Artificial Intelligence Project Documentation

Matteo Conti and Francesco Iannaccone

8 June 2021

Contents

1 Introduction **3**

2 Ontology Engineering **3**

3 Domain of interest **4**

4 Ontology development **4**

4.1 Classes 4

4.2 Properties 7

4.2.1 Object properties 8

4.2.2 Data properties 9

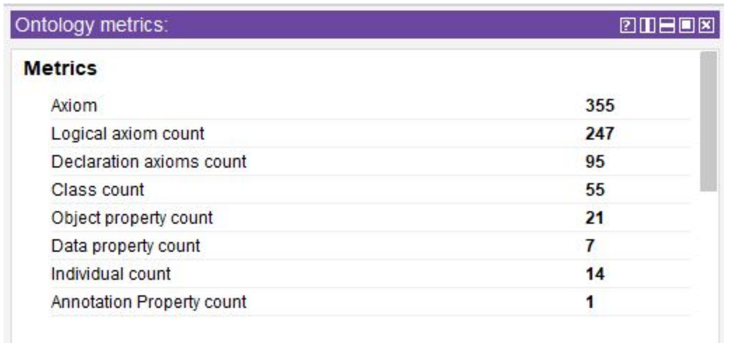
4.2.3 Annotation properties 11

4.3 Individuals 11

1 Introduction

The project, carried out on the occasion of the Artificial Intelligence exam of the three-year Computer Engineering course of the University Federico II of Naples, is based on the creation of an ontology on Protégé starting from a scientific article.

Considering an approach typically called top-down, we will initially analyze the backbone of the ontology: the entities of higher hierarchical level and the main object properties to which they are related. Then, we will proceed to evaluate the various sub-classes, their object and data properties and finally the instances we have decided to implement.


Ontology metrics:

Metrics

Axiom	355
Logical axiom count	247
Declaration axioms count	95
Class count	55
Object property count	21
Data property count	7
Individual count	14
Annotation Property count	1

2 Ontology Engineering

Nowadays, in order to make information processable and thus machine understandable, we need ontologies. An ontology is a formal, shared and unambiguous representation of information in a given domain.

The languages used for creating, composing and editing an ontology are OWL (Ontology Web Language) and RDF (Resource description Framework). They describe ontologies and have practically the same syntax.

For our project, we used Protégé, a tool that allowed us to represent the information of our domain graphically and, above all, allowed us to subsequently export the ontology in **.owl** and **.tex** format.

Ontology engineering means defining terms within a domain and the relationships between them. In particular, it is necessary to define the concepts in the domain (classes), distinguishing them from the attributes, which have no properties or structure; to organise the concepts in a hierarchy; to define what attributes and properties the classes can have and the constraints on their values; and finally, to define the individuals (instances) and fill in the values of the properties.

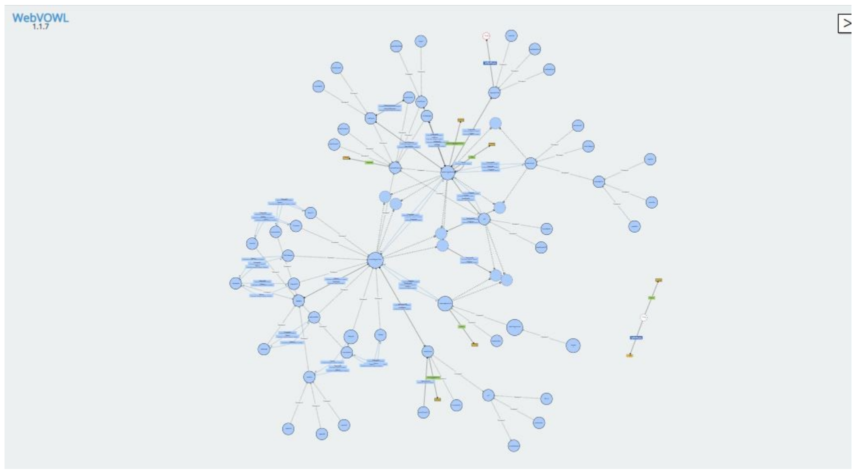


Figure 1: Ontology represented using WebVOWL 1.17 with degree of collapsing equal to 0

3 Domain of interest

The article [1] on which our ontology is based describes a standard used in CDS (Clinical Decision Support) systems since the 1990s, **Arden Syntax**. The standard is at the forefront of representing clinical and scientific knowledge in an executable format.

