], which has been examined closely, and built upon: we have, for example, followed Hogan in subsuming SNOMED-CT’s ‘Procedure’ concept to a BFO ‘process’ (see #3 in the list below), and ‘Organism’ to BFO:object. Briefly, some of the major moves that have been undertaken in this respect are the following:

1. Moved several classes from SNOMED’s ‘Observable entity (observable entity)’ top-level branch to BFO’s ‘quality’;
2. Moved several classes from SNOMED’s ‘Function (observable entity)’ to BFO’s ‘function’;
3. Moved the ‘Procedure (procedure)’ class to BFO’s ‘process’;
4. Moved ‘Legal intervention (event)’ to BFO’s ‘process’;
5. Moved ‘Death (event)’ to BFO’s ‘process_boundary’;
6. Populated BFO:material_entity with classes from SNOMED-CT (e.g. ‘Specimen (specimen),’ ‘Substance (substance),’ etc.);
7. Populated PCSO’s ‘DeclarativeProposition’ branch with classes from SNOMED-CT.

Finally, even though SNOMED contains an impressive number of relations (‘object properties’), given the obscure character of many of them, we found it necessary to add many such relations manually: admittedTo, dischargedFrom, undertakes, undergoes, etc.

The effort to integrate ICNP deserves a special mention. Its almost 3,000 classes and object properties have been carefully scrutinized; to our surprise, a great deal of overlap between ICNP and SNOMED-CT has been uncovered, such that the great majority of ICNP terms are comfortably covered by SNOMED-CT equivalents. In the very few instances where an equivalent SNOMED-CT term has not been found, the corresponding ICNP term has been added to the ontology.

A great deal of time and energy has been invested in examining some of the most patently palliative-care-specific term lists included in the term pool, and ensuring that ontology-worthy terms are captured in the PCSO either under the very same label, or under a slightly modified sticker. In the very few instances where an equivalent term has not been found, we have added it to the ontology. Development is currently ongoing both by re-analyzing ontologically-problematic terms, and by examining the remainder of the term pool items.

PCSO interacts with the workflow in tandem with the patient database (EHR), given that the ontology only contains generic information, but no information pertaining to individual patients. As stressed above, PCSO will provide logic-based guidance for the workflow at the so-called decision points, which are to be suitably chosen from those points in the workflow where it branches, and where palliative and seniors’ care knowledge is involved in the decision. The sequence of actions is the following: (a) the workflow reaches a decision point; (b) the ontology is queried with the relevant patient data contained in the EHR, and furnishes information regarding the workflow branch that the process is supposed to follow; (c) the process follows the path indicated by the ontology query. Most of the time, however, the ontology output will be used to make recommendations for the system users, instead of simply prompting a certain mandatory conduct. Some of the reasons for involving an ontology in the decision process—rather than simply building the knowledge directly in the
workflow via the WfMS—are: (i) as opposed to re-writing the workflow (hence tinkering with the workflow engine), updating an ontology is a relatively simple task, (ii) the criteria involved in the decisions are, most of the times, cumbersome, knowledge intensive, and intellectually demanding, hence capturing them via workflow is quite costly in terms of the number of (repetitive) workflow branches required and, ultimately, system resources; representing these criteria as OWL rules in PCSO makes the utmost sense, especially since they are comfortably captured by the underlying logic of OWL 2.

We will provide in the following a concrete scenario where such interaction takes place, accompanied by the corresponding queries. Based on the information contained in one of the GASHA forms (the ‘Issues Log’), a patient may be referred to a number of professionals (e.g. Social Worker, Physiotherapist, Occupational Therapist, etc.) (figure 14).

