GUIDING CRITERIA FOR BUILDING VALUABLE ONTOLOGY: EXAMINING SITUATION-BASED ACCESS CONTROL (SITBAC) ONTOLOGY IN THE LIGHT OF THESE CRITERIA

Beimel, Dizza, Ruppin Academic Center, Emek Hefer 40250, Israel, dizzab@ruppin.ac.il

Peleg, Mor, University of Haifa, Haifa 31905, Israel, morpeleg@is.haifa.ac.il

Abstract

*In recent years, **Access Control** became an important issue, as organizations are required to manage the confidentiality of the data they maintain. To the end, organizations need to define their policies regarding access to data. One of the interesting challenges in this context is how to represent the required knowledge for these data-access policies. One of the common techniques for representing knowledge is using an **Ontology**, in particular, we observe the progressively use of Web Ontology Language (OWL). As a result, a requirement for a methodology that will assist developers to build **valuable ontologies** arises.*

In this paper we define a set of twelve valuable criteria for ontologies, based on a literature search. The set of criteria include six criteria that are inherent to an ontology (e.g., reflection of reality and structure and uniformity). Six other criteria are derived from the ontology's usage (e.g., non-redundancy and consistency).

*We then examine the proposed criteria against an ontology that we previously introduced - **Situation-Based Access Control (SitBAC)** OWL ontology. SitBAC ontology is used for representing access-control policies for electronic medical records, where health organizations can specify their access regulations to patients' data according to the context of the request. On top of the SitBAC ontology, we provide an associated reasoning-based inference method, for real time reasoning about new incoming data-access requests.*

Keywords: Access-Control, Data-Access Policies, Knowledge Representation, Ontology, Web Ontology Language (OWL)

1 INTRODUCTION

The issue of **Access Control** (AC) turned to be important in the day life of many organizations, as a result of computerization process. In particular, Organizations are required to protect their data from been accessible by unauthorized entities, especially, if the data is considered to be sensitive (e.g., healthcare data). To this end, organizations should define their data-access policies. The policies are based on organization knowledge that we believe should be formally represented. In this paper we focus on **Ontology** representation and discuss ways to create a valuable one.

According to Van Bemmel and Musen [1] **Knowledge Representation** (KR) methods include a) First-order logics; b) If…then…else rules; c) Semantic Networks; d) Decision-theoretic models (e.g., Bayesian Networks, Decision trees); and e) Ontologies (e.g., Frames, Description logics). An ontology [2] is a machine-interpretable specification of a conceptualization-that is, the entities, presumed to exist in some area of interest, their attributes, and the relationships that hold among them. Ontology defines a common vocabulary for humans and machines that need to share domain information.

Many ontologies were created during the last years, in particular, ontology developers use Web Ontology Language (OWL) [3] to create their ontologies. Though the ontology principles are simple, still a methodology that will guide the ontology developers to create a **valuable ontology** is missing.

In this paper we introduce a set of criteria for valuable ontologies based on a literature search. The set of criteria include the following six criteria that are inherent to an ontology: (1) reflection of reality, (2) structure and uniformity, (3) definition of generalizations, (4) minimalism and non-ambiguity, (5) completeness and correctness, and (6) richness. Six other criteria are derived from the ontology's usage: (7) non-redundancy, (8) consistency, (9) easy maintenance, (10) uniqueness and originality, (11) scalability, and (12) use by real-life applications.

We then present **Situation-based Access Control (SitBAC)** [4] – an OWL ontology for representing and reasoning with access control polices in real-time that we developed following a thorough qualitative research of this domain. We demonstrate how the first nine of the valuable criteria are met by the SitBAC ontology, while the last three criteria are not.

2 VALUABE CRITERIA FOR ONTOLOGIES

We are interesting in defining valuable criteria with respect to ontologies. For that purpose, we searched the web [5-11] looking for valuable or beauty criteria, which relate to scientific domains (e.g., the mathematics domain, the information-systems domain). We integrated the search findings with our own ideas and finalized the following list of valuable criteria for ontologies. In the rest of this Section we present and explain the valuable criteria. After providing the background for the ontology of Situation-Based Access Control (SitBAC) that we developed in Section 3, we analyze our ontology, in Section 4, in the terms of the valuable criteria, demonstrating that it is a valuable ontology. Section 5 includes a discussion and conclusions.

A valuable ontology is an ontology that: (1) Reflects reality, (2) Is structured and uniform, (3) Defines generalizations, (4) Is minimal and unambiguous, (5) Is complete and correct (true), (6) Is rich, (7), Is non-redundant, (8) Is consistent, (9) Is easy to maintain, (10) Is unique and original, (11) Is scalable, and (12) Is used by real-life applications

We divided the criteria into two groups: the first group (1-6) address characteristics that are inherent to the ontology, while the second group (7-12) address characteristics that derive from the manner in which the ontology is used (e.g., a reasoner is used to infer new facts from the ontology).

2.1 Reflection of reality

The ontology has to reflect the reality of a specific domain. Thus, the knowledge maintained within the ontology has to be based on facts taken from the real world and needs to represent these facts accurately. The criterion is derived from the classic definition of an ontology, provided by Gruber [3]: "An ontology is an explicit, machine-interpretable specification of a conceptualization - that is, the

entities, or concepts, that are presumed to exist in some area of interest, their attributes, and the relationships that hold among them."

2.2 Structure and uniformity

The ontology has to follow a small number of structural rules. The rules have to be simple and should have an easy explanation. Following such rules provides a sense of uniformity to the eye of the observer, which leads to clarity and comprehension.

2.3 Generalization

An ontology, in its essence, maintains a collection of concepts that specifies the nature of a domain. An ontology may be tailored to describe a specific domain or it can be constructed in a way that enables simple integration of the ontology with another domains. The principles that enable integration should be described by guidelines that can be followed by ontology developers. Such an ontology should include general concepts, organized in hierarchies, where the roots of the hierarchies (i.e., the high-level concepts) represents the most generic concepts of the domain. The high-level concepts support the integration of the ontology with other domain ontologies by using these concepts as the basis for the creation of a new ontology that specializes the high level concepts and specifies the nature of another domain within the framework of the high-level concepts.