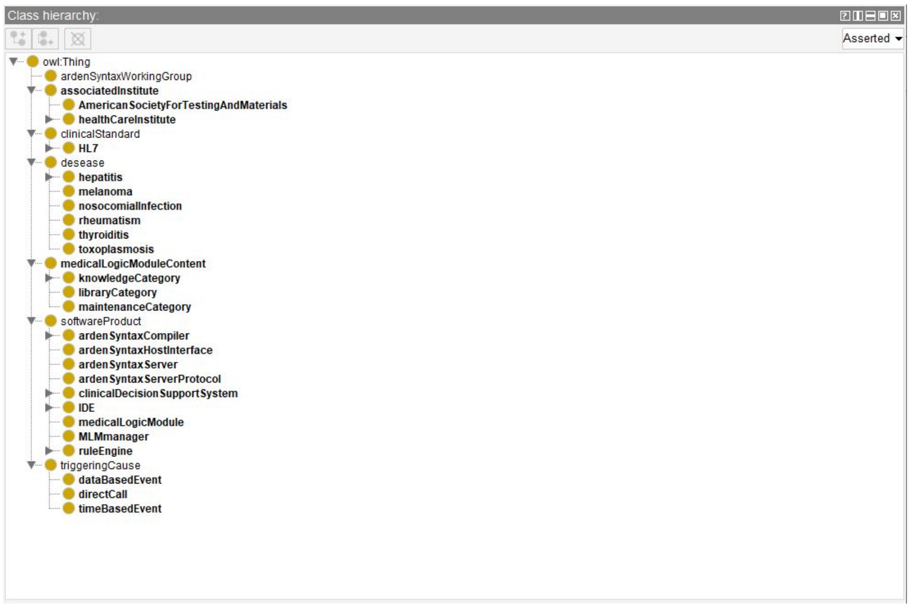
In the article, it is explained how Arden Syntax organises the code into independent files called **Medical Logic Modules** (MLMs). The execution of an MLM can be triggered by specific time-based, data-based or direct call events. The intent of the article is clearly to give a general overview of what Arden Syntax is about, the software products related to it, its applications in the medical field and future implementations that aim to improve the efficiency of CDS systems.

Finally, it must be said that the choices we have made throughout the design of the ontology come from a deep analysis of the article, carried out by identifying the terms of interest in the text.

4 Ontology development

4.1 Classes

A class is a concept belonging to the domain and can be defined as a collection of elements with similar properties. The main classes of our ontology, i.e. classes that are direct daughters of the *owl:Thing* class, are *ardenSyntaxWorkingGroup*, *associatedInstitute*, *clinicalStandard*, *desease*, *medicalLogicModuleContent*, *software Product*, *triggeringCause*.

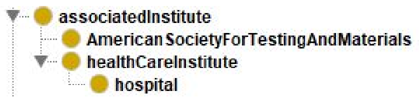


```
Class hierarchy:                                                     Asserted
owl:Thing
    ardenSyntaxWorkingGroup
    associatedInstitute
        AmericanSocietyForTestingAndMaterials
        healthCareInstitute
    clinicalStandard
        HL7
    desease
        hepatitis
        melanoma
        nosocomialInfection
        rheumatism
        thyroiditis
        toxoplasmosis
    medicalLogicModuleContent
        knowledgeCategory
        libraryCategory
        maintenanceCategory
    softwareProduct
        ardenSyntaxCompiler
        ardenSyntaxHostInterface
        ardenSyntaxServer
        ardenSyntaxServerProtocol
        clinicalDecisionSupportSystem
        IDE
        medicalLogicModule
        MLMmanager
        ruleEngine
    triggeringCause
        dataBasedEvent
        directCall
        timeBasedEvent
```

Figure 2: General overview of the ontology

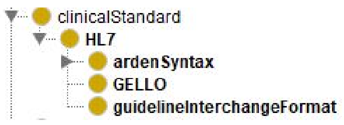
ardenSyntaxWorkingGroup represents the working group that makes decisions on Arden Syntax.

associatedInstitute are all the companies/organisations that have used Arden Syntax, mentioned in the article. The first institute is the American Society For Testing and Materials, to which we have added healthcare institutes, which can also be hospitals. Of this last class we will then go on to analyse several individuals we have created, mentioned in the text as examples.



