March 2025

SULO – a simplified upper level ontology

Abstract.
Ontology-based data integration is a fundamental challenge in the life sciences and health care, particularly given the proliferation of heterogeneous datasets and the need for semantic interoperability. While upper-level ontologies provide a formal foundation, they are often perceived as too complex for domain experts and standards developers lacking formal training in logic and ontology. In this paper, we argue for a *Simplified Upper-Level Ontology* (SULO) that retains a minimal set of foundational categories and relations, facilitating intuitive mapping across biomedical terminologies, ontologies and schemas. We examine the limitations of existing upper-level ontologies and propose a streamlined approach to bridging data standards in the biomedical domain.

Keywords. upper level ontology, semantic interoperability, ontology-design patterns, schema design

1. Introduction

1.1. Background

The need to capture knowledge in a standardized representation has led to the development of domain-specific terminologies and ontologies. In the past three decades, the evolution of the discipline of Applied Ontology has stimulated the development of foundational or upper-level ontologies (ULOs), which provide a general theory of *what exists*, to be used as an overarching axiomatic foundation for domain ontologies (DOs), which express *what exists in a specific domain*. When implemented in formal logic, ULOs enable automated reasoning over the content of DOs and over data, whose semantics is given by a specific DO. ULOs such as BFO [1], UFO [2], GFO [3], DOLCE [4], and domain upper level ontologies such as SIO [5] and BioTop [6], play a significant role in guiding the development of DOs and domain schemas such as SPHN [7] and CARE-SM [8], as they instruct domain specialists in how domain terminology can be formalized in an ontology and also detail the building blocks by which classes can be reasoned about using automated reasoning systems.

1.2. Limitations in ULOs

Formalized ULOs present several barriers to adoption by domain experts, including:

- They offer philosophical guidance on how to partition a domain and generally provide the building blocks (classes and relations) to extend the ULO with their own formalized ontology. However, understanding subtle philosophical considerations required by its correct modeling, needs familiarity with logic and philosophy, which domain experts often hesitate to acquire.

- Some ULOs posit fundamental categories, e.g. *Object*, *Process*, *Event*, *Function*, *Quality*, *Disposition* whose labels carry implicit assumptions that differ not only between ULOs but also between domain experts, leading to misunderstandings and unintended models.
- Other ULOs intentionally use unfamiliar or technical labels inspired by philosophical ontology, such as *Continuant*, *Endurant*, *Perdurant*, *specifically dependent continuant*, etc. Although this avoids the ambiguities brought about by the previously mentioned issue, domain experts often hesitate to understand them properly, which again compromises the objective of reaching a common understanding.
- Domain ontologies such as SNOMED CT [9] often come with their own upper level, which reflects a domain view influenced by language domain-specific conceptualizations, e.g. *Clinical Finding*, *Observable*, *Qualifier Value*. These categories reflect the legacy of bottom-up evolution. Some of them are vague, lack definitions, are difficult to translate to other languages and do not even reflect an agreement between domain experts.
- Healthcare ontologies (e.g., SNOMED CT) and schemas (e.g. FHIR, SPHN) address the need to structure data graphs to conform to particular shapes. Typically they abound of specialized but nevertheless vague relations of the form `hasRangeType` such as `hasPatient`, with their range constrained by the target class. They are not linked to relations from ULOs.

2. Proposal: A Simplified Upper-Level Ontology (SULO)

The proposed solution for semantic integration of healthcare data is SULO (*Simplified Upper-Level Ontology*), a minimalistic approach to ontology design and reuse. The goal is to balance formal rigour with practical usability by reducing the number of upper-level classes and relations to an absolute minimum while maintaining compatibility with existing ULOs, domain ontologies and hybrids that fuse elements of terminologies, ontologies and schemas. We hypothesize that a minimized ULO will foster adherence and interoperability by domain specialists, particularly regarding biomedical science and healthcare. SULO is expected to reconcile expressivity with practical usability addressing the following principles:

- **Minimalism**: SULO proposes a small taxonomy of disjoint classes and a minimal set of constrained relations to ensure broad applicability across domains.
- **Compatibility**: SULO maintains compatibility with core components of well-known ULOs while remaining accessible to domain experts. It does not claim to compete with established ULOs but represents an umbrella above established ULOs.
- **Accessibility**: SULO aims to be accessible to non-ontologists through intuitive labeling, a simpler tree, and an general overview that fits in a single diagram.
- **Composability**: The building blocks of SULO enable domain experts to describe the attributes and relations of a domain class through a core set of design patterns.
- **Interoperability**: SULO promotes interoperability by enabling the description of classes from different ontologies to be merged or used together, and to allow the lossless transformation from a given representation to SULO.