2.4 Minimalism and non-ambiguity

An ontology, by definition, reflects a certain domain of a real world. As the real world is made of "things", during an ontology's creation, the ontology's developer carries out an abstraction process in which he decides which real-world things to represent as ontology concepts, along with their properties and values, and which things to leave out of the ontology. According to Wand and Wang 11], "to be a good representation of a real-world system, the lawful states of an ontology should reflect the lawful states of the respected real-world system", where a *state* is a set of values assigned to the things (or assigned to the concepts in the respected ontology), at a given time. While creating an ontology, three possible problematic situations can happen: (a) the ontology misses relevant concepts (i.e., there exist real-world states that are not represented in the ontology at all), (b) the ontology does not make enough distinctions between concepts, so that situations that need to be classified as different states of the real world are mapped into a single ontology state, or (c) the ontology includes concepts that are not associated with any real-world state. The first problematic situation regards *completeness* (discussed in Section 2.5), the second problematic situation regards *ambiguity* [11], and the third problematic situation regards *minimalism*.

In an ambiguous ontology, several real-world states are mapped into the same ontology state, thus there is insufficient information to infer which state in the real world is represented and to make a correct inference. A minimal ontology is defined as an ontology that includes only those concepts that can be mapped into lawful states in the real world.

2.5 Complete and correct (truth)

Completeness of an ontology can be evaluated according to the expected usage of the ontology, which defines its scope. We focus on ontologies that are used as an aid for making decisions about action selection. This is in accordance with the behavior at the knowledge level, as it is defined by Newell [12]: "The agent processes its knowledge to determine the actions to take. The behavior law is the principle of rationality: Actions are selected to attain the agent's goals".

Thus, an ontology is complete with respect to a decision-making task if all the situations in the real world for which a decision is applicable can be represented in the ontology and a decision about an action selection can be inferred from the ontology for any of these situations. Our definition is scoped to the decision-making tasks that the ontology is scoped to support. Within this scope that defines the relevant lawful states of the real world that should be represented in the ontlogy, we follow the definition of Wand and Wang 11] where a complete ontology is an ontology that includes an exhaustive mapping of real world states to the ontology's concepts.

An ontology is non-conflicting if it does not contain garbling [11] of states, where a state of the real-world is mapped to an incorrect state of the ontology. When a state of the real world is mapped into more than one state of the ontology (which by itself is not considered problematic according to 11]) and the implications from these states is contradicting, we have a conflict.

An ontology is correct if the correct inference is made for all the possible situations for which a decision is applicable. For this to hold, an ontology is correct if and only if it is complete and non-conflicting. Completeness guarantees that a decision can be made for each applicable situation in real life. Non conflict guarantees that the decision response(s) that would be inferred from the ontology for a given situation would either be a single response or that the responses would not be conflicting.

2.6 Richness

A rich ontology is an ontology that makes a wide use of all the features that OWL provides and exhausts the entire range of the language capabilities to increase the expressive power of the ontology.

The following six criteria are derived from the usage of an ontology.

2.7 Non-redundancy

An ontology that reflects reality has to represent each relevant real-world "thing" via exactly one representation construct, thus, it has to be non-redundant. If redundant knowledge is found, it has to be removed from the ontology. Keeping a non-redundant ontology keeps the volume of the ontology small. When the ontology is updated or extended, the changes are carried out once instead of in different places. Accordingly, the number of potential human errors is decreased and this results in a clear and understandable ontology in the same way that the uniformity criterion affects the ontology.

2.8 Consistency

A consistent ontology [13] does not contain any contradictory facts. To maintain this feature, consistency checking should be performed. This checking feature is part of the standard inference services that are traditionally provided by a DL reasoner [13]. Within the consistency checking, the reasoner verifies the consistency with respect to (a) functional properties (e.g., hasBirthMother means that each relevant concept can only have one birth mother), (b) restrictions (e.g., existential restrictions), (c) class inconsistency, which is a side effect of the ontology classifying task, and more.

2.9 Ease of maintenance

An ontology should be easy to maintain. Maintaining the ontology means modifying and extending the ontology to represent new distinctions of reality that become relevant for the inference tasks for which the ontology is to be used. The ease is derived from providing a small number of guidelines for ontology maintenance, which are simple and easy to explain and follow. The ease of maintenance stems from the structure and uniformity of the ontology. The guidelines for ontology maintenance are written in a way that maintains the existence of the other valuable characteristics.

2.10 Uniqueness and Originality

A unique and original ontology has to introduce a new idea or a new approach in a surprising way.

2.11 Scalability

An ontology has to be scalable, thus, it has to be able to grow as much as required (by introducing new concepts) while keeping the optimal complexity of the activities that use the ontology.

2.12 Use by real-life applications

Developers create ontologies as part of their researches, as part of developing applicable information systems, as part of producing software artifacts, as part of generating artificial intelligent systems, etc. All the various developers share a common goal: that the work they carried out will become a real-life

applicable tool that serves the requirements of an organization and promotes its vision. According to this, we believe that an ontology that is part of a real-life application is a valuable ontology.

3 SITUATION-BASED ACCESS CONTROL (SITBAC) ONTOLOGY

Many organizations use confidentiality protection mechanisms, which are based on **Role-Based Access Control (RBAC)** model [14], whose main principle is to grant permission to an entity that requests the access to data, based on the entity's organization role. The RBAC model is considered as a simple and an elegant model. However, its expressive power is limited. Therefore, there may exist scenarios of data-access request that are required according to the organization regulations but cannot be supported or represented via the RBAC model.

In order to overcome this expressive power limitation, we proposed our model: **Situation-Based Access Control (SitBAC)** model [4]. SitBAC is a healthcare-oriented conceptual model that enables to formally represent data-access scenarios. We developed SitBAC after a thorough qualitative study of various data-access scenarios in healthcare organizations, where we collected by structured interviews close to two hundred scenarios of data-access requests and analyzed their characteristics.