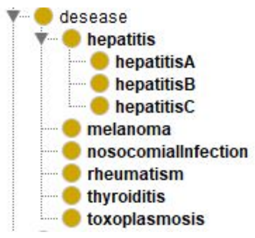
```
associatedInstitute
    AmericanSocietyForTestingAndMaterials
    healthCareInstitute
        hospital
```

clinicalStandard is the class of clinical standards mentioned in the article. They are all Health Level 7 (HL7) and are: GELLO, GuideLineInterchangeFormat and ardenSyntax, of which the fuzzy logic version is also specified.

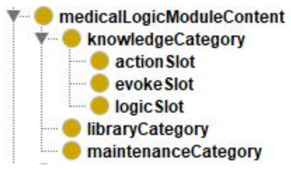


```
clinicalStandard
    HL7
        ardenSyntax
        GELLO
        guidelineInterchangeFormat
```

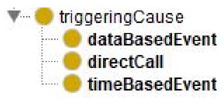
The class desease was created to represent the diseases that are detected by the CDS systems presented as Arden Syntax use cases in the article.



medicalLogicModuleContent indicates the content type of an MLM. It can be of 3 different categories: knowledge category, library category and maintenance category. The knowledge category is also divided into several slots to improve transparency. Three types of slots are mentioned in the text, namely action slots, evoke slots and logic slots.



triggeringCause is the class representing the causes for an MLM to be triggered. There are three types: data-based event, time-based event and direct call.



softwareProduct on the other hand is the largest class in our ontology, as it contains specific information about software implementations of Arden Syntax. A software product can be: ardenSyntaxCompiler, ardenSyntaxHostInterface, ardenSyntaxServer, ardenSyntaxServerProtocol, clinicalDecisionSupportSystem, IDE, medicalLogicModule, MLMmanager and ruleEngine.

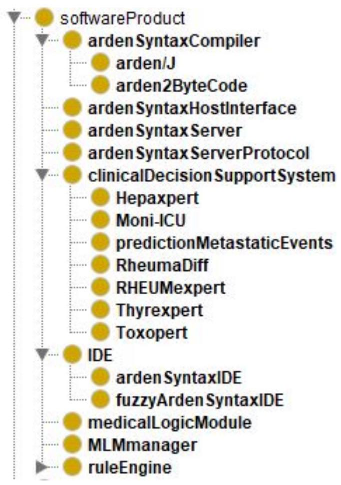


Figure 3: softwareProduct class and his sub-class

In particular, ardenSyntaxCompiler has two sub-classes: the compiler types that are indicated in the text are arden/J and arden2ByteCode.

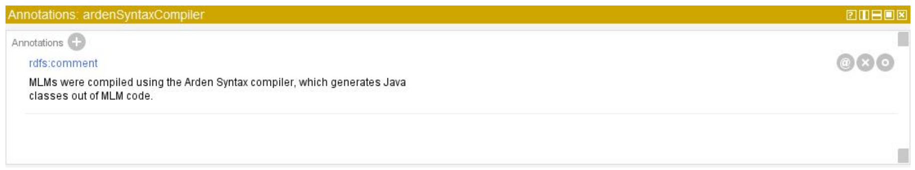
ruleEngine can be specialised into ardenSyntaxRuleEngine and its extended version fuzzyArdenSyntaxRuleEngine. Similar reasoning has been applied to the IDE class, as can be seen in figure 3.

For the clinicalDecisionSupportSystem class, it was decided to specialise it in the various CDS which are mentioned in the article, in section 3.1. We therefore find Hepaxpert, Moni-ICU, RheumaDiff, RHEUMexpert, Thyrexpert, Toxopert and prediction of metastatic events.

4.2 Properties

Properties describe the attributes of instances of the class and the relationships with other instances. Properties can be simple (data property) in the sense that they contain primitive values such as a string or a number, or complex (object property), i.e. having an instance of another class as a range.