- **Data validation**: SULO constrains real-world knowledge graphs through schema-based constraints and automated reasoning.

These principles assure a flexible framework for aligning and integrating biomedical data standards without imposing excessive, controversial ontological commitments that domain experts are mostly unaware of.

3. Methods

We followed the NEON methodology [10], consisting of i) domain analysis, ii) requirement gathering, iii) development of modular and pattern-based designs, iv) alignment with standards and existing ontologies, iv) iterative development and validation, and v) integration and maintenance. SULO development was initiated at a workshop with six participants with three having strong ontology engineering expertise, three with domain expertise and specific use cases, two being ETL developers, and one being a high-level domain expert. The in-person workshop was followed by 3 additional online workshops for further iterative development and validation. Owing to anonymization requirements in this submission, the ontology will be made available with a persistent identifier and hosted in a public repository, along with automatically generated documentation.

4. Results

4.1. SULO classes

SULO first comprises seven taxonomic levels with 19 classes, which are mutually disjoint at each level (see Figs. 1 and 2).

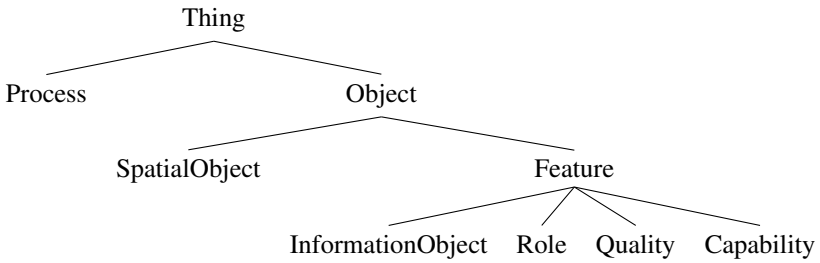

Figure 1. Taxonomy of SULO classes.

A *Process* has temporal parts, unfolds in time, has a duration and has objects as participants (*e.g. the life of an organism, a heart transplant, the administration of medication*). Processes that have temporal parts can only be fully determined if the last part of the process is finished. Consider a successful heart transplant process, which includes a sequence of temporal parts: opening the chest, establishing extracorporeal circulation, the extraction of the heart, the implantation of the heart, the restitution of circulation, and finally, the closing of the chest. Let p be a *Process*, only after the instantiation of the

last temporal part of a *Heart Transplant*, *p* can be asserted as being an instance of *Heart Transplant*.

An *Object* maintains its identity through time, and does not have processes as its parts. Objects are dichotomized into *SpatialObject* (e.g. a person, a particle or hospital) and *Feature*. The class *Feature* includes *Quality* (e.g. the redness of an apple), *Role* (e.g. the role of a patient and the role of a care provider), *Capability* (e.g. the capability to breathe), as well as *InformationObject* (e.g. a document, a clinical practice guideline, a mathematical formula).

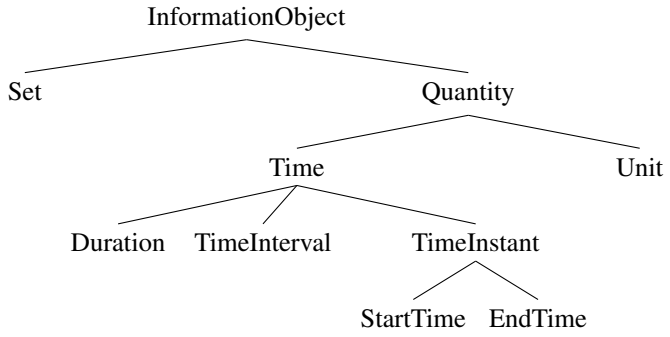

Figure 2. Taxonomy under SULO Information Object.