We then implemented the conceptual model of SitBAC as an OWL ontology. The SitBAC OWL ontology aims (a) to enable defining **data-access policies** by representing situations that describe prototypical institutional settings in which access to patient records should be granted to a requesting agent, (b) to **share** the data-access policies among several organizations by using the support of OWL in interoperability, and (c) to serve as a knowledgebase for a **reasoning process**, concerning an incoming data-access request. The reasoning process involves a realization-based method and infers an 'approved/denied' response for the incoming request based on the defined data-access policies.

Though SitBAC was created for the domain of healthcare, the principles are generic and can be used as a basis for creating access control policies for other domains, as well. We have applied the design patterns used in our SitBAC ontology to other domains such as authorizing credit card requests or attending to student discipline problems [15].

The SitBAC OWL ontology consists of three components: the *SitBAC Vocabulary*, the *Situation Classes*, and the *Situation Individuals*. The following is a detailed description of each component. The ontology was implemented in OWL using Protégé [16].

3.1 The SitBAC Vocabulary

The SitBAC Vocabulary is composed of concepts representing various characteristics of data-access scenarios that we elicited from our observational study. Based on our findings, the defined concepts were arranged in three fundamental categories, as shown in Fig. 1: (1) Entities, (2) Refineables (i.e., attributes of the Entities), and (3) Relations between Entities or between Refineables. On top of these three categories and their related concepts, we defined the heart of our model – the **Situation**, a formal representation of a data-access scenario. The Situation is a pattern consisting of six Entities, along with their Refineables and Relations, as shown in Fig. 1.

Each of the fundamental categories is represented in our ontology as a super-class. All the concepts that belong to a category are defined as its subclasses (creating classification hierarchies) or as individuals of the subclasses (defining enumerated classes). The following three paragraphs provide more details on each of the super-classes. The Situation is discussed in the next subsection.

Entities: we represented the following six Entities as subclasses of the `Entity_Concept` super-class:

1. `Data-Requestor` is the entity requesting access to the patient's data.
2. `Patient` is the entity who is the subject of the requested data.
3. `EHR` (Electronic Health Record) is the record where the patient's data is maintained.
4. `Task` is the operation on the data that the Data-Requestor wishes to carry out. In particular, the Task refers to an `Action` and an `EHR-section` (e.g., view medications, update prescriptions).
5. `Legal-Authorization` is a legal document authorizing the requested Task (e.g., a caregiver requests access to a patient's hospital record to view its discharge letter for a follow-up procedure).
6. `Response` is the data-access decision, which may be `Approved` (granted) or `Denied`.

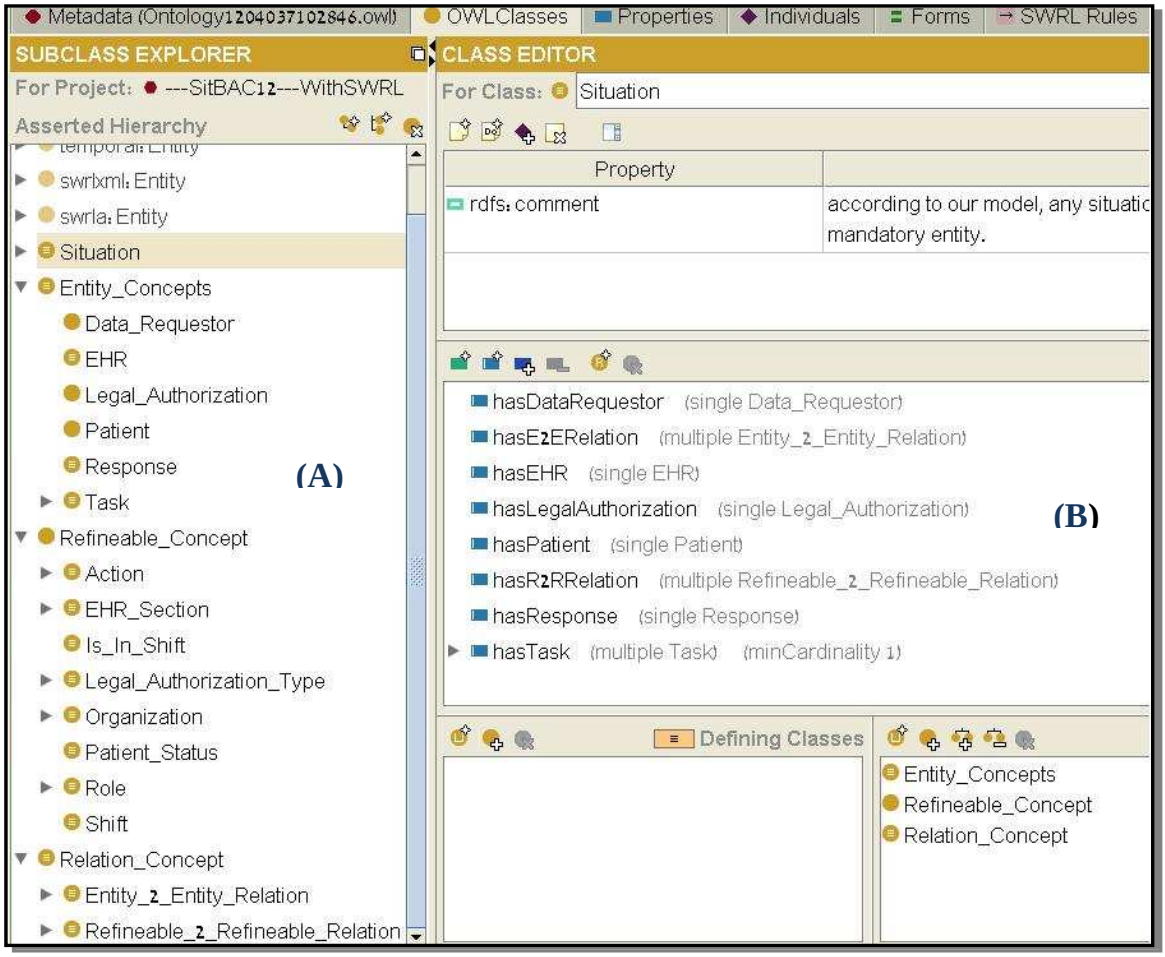

Fig 1: A screen shot of the SitBAC ontology, implemented via Protégé [16], including the concept classes (A), and the properties of the of the Situation class (B)