Another type of property we have considered too are Annotation properties. They are used to provide explanations for certain choices or even simply to add comments on certain entities.


Annotations: ardenSyntaxCompiler

Annotations

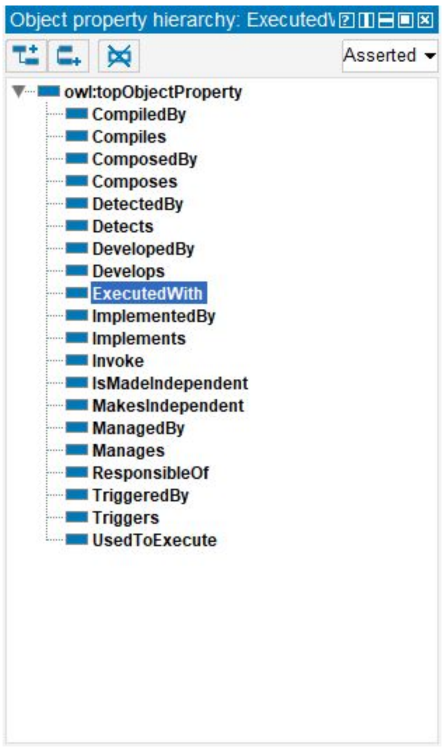
rdfs:comment

MLMs were compiled using the Arden Syntax compiler, which generates Java classes out of MLM code.

Figure 4: Example of annotation property

4.2.1 Object properties

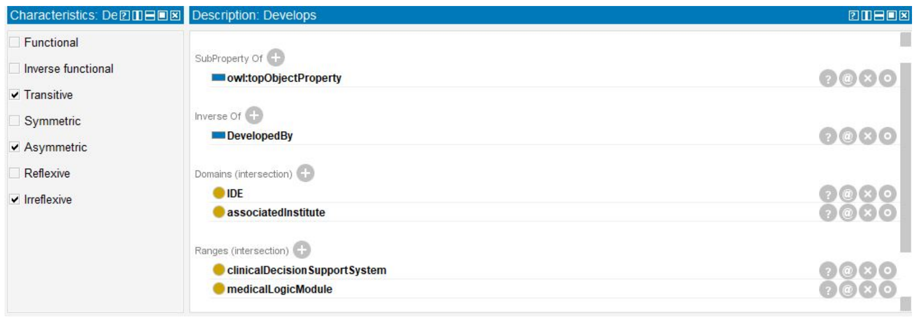
As can be seen from the figure 5, we have added object properties of various types, from composition properties (Composes) to more intrinsic ones, such as TriggeredBy or Develops.



owl:topObjectProperty
- CompiledBy
- Compiles
- ComposedBy
- Composes
- DetectedBy
- Detects
- DevelopedBy
- Develops
- ExecutedWith
- ImplementedBy
- Implements
- Invoke
- IsMadeIndependent
- MakesIndependent
- ManagedBy
- Manages
- ResponsibleOf
- TriggeredBy
- Triggers
- UsedToExecute

Figure 5: Object properties added in our ontology

Develops is an object property that links both associatedInstitute with clinicalDecisionSupportSystem and IDE with medicalLogicModule. The choice was to merge the two relationships into a more general one, as the concept of ”develop” is the same for both MLMs and CDS systems. The characteristics of the property are transitivity, asymmetry and irreflexivity. The inverse of this relationship is DevelopedBy.


Characteristics: De

- [ ] Functional
- [ ] Inverse functional
- [✓] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [✓] Irreflexive