An *InformationObject* is a *Feature* that generically or specifically depends on and is about (i.e. represents) something. Two subclasses of *InformationObject* are introduced, *viz. Set* and *Quantity*. A *Set* is an *InformationObject* pertaining to classes, collections and mathematical sets, while a *Quantity* is an *InformationObject* that captures a numerical aspect (of an attribute) with a value and optional *Unit*, and is further divided into two subclasses *Time* and *Unit*. *Time* is a *Quantity* and is divided into 3 subtypes: a *TimeInstant*, *TimeInterval*, and *Duration* of time. *StartTime* and *EndTime* are two types of *TimeInstant* that are useful in demarcating the boundaries of a *Process* or a particular *TimeInterval*.

4.2. SULO relations

SULO proposes 21 object properties (including 10 inverses) and one datatype property.

4.2.1. temporal relations

The `atTime` relation holds between any entity and some measurement of time. It captures the notion of `when` in time something exists or occurs. Thus, SULO does not explicitly admit or refer to spatial or temporal regions, rather it focuses on associating entities with time measurements. The `precedes` relation holds between two distinct processes, $p1$ and $p2$, indicating that $p1$ ends before $p2$ begins. It captures strict temporal ordering and sequential relationships between processes.

4.2.2. Location

The `isLocatedIn` relation holds between any thing and a *SpatialObject*. It intends to capture *where SpatialObject* and *Process* can be located in space.

4.2.3. Parthood

The `isPartOf` relation is a *transitive*, *reflexive* relation expressing that entity *x* is included, contained, or incorporated within entity *y*, either physically, conceptually, or informationally. Entity *x* contributes to the composition or structure of *y*. The relation does not have a reference to time, and as such indicates the notion that *x* `isPartOf` *y* during some *Time T*4. Examples include *wheel* `isPartOf` *car*, *administration of ibuprofen* `isPartOf` *treatment of headache*, *premise* `isPartOf` *argument*. `isPartOf` is a subproperty of `isLocatedIn`, therefore any *x* `isLocatedIn` *y*.

The `isDirectPartOf` relation is a non-transitive mereological subrelation of `isPartOf` to specify cardinality constraints in a OWL class expression. This relation is needed because OWL 2 does not allow cardinality constraints over transitive relations. `hasDirectPart` is the inverse relation of `isDirectPartOf`. An example class expression is *Car* `subClasssOf` *SpatialObject that* `hasDirectPart` *exactly 4 Wheel*.

4.2.4. Descriptors

The `hasFeature` relation holds between any entity and a *Feature*. For instance, a three-dimensional *SpatialObject* is associated with the *Quality* of *Volume*, which may be measured and reported as a *Quantity* `atTime` *Time*.

4.2.5. Process participation

The `hasParticipant` relation holds between a *Process* and an *Object*. We capture the notion that when a **Feature** participates in a *Process*, then so does the **Object** for which it is a *Feature* of through the role chain `hasParticipant` o `isFeatureOf` $\rightarrow$ `hasParticipant`. A key design pattern is to represent the different roles that objects play in a particular process. For instance, the *Process* of *Administration of Medication* might involve one individual playing a *Care Provider Role*, which administers the medication to another individual playing a *Patient Role*.

4.2.6. Relations for information objects

An *InformationObject* always `refersTo` some other object it mentions, describes, represents or otherwise is information about. The `hasValue` relation is a functional data property to capture a single literal associated with an *InformationObject*. No other data property is admitted. The `hasItem` relation holds between a *Collection* and the items that constitute this collection, whether of type *Process* or *Object*. Finally, the physical representation of an *InformationObject* to a *SpatialObject* can be indicated with the `isFeatureOf` relation.

4.3. SULO Design Patterns

Two key ontology design patterns (ODPs) guide the implementation of domain ontologies and knowledge graphs using SULO.

4.3.1. SOLID (Single Object Literal Information Datum) Ontology Design Pattern

The SOLID pattern uses a single functional datatype property, `hasValue`, to assign a literal value to an instance of an *InformationObject*. This pattern precludes