Refineables are attributes of the Entities or the Refineables. Refineables can be represented in three possible ways. In the first way, Refineables are represented as sub-classes of the `Refineable_Concept` super-class (e.g., the `location` of the Data-Requestor, which is a subclass of the hierarchy of `Medical_Site`, and the `Role` of the Data-Requestor, which is a subclass of the `Role` hierarchy). In the second way, Refineables are represented as an enumerated set of individuals (e.g., the class `Patient_Status` is enumerated by the individuals `inpatientStatus`, `outpatientStatus`, and `nursingStatus`; `Yes` and `No` are the possible enumerations of the `Is_In_Shift` subclass). The third way to represent Refineables is using data-type properties (e.g., the `hasAcccessTime` property of the Data-Requestor has `Date` as its range).

The Entities relate to the first two types of Refineables via object-properties (`hasRole` and `hasEmployer` for the Data-Requestor and `hasInsurance` and `hasLocation` for the Patient). Some Refineables are attributes of other Refineables (e.g., the `Hospital` subclass relates to its various departments via `hasHospitalDepartment` object property).

Relations can be established between two Entities (E2E-Relation) or between two Refineables (R2R-Relation) and are represented as subclasses (`E2E_Relation` and `R2R_Relation`) of the `Relation_Concept` super-class. The different types of Relations were grouped and each group is represented as a subclass. For example, the `Location_Relation` and the `Organization_Relation` are subclasses of the `R2R_Relation`. The specific E2E and R2R Relations that may exist are represented as individuals of these Relation classes (e.g., the individual `R2Rl.DR.Location.IsEqualTo.Patient.Location` represents the fact that the Patient and the Data-Request are located at the same place. We used a naming convention to ease the user understating of the various individual Relations.

3.2 The Situation and the Situation Classes

The Situation is represented in the ontology as a super-class named Situation. Recall, it structures a data-access scenario into a formal representation. The Situation's properties relate to the Entities, along with their Refineables and Relations, all taken from the SitBAC vocabulary. The properties include:

- `hasPatient` – a functional property whose range is the Patient entity.
- `hasDataRequestor` – a functional property whose range is the Data-Requestor entity.
- `hasEHR` – a functional property whose range is the EHR entity.
- `hasLegalAuthorization` – a functional property whose range is the LA entity.
- `hasTask` – a functional property whose range is the Task entity. Each Situation can describe many tasks (at least one), where each task refers to a specific action on a specific EHR-section.
- `hasResponse` – a functional property whose range is the Response entity, which either approves or denies access to data.
- `hasE2ERelation` – a non-functional property whose range is the E2E Relations.
- `hasR2RRelation` – a non-functional property whose range is the R2R Relations.

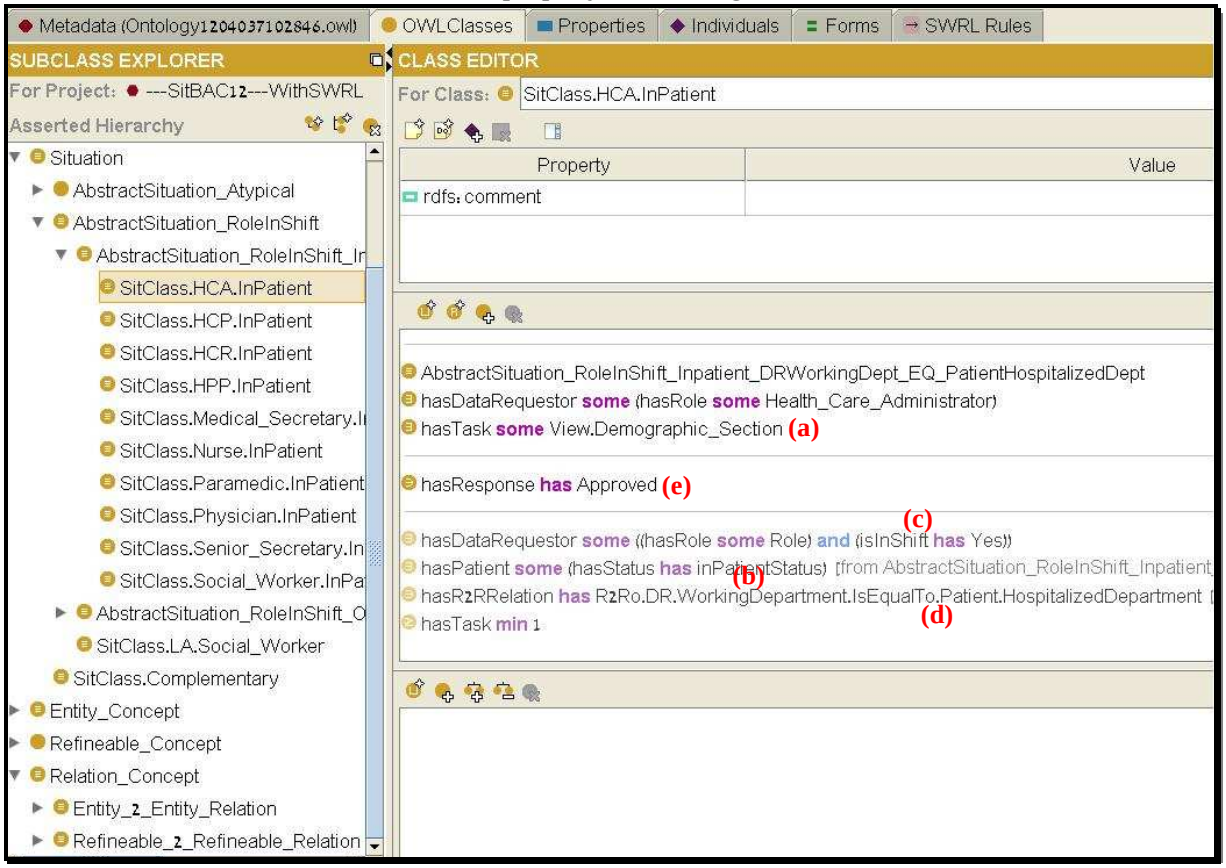

Fig. 2: *A screen shot of a particular SitClass named SitClass.HCA.InPatient. The SitClass is a subclass of the Situation class and has asserted conditions.*