Description: Develops

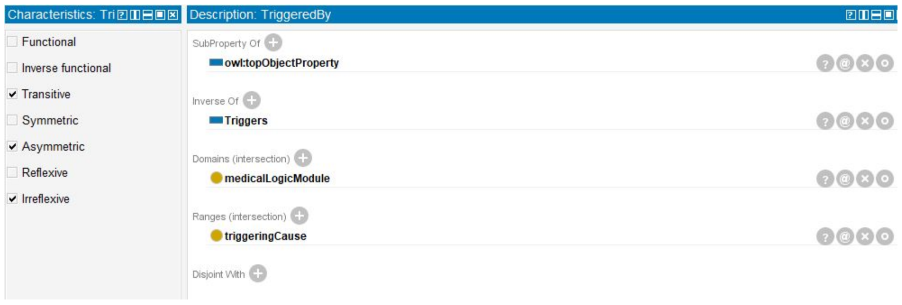
SubProperty Of
- owl:topObjectProperty

Inverse Of
- DevelopedBy

Domains (intersection)
- IDE
- associatedInstitute

Ranges (intersection)
- clinicalDecisionSupportSystem
- medicalLogicModule

TriggeredBy is the object property with the domain medicalLogicModule and the range triggeringCause. The explanation is trivially clear from the names, as MLMs are triggered by very specific causes. Clearly, it is a transitive, asymmetric and unreflective relationship and its inverse is Triggers.


Characteristics: Tri

- [ ] Functional
- [ ] Inverse functional
- [✓] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [✓] Irreflexive

Description: TriggeredBy

SubProperty Of
- owl:topObjectProperty

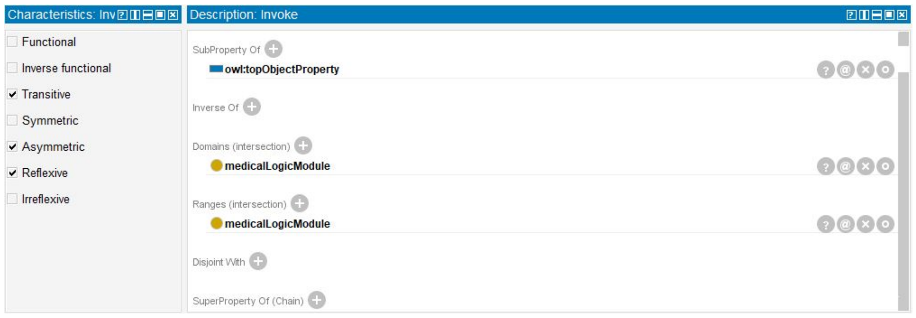
Inverse Of
- Triggers

Domains (intersection)
- medicalLogicModule

Ranges (intersection)
- triggeringCause

Disjoint With

Invoke, finally, has both as domain and range medicalLogicModule, which makes us realise that it is a reflexive relation, since one MLM can invoke another MLM. Furthermore, the relationship is transitive and asymmetric.


Characteristics: Inv

- [ ] Functional
- [ ] Inverse functional
- [✓] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [✓] Reflexive
- [ ] Irreflexive

Description: Invoke

SubProperty Of
- owl:topObjectProperty

Inverse Of

Domains (intersection)
- medicalLogicModule

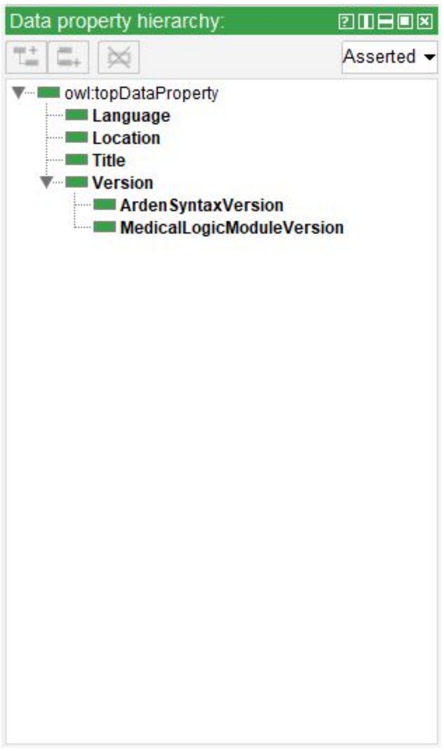
Ranges (intersection)
- medicalLogicModule

Disjoint With

SuperProperty Of (Chain)

4.2.2 Data properties

The data properties we have added to the ontology are: Language, Location, Title and Version, which can be specialised into two sub-properties, ArdenSyntaxVersion and MedicalLogicModuleVersion.