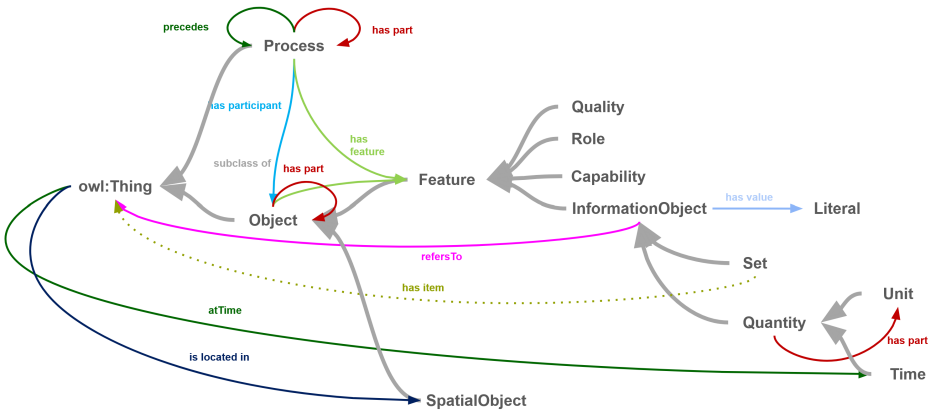

Figure 3. Overview of SULO classes and relations.

the introduction of domain-specific datatype properties such as `hasTemperature` or `hasAdmissionDateTime` by reifying these data relations as classes (e.g., *Temperature*, *DateTime*) under *InformationObject*. Thus, SOLID pushes the semantics out of data properties and into the classes, where they can be better described through class expressions. Moreover, it simplifies representations, as it is always predictable where any stored value will be in the range of `hasValue` of some information object.

4.3.2. PRO (Process-Role-Object) Ontology Design Pattern

The Process-Role-Object (PRO) ontology design pattern (Figure 4) provides a structured way to represent how entities participate in processes through specific roles. PRO specifies the roles that are necessary and specific to each object and which are direct participants in the process. We recover the participation of the object in the process by inference through the role chain described in Figure 5.

Process `hasParticipant` *Role* `isFeatureOf` *Object*

Figure 4. PRO Ontology Design Pattern

$$\texttt{hasParticipant} \circ \texttt{isFeatureOf} \rightarrow \texttt{hasParticipant}$$

Figure 5. PRO role chain

Listing 1 illustrates a scenario in which a physician (played by Dr. Smith) sees a patient (Alice) to discuss her condition. To capture the distinct roles of each individual, the visit (a *Process*) has the two roles (*alice_patient_role*, *smith_physician_role*) instantiating *PatientRole* and *PhysicianRole*, respectively, as participants. Reasoning over this graph using the role chain in Figure 5, we obtain two additional triples, *viz.* visit12345 has *alice* and *drsmith* as participants.

```
:visit12345 a :PatientVisit ;
    sulo:hasParticipant :alice_patient_role, :smith_physician_role .
:alice_patient_role a :PatientRole,
    sulo:isFeatureOf :alice .
:smith_physician_role a :PhysicianRole .
        :isFeatureOf :drsmith .

### inference from PRO role chain
:visit12345
    sulo:hasParticipant :alice, :drsmith .
```

Listing 1: Example instantiation of PRO role chain using RDF/Turtle syntax

Thus, the PRO pattern combined with the PRO role chain allows a clear mechanism by which dynamic, context-dependent roles of objects can be described in relevant processes, which retain their association to facilitate downstream query answering or data validation.

5. Discussion

5.1. SULO-based knowledge representation

5.1.1. SULO with SNOMED CT

The use of upper-level ontologies with domain terminologies like SNOMED CT is increasingly recognized as essential for achieving semantic interoperability in healthcare and biomedical research. In [11], an alignment of SNOMED CT with the upper-level ontology BioTopLite2 (BTL2) [12] was proposed. Based on that, SNOMED CT upper-level classes and relations have been aligned with SULO. Tables 1 and 2 show the mapping of the main SNOMED CT classes and relations. It was done manually, by analysing the meaning of the candidate classes and relations, considering the formal axioms as well as text definitions and hierarchical context.

Table 3 illustrate the representation of the SNOMED CT concepts Fracture of femur (disorder) and Chronic Kidney Disease Stage 1 (disorder) using SULO.

5.1.2. SULO with SPHN

The Swiss Personalized Health Network (SPHN) initiative developed an RDF schema to share health-related data and ensure its interoperability [7]. Table 4 presents candidate mapping for selected SPHN (v2025.1) classes to SULO. However, these mappings are underdetermined as SPHN classes often contain a mixture of relations for different SULO types. For example, *Drug* contains properties represented in both *SpatialObject* and *InformationObject* concepts. While the property `hasActiveIngredient` *Substance* implies that this class should belong to *SpatialObject*, the property `hasSourceSystem` *SourceSystem* indicates that this class should a kind of an `InformationObject`. SULO can help identify such schema design problems and improve the schema by separating combined classes more effectively.