Part B of Fig 1 demonstrates the object properties of the Situation super-class. In the OWL ontology, each specific data-access scenario is formalized into a **Situation Sub-Class** or **SitClass** for short, which is a descendant of the Situation super-class (see Fig. 2). The various SitClasses inherit object properties from the Situation super-class and in addition hold asserted conditions to represent different types of restrictions. For example, all SitClasses include the necessary & sufficient (n&s) condition: ≥`hasTask min 1`, which expresses the requirement that each SitClass has to include at least one

Task. An example for a SitClass is presented in

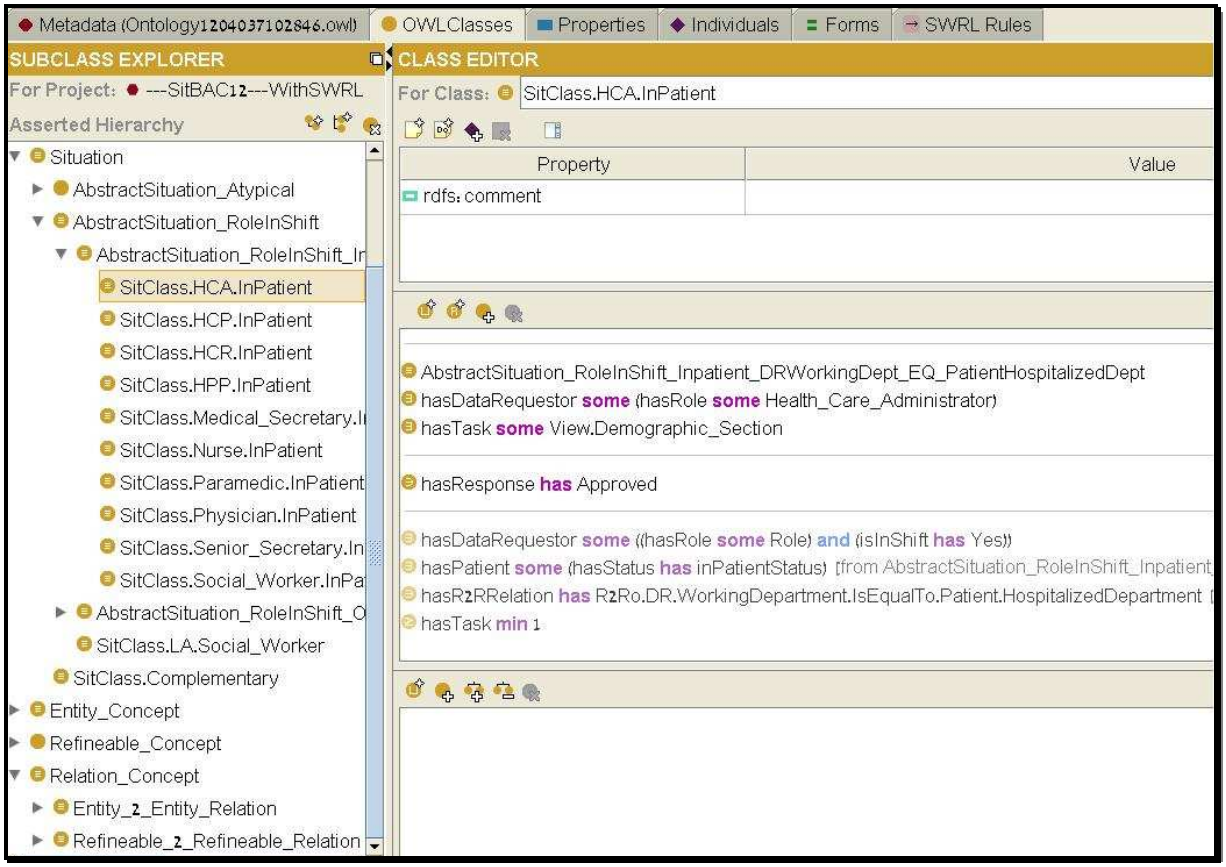

Fig. 2: HealthCare Administrator (HCA) requests to view the demographic section of a patient's record (a), in case the patient is hospitalized (`inPatientStatus`) (b), the HCA is in shift (c), and the patient is hospitalized at the same working department of the HCA (d). The response is `Approved` (e).

One of our design principles is that all the SitClasses define Situations in which access should be approved (the value of `hasResponse` is set to `Approved`). The Situations are defined using necessary and sufficient axioms. There is a single class that is complementary to all the 'approved' SitClasses, which denies access in all other scenarios. This class is defined as the negation of all other SitClasses. Its necessary implication is to deny access (the value of `hasResponse` is set to `Denied`).

3.3 The Situation Individuals

Situation Individuals are members of the Situation super-class. In the SitBAC ontology, we use the Situation individual to represent an incoming data-access request. Such individuals are realized as members of particular SitClasses, using reasoners, such as Pellet [13]. If an individual is realized into any of the SitClasses (excluding the complementary class) then the inferred response is `Approved`. The Situation individual is generated as an input for the decision process and is removed when the decision process with respect to the above input ends.

Realizing a Situation individual to a single SitClass requires *closed-world inference*. Closed-world inference is necessary when an explicit response (`Approved/Denied`) is required. Closed-world is not naturally supported by OWL, as OWL supports open-world reasoning. For that purpose, we developed ontology design patterns that result in a closed-world ontology. A full description of our list of design patterns can be found in [17].

4 REVIEWING SITBAC WITH RESPECT TO THE VALUABLE CRITERIA

In this Section we explain how SitBAC addresses the valuable criteria. Criteria that SitBAC does not meet are only discussed in Section 5.