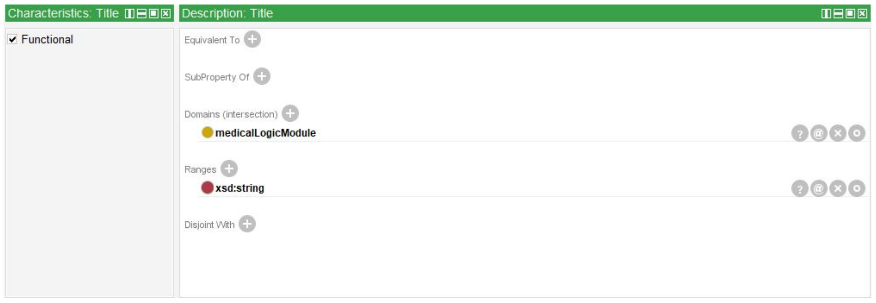
Data property hierarchy:

Asserted

- owl:topDataProperty
  - Language
  - Location
  - Title
  - Version
    - ArdenSyntaxVersion
    - MedicalLogicModuleVersion

Figure 6: Data properties added in our ontology

Title is a data property of medicalLogicModule as each MLM has a title which summarises its functionality. The range of the relation is xsd:string and can be classified as a functional property because each MLM corresponds to one and only one title.



Characteristics: Title

✓ Functional

Description: Title

Equivalent To

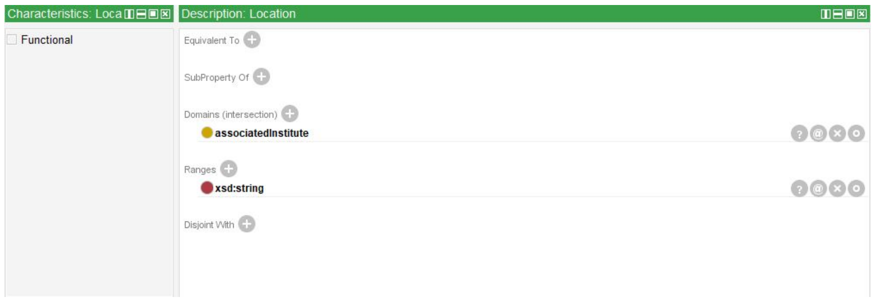
SubProperty Of

Domains (intersection)
medicalLogicModule

Ranges
xsd:string

Disjoint With

Location is the data property whose domain is the associatedInstitute class and the range xsd:string. It indicates the geographic location of a given institute/organisation.


Characteristics: Loca

Functional

Description: Location

Equivalent To

SubProperty Of

Domains (intersection)

associatedInstitute

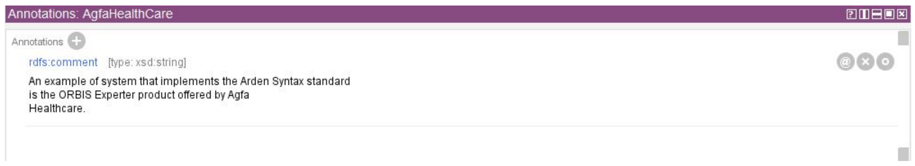
Ranges

xsd:string

Disjoint With

4.2.3 Annotation properties

We have added several annotation properties, of type rdfs:comment, to classes, object properties and data properties and also to individuals. In figure 7, we can see an example of an annotation applied to the individual AgfaHealthCare.


Annotations: AgfaHealthCare

Annotations

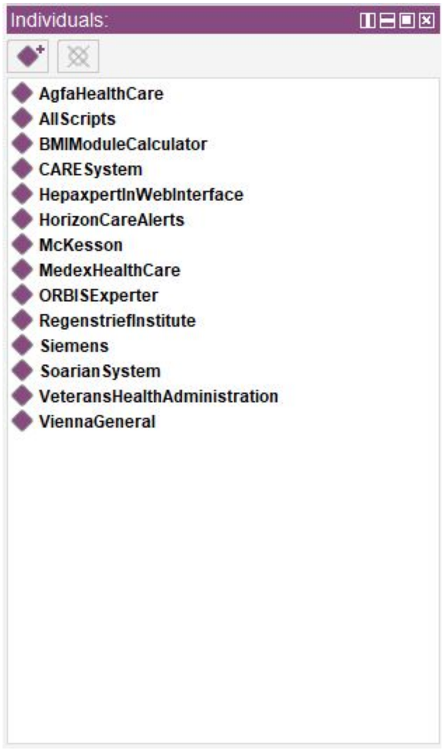
rdfs:comment [type: xsd:string]

An example of system that implements the Arden Syntax standard
is the ORBIS Experter product offered by Agfa
Healthcare.