The criteria according to which such professionals are chosen can be expressed as Manchester syntax restrictions; one such example is the following criterion: ‘Adult patients manifesting psychological symptoms such as fears and independence concerns, and whose primary diagnosis is not cancer, should be referred to a social worker, unless they are supposed to undergo emergency kidney surgery.’ The corresponding Manchester syntax expression/restriction is: Patient and (hasAge some (Age and (hasValue some int[≥“18”integer])) and (hasSymptom some (Fear or IndependenceConcern)) and not (hasPrimaryDiagnosis some Cancer) and not (hasPlannedOperation some EmergencyKidneySurgery)). Consequently, a defined class ReferredToSocialWork is defined as equivalent to the above restriction. Upon querying, the workflow system launches a query to the ontology containing the relevant data collected via the Issues Log: the patient age, the list of symptoms, patient’s primary diagnostic, and the list of planned surgeries. In effect, the query amounts to a satisfiability check, whereby the patient data is tested as to whether it complies with the above criterion or not. The ontology returns a Boolean value, according to which the patient in question belongs to the ReferredToSocialWork class or not. This will be treated by the workflow as a suggestion, since the decision ultimately belongs to the clinician currently interacting with the patient.

One final issue to be discussed in connection with the PCSO is the tractability aspect. Given the size of the ontology, consistency checking currently takes approximately three hours on a run-of-the-mill desktop PC using the latest Pellet version, and approximately 10 mins. using FaCT++. Querying is faster, though still prohibitive from the point of view of real time action. At this point in time, PCSO contains approx. 310,000 classes, 100 object properties, and its DL expressivity level is \( \mathcal{SROIF} \).

We are presently evaluating several methods in order to handle querying in real time, including:

- the relational database route—that is, exporting/storing the ontology in relational tables, and employing database techniques to carry out rea-
soning/inferring; such techniques have yielded promising results for some OWL 2 subsets such as OWLPrime ([15]);

- ontology modularization, i.e. extracting a “logically correct” module ([6], p. 15) capable of tackling the requisite queries, and using it “as a way to keep performance of ontology services at an acceptable level” (ibid, p. 7);
- parallel and distributed methodologies such as the MaRVIN platform described in [20], to get the advantages afforded by multithreaded and distributed reasoning.

The ACO is one example of modularization, as it has been spun off from the initially-projected all-encompassing ontology in order to avoid tractability issues.

5. Conclusions and Future Work

We have (a) outlined the access control policies and implementation principles that lay the foundation of the RBAC ontology built as part of a palliative and seniors’ care workflow project presently developed in Nova Scotia, and (b) given an overview of the palliative and seniors’ care ontology that guides the same workflow. A first pilot involving clinicians and patients in real-life palliative and seniors’ care setting is currently underway in GASHA. This project makes essential use of semantic web techniques, and constitutes a living expression of the efficacy of ontological knowledge bases and their employment in concrete everyday situations, and of the usefulness of standards such as SNOMED-CT. It is our belief that semantically-structured knowledge has yet to bear its fruit as a large-scale implementational venue for health informatics systems, and we view our present endeavor as a contribution towards streamlining efforts targeting massive adoption of such techniques.

The evolution of the interaction between the workflow system and the ACO includes a customization phase, which requires implementing a workflow mechanism that queries the patient/client on specific access control preferences during several predetermined phases of the workflow, plus a mechanism that builds new patient-specific access control ontologies that will be combined with the default ACO described above in order to customize the access control policy for each patient. Further, more sophisticated, access control scenarios may require implementing task-based access control policies; this, however, may require tools more expressive than DL fragments of OWL. Among others, we envisage implementing an emergency ‘break the glass’ mechanism, according to which certain system users can bypass security policies in case of emergency.

Development of the PCSO will progress under the guidelines detailed above: we will continue to inject more palliative and seniors’ care knowledge into it, particularly via enriching it with terms relating to the care aspect (as opposed to terms and knowledge pertaining to strictly medical issues), plus workflow-specific axioms and restrictions. The growth and fine-tuning of the PCSO will be essentially informed by the results and experience accumulated during the run of the first pilot. At the same time, our efforts will also be directed at coping with the tractability aspect, and possibly investigating new paths in this respect.

References


[35] http://www.ebi.ac.uk/chebi/