4.1 The ontology is a reflection of reality

Our OWL ontology is based on the outcome of a thorough qualitative research that lasted over than a year [4]. Within this research, we used acceptable qualitative research methods to elicit and analyze the required data, mainly interviews. Every interview lasted between 30 minutes to one hour. We interviewed different stakeholders who were involved in access-control scenarios, such as: a family doctor, an expert physician, two nurses, a social worker, a dietitian, an insurance person, medical information-systems experts, and patients. The interviews yield 197 scenarios describing requests of access to patients' data. We used a qualitative method to analyze the transcribed interviews. The three fundamental categories that we identified were: Entities, Refineables, and Relations.

In the next step of our research, we defined a common structure to the collection of scenarios by using a graphical information system analysis and design methodology called Object Process Methodology (OPM) [18]. OPM allowed us to have a visual representation of the model that we were formulating, which helped us conceive it. Using OPM, we structured the scenarios into Situations. The Situation is the heart of our model. The Situation is a formal specification of a generic scenario, composed of Entities, Refineables and Relations. In order to specify a specific scenario, we defined a Situation Instance, which inherits the structure of the Situation and includes Refineable allowed-values.

The design of the SitBAC ontology reflects the above description of the conceptual model. A SitBAC concept is an ontology class, an ontology individual, or a data-type property (e.g., `hasAccessTime`). The classes are arranged in three main hierarchies, whose super-classes are: `Entity_Concept`, `Refineable_Concept`, and `Relation_Concept`. The individuals represent some of the allowed values (e.g., `Hospitalized` is an individual of the `Patient_Status` class) while the other allowed values are defined as subclasses (e.g., a `Secretary` is a subclass of the `Role` Refineable class). The Situation is represented as a Situation Super-Class, and the Situation Instances that represent particular data-access request scenarios are represented as Situation Sub-Classes (SitClasses for short).

Based on the methodological process that we followed in order to construct the SitBAC ontology from interviews of stakeholders, we can state that the SitBAC ontology reflects a reality concerning the healthcare domain, in particular healthcare data-access scenarios.

4.2 The ontology is structured and uniform

The SitBAC vocabulary is composed of classes and individuals. The classes are organized as descendants of one of the three super-classes (Entity, Refineable, and Relation).

In order to define the closed sets of allowed values for class properties, the SitBAC vocabulary uses two design patterns: (a) a value-partition design pattern [19] (e.g., the `HCA` Role class is partitioned into its subclasses `Medical_Secretary`, `IS_Administrator`, and `Ordinary_HCA`) and (b) an enumerated-class design pattern [19] (e.g., the class `Is_In_Shift` contains only its two possible individuals: `Yes`, `No`). We use value partitions to partition concepts that can be further specialized into subclasses and we use enumerated classes for simple categorical values. The use of these two design patterns resulted in a uniform and a well-structured ontology.

The structured and uniform feature enables to easily extend the SitBAC vocabulary. The extension ability is part of the maintenance ability, which we refer to as an additional valuable criterion. The ease of maintenance criterion is discussed in subsection 4.9.

4.3 The ontology can be generalized

The SitBAC ontology was created for representing and reasoning with access control policies in the domain of healthcare. Although there exist in the SitBAC ontology some principles, structures and

guidelines that can be followed to apply it to another domain, still an intermediate layer is missing in order to state that the SitBAC ontology is general enough to enable a quick integration with other domains. However, we believe that extending the SitBAC ontology to support this intermediate layer can be easily carried out. We now describe our suggestion for constructing the intermediate layer:

- The super-classes of the ontology (Entity, Refineable and Relation) should remain without changes. The intermediate layer along with the super-classes establishes the "high-level classes".
- Some of the defined entities are general (Data_Requestor, Response, Task and Legal_Authorization) and some require generalization (the Patient should be a subclass of a new general class named Subject, and the EHR should be a subclass of a new general class named Resource).
- Some of the defined Refineables are general (Role, Organization, Shift, Action) and some require generalization (Patient_Status should be a subclass of a new general class named Subject_Status and EHR_Section should be a subclass of a new general class Resource_Section).
- All the Refineables that are specific-domain (e.g., the Is_In_Department refineable defined for the healthcare domain) should be added by the ontology creator.
- The Relations are grouped into two subclasses: Entity_2_Entity_Relation and Refineable_2_Refineable_Relation. This structure is general. New domain-specific Relations will be added as individuals of these subclasses by the ontology creator.

Such an extension can be easily used for representing an access control policy of another domain, for example an access control policy of an academic organization: A Student class should be created as a subclass of the Subject, the Student_Record should be a subclass of the Resource class, etc.

To summarize, the SitBAC ontology is not general enough as it is. However, a simple extension could convert the ontology to be general enough for representing other domains as well.

4.4 The ontology is minimal and non-ambiguous

When examining the SitBAC ontology with respect to minimalism, we need to validate that there not exist a vocabulary concept that we do not use within our SitClasses, which formulate the access-control policy. In such case, we can deduce that the concept is not required for the ontology's refection of the domain. After examining our ontology we can state that it is minimal.

In addition, we need to examine our ontology with respect to the non-ambiguity criterion. After checking the mapping of the various scenarios that we collected during the qualitative stage into different SitClasses, we can state that our ontology is non-ambiguous.

4.5 The ontology is complete and correct (true)

Completeness: According to the completeness definition, an ontology is complete if a decision about action selection can be inferred from the ontology for each situation in the real world, which is also represented in the ontology. With respect to this definition, our ontology is complete. The completeness feature derives from the ontology's design. The ontology's design guidelines are to create a SitClass for every permitted access-request scenario (i.e., a SitClass whose `hasResponse` property has the value `Approved`) and one SitClass to deny all the rest (i.e., `hasResponse` has the value `Denied`). We name the latter *Complementary* class. Fig. 3 presents the Complementary class.