Figure 7: Example of annotation property added in the ontology

4.3 Individuals

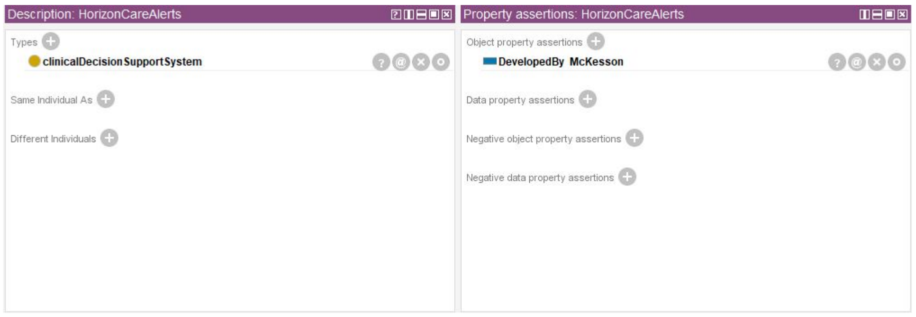
In conclusion, we have created some instances (individuals) of the classes in our domain of interest. In this way, the class becomes a direct type of the instance, and in addition, we can assign property values to the instances, in a manner that conforms to the constraints imposed when defining the property itself.


Individuals:

AgfaHealthCare
AllScripts
BMIModuleCalculator
CARESystem
HepaxpertInWebInterface
HorizonCareAlerts
McKesson
MedexHealthCare
ORBISExperter
RegenstriefInstitute
Siemens
SoarianSystem
VeteransHealthAdministration
ViennaGeneral

Figure 8: All the individuals created in our ontology

An example of an instance is HorizonCareAlerts, a CDS system mentioned in the text, which has the object property DevelopedBy, whose range is an instance of associatedInstitute (McKesson).


Description: HorizonCareAlerts

Types
clinicalDecisionSupportSystem

Same Individual As

Different Individuals

Property assertions: HorizonCareAlerts

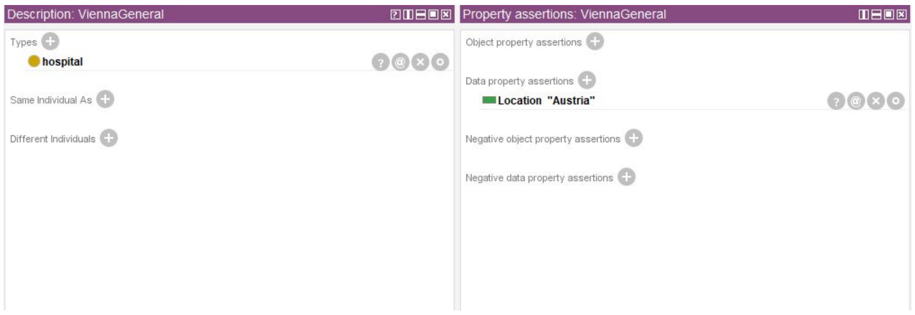
Object property assertions
DevelopedBy McKesson

Data property assertions

Negative object property assertions

Negative data property assertions

ViennaGeneral is an instance of hospital and has as data property Location, whose corresponding value is a string, in this case ”Austria”. Note that in this case the data property has been inherited from the associatedInstitute superclass.


Description: ViennaGeneral
Types
hospital
Same Individual As
Different Individuals
Property assertions: ViennaGeneral
Object property assertions
Data property assertions
Location "Austria"
Negative object property assertions
Negative data property assertions

References

[1] Matthias Samwald, Karsten Fehre, Jeroen De Bruin, and Klaus-Peter Adlassnig. The arden syntax standard for clinical decision support: Experiences and directions. *Journal of biomedical informatics*, 45(4):711–718, 2012.