The complementary class negates all the 'approved' SitClasses; thus, when the reasoner tries to realize a Situation individual into one of the SitClasses, it will realize the individual into the Complementary class only if the individual cannot be realized into one of the 'approved' SitClasses. In such case, the individual inherits the `hasResponse` value of the Complementary class, thus, access is denied. Using this design, a Situation individual will always be realized into one of the Situation subclasses.

Non-Conflicting: The non-conflicting feature derives from its design. The response is `Approved` for all SitClasses except for the complementary class. As a Situation individual would always be realized into either a SitClass or the Complementary class, it would never have conflicting responses.

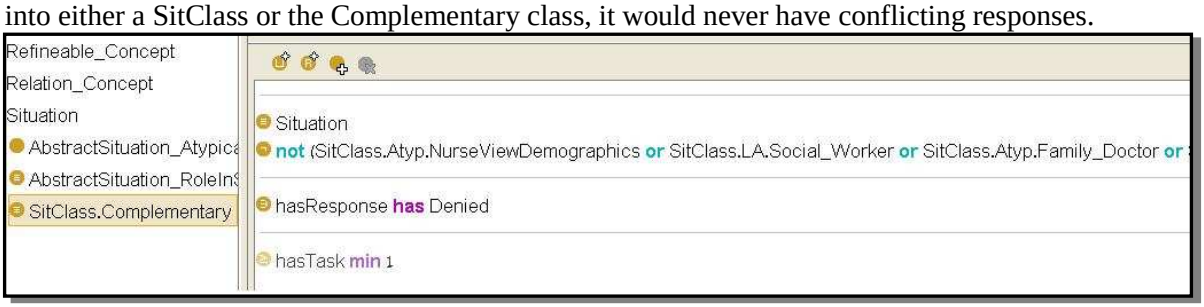

Fig. 3: a screen shot from Protégé showing the Complementary Class

Correctness: Because the ontology is complete and non-conflicting it follows that it is correct.

4.6 The ontology is rich

The SitBAC ontology makes a wide use of the features that OWL provides. The basic features that we used were to structure the SitBAC concepts into hierarchies and to create properties and individuals. These basic features are provided also by the frames ontology version. In addition, we used advanced features of OWL such as: (a) creating primitive and defined classes using necessary and sufficient restrictions, (b) defining functional and non-functional object properties and stating the range and the domain of these properties, (c) constructing all types or restrictions: Quantifier restrictions (i.e., Existential and Universal), Cardinality restrictions and hasValue restrictions, (d) we used the features of closure axioms, value partitions, covering axioms, disjoint classes, and enumerated classes, (e) we created necessary implications in some of our classes, and (f) we took advantage of almost all the services that are provided by the DL-reasoner: consistency checking, classification, and realization.

4.7 The ontology is non-redundant

We maintain a non-redundant ontology. The SitBAC vocabulary represents the healthcare domain characteristics without repetitions. The set of SitClasses does not include redundant classes as we execute the reasoner after adding a new SitClass to the ontology to check whether the new SitClass is a subclass of an existing SitClass, or, an existing SitClass is a subclass of the new SitClass. If so, the subclass is eliminated, since the more general class already grants access to data. The non-redundancy feature results in decreasing the risk for errors as the set of SitClasses is as small as possible.

4.8 The ontology is consistent

The ontology does not contain any contradictory facts, as we executed the consistency checking after introducing a new concept to the SitBAC vocabulary or after updating an existing concept. If the reasoner notifies an inconsistent state in the ontology, we fixed the problem.

4.9 The ontology is easily maintained

Ontology maintenance is composed of two major tasks: (a) updating the existing knowledge and (b) extending the knowledge by adding new knowledge entries. These maintenance tasks refer to two main components of our ontology: (1) the SitBAC vocabulary and (2) the set of SitClasses.

When it is evident that the ontology does not accurately reflect something in the real-world or (in our case, it may be a data-access policy (i.e., SitClass) or some vocabulary concept to which the policies refer, the ontology administrator needs to locate the problematic class and update it. When updating the ontology, a modular design that does not include redundancy, supports reuse, and is easy to comprehend, allows for easy update. More details regarding the ways in which locating classes and updating them can be easily done can be found in [17].

5 DISCUSSION AND CONCLUSIONS

In this paper we defined a set of twelve criteria that make an ontology valuable. As we discuss in the following paragraphs, SitBAC meets the first nine of these criteria.

SitBAC is an ontology that formally represents scenarios of data access. Each SitClass represents a specific scenario in which access should be granted and a class that is complementary to all other SitClasses denies access. We use an OWL reasoner to realize an incoming data-access request into one of the SitClasses or to the complementary class and infer the value of the response property of these classes: approved or denied. The rules for translating the conceptual model into the OWL ontology produced a well-structured and uniformly designed ontology that is rich, minimal, non-ambiguous, complete, non-conflicting, and hence, correct. We later formulated these rules as guidelines that are used by ontology developers to maintain the ontology such that it retains the above-mentioned criteria.

The same principles and design patterns that guide us in maintaining the SitBAC ontology can be used to develop policy-based ontologies in other domains. Of course, the adaptation to other domains needs to be done following a comprehensive study and literature search of the other domain. Recently, we gave our guidelines for creating and maintaining the SitBAC ontology to several undergraduate students in an elective knowledge representation course that the second author was teaching during the Fall semester of 2009. Based on SitBAC's design patterns, the students formulated several policy-based decision support systems that are similar to SitBAC in other domains, including (1) a system that decides on which pest affected crops based on symptoms that appear on crops and descriptions of the pest found, (2) personalizing a visit to a museum based on user profiles, (3) matching user constraints with tourist plans, (4) classification of sentences containing negation into sentence patterns, and (5) classifying the type of crime based on evidence collected. While all of these ontologies followed the design principles that we used in SitBAC and duplicated the structure of SitBAC, they did not extend SitBAC's generic classes. Future work will examine whether SitBAC could be extended to support these ontologies development by extending SitBAC through its generic high-level classes.

Limitations: The last three valuable criteria are not met by SitBAC as well as the first nine, as we explain below: (10) *the uniqueness of SitBAC*. The traditional way to implement access control is to create database queries that comply with the RBAC model. However, RBAC is limited in its expressiveness power. There exist other policy languages that extend RBAC, for example, XACML [20], using XML as its representation format, Trust Policy Language [21], and works that support context characteristics [22-24]. Some of these policy languages are defined on top of a robust formalism (e.g., Logic programming or Description Logics). Such policy languages are considered as 'well-defined semantics' languages according to [25], who define this feature as "the language ability to be independent of its particular implementation". The SitBAC OWL ontology can be considered as a "well-defined semantics" language, but cannot be considered as unique, though the number of policy languages that are DL-Based languages is small [25]. However, our work uses OWL in a way that infers the response for an incoming data-access request in real-time. Other works [26-28] do not use the reasoner for providing a real-time response for an incoming data-access request as we do. Thus, the reasoning usage that we present is unique. (11) *Scalability*: As our AC mechanism works with existing reasoners, the complexity of the reasoning is exponential (ALCON(D)). However, Horrocks et al [28] tested the reasoner performance as a function of the knowledge-base size and found that the actual performance was better than the theoretical one. (12) *The real-life use* of our ontology has not yet been assessed. However, the ontology has been tested with a set of 29 SitClasses corresponding to scenarios elicited from Carmel Medical Center and 24 Situation individuals.

Finally, a well- known phrase says: "Beauty is in the eye of the beholder". One can say that the list of valuable characteristics presented in this paper is partial.

References

[1] van Bemmel, J.H. and Musen, M.A. (1997). Handbook of Medical Informatics. Springer.
[2] Gruber, T.R. (1995).Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Human-Computer Studies, 43, 907-928.
[3] Web Ontology Language (OWL), http://www.w3.org/2001/sw/WebOnt/
[4] Peleg, M. Beimel, D. Dori, D. and Denekamp, Y. (2008). Situation-Based Access Control: privacy management via modeling of patient data access scenarios. Biomedical Informatics, 41(6), 1028-40.
[5] Hardy, G.H. (1956). A Mathematician's Apology. The World of Mathematics, 7, 2027-2040.
[6] Dienes, Z.P. (1971). Building Up Mathematics. 4th Edition. Hutchinson, London.
[7] Birkoff, G.D. (1956). Mathematics of Aesthetics. The World of Mathematics, 4, 2185-2195.
[8] Hadamard, J. (1945). The mathematician's mind: The physiology of invention in the mathematical field. Princeton University Press, Princeton, NJ.
[9] Gardiner, C. (1983). Beauty in Mathematics. Mathematical Spectrum 84(16), 78-84.
[10] Dreyfus, T. and Eisenberg, T. (1986). On the aesthetics of mathematical thoughts. For the Learning of Mathematics, 6 (1), 2-10.
[11] Wand, Y. and Wang, R. (1996). Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM, 39 (11), 86-95.
[12] Newell, A. (1982). The knowledge level. Artificial Intelligence, 18, 87-127.
[13] Sirin, E. Parsia, B. Grau, B.C. Kalyanpur, A. and Katz, Y. (2005). Pellet: A Practical OWL-DL Reasoner. Web Semantics, 5 (2), 51-3.

[14] Sandhu, R. Ferraiolo, D. Kuhn, R. (2000). The NIST Model for role-based access control: toward a unified standard. In Proc of the 5th ACM workshop on role-based access control, p. 47-63.
[15] Peleg, M. and Beimel, D. (2010). Using closed-world reasoning with OWL to implement policy-based classification. In Proc of the 3rd Conf on Model-Based Systems Engineering, Fairfax, Virginia.
[16] Knublauch, H. Fergerson, R. Noy, N.F. and Musen, M.A. (2004). The Protege OWL Plugin: An Open Dev. Env. for Semantic Web Applications. In Proc of the 3rd Intl Semantic Web Conf, Japan.
[17] Beimel, D. and Peleg, M. (2011). Using OWL and SWRL to represent and reason with situation-based access control policies. Accepted for publication in Data & Knowledge Engineering.
[18] Dori, D. (2002). Object-Process Methodology - A Holistic Systems Paradigm. Springer Verlag, Berlin, Heidelberg, New York.
[19] Horridge, M. Knublauch, H. Rector, A. Stevens, R. and Wroe, C. (2004). A Practical Guide to Building OWL Ontologies Using the Protege-OWL Plugin and CO-ODE Tools. Univ Of Manchester.
[20] OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
[21] Herzberg, A. Mass, Y. Michaeli, J. Ravid, Y. and Naor, D. (2000). Access control meets public key infrastructure, or: Assigning roles to strangers. IEEE Symposium on Security and Privacy.
[22] Motta, G. Furuie, S. (2003). A Contextual Role-Based Access Control Authorization Model for Electronic Patient Record. IEEE Transactions on information technology in biomedicine, 7, 202-207.
[23] Wang, L. Wijesekera, D. Jajodia, S. (2004). A logic-based framework for attribute based access control. Proc ACM workshop on Formal methods in security engineering, 45–55.
[24] Carminati, B. Ferrari, E. Perego, A. (2009). Enforcing Access Control in Web-Based Social Networks. ACM Trans Inf Syst Secur Article 6, 1-38.
[25] De-Coi, J.L. and Olmedilla, D. (2008). A Review of Trust Management, Security and Privacy Policy Languages. SECRYPT.
[26] Finin, T. Joshi, A. Kagal, L. Niu, J. Sandhu, R. et al. (2008). ROWLBAC - Representing Role Based Access Control in OWL. In Proc of the 13th Sym on AC Models and Technologies, p. 73-82.
[27] Bradshaw, J. Uszok, A. Jeffers, R. Suri, N. Hayes, P. et al. (2003). Representation and Reasoning for DAML-Based Policy and Domain Services in KAoS and Nomads. In proc of the Intl Conf Autonomous Agents.
[28] Kagal, L. Berners-Lee, T. Hendler, J. (2009). The Semantic Web and Policy. Web SeM, 7, 1-56.
[29] Horrocks, I. Sattler, U. Tobies, S. (2000). Practical Reasoning for Expressive Description Logics. Logic Journal of IGPL, 8 (3), 239-63.