Self-modifying ontologies: Displaying the Domain Variability

Olegs Verhodubs
oleg.verhodub@inbox.lv

Abstract. Ontology is a proven way to describe some domain. A toolkit for describing ontologies has been developed, and this toolkit is the OWL (Web Ontology Language) language. Despite the abundance of constructions in OWL, there are tasks that are difficult to describe using the OWL language. This complexity of description is associated with the need to depict transformations that occur with objects in the domain constantly or from time to time, such as radioactivity, which transforms one element into another or others. To some extent, here the statics of the expressive means of the OWL language comes into conflict with the dynamics of the domain. This paper addresses the problem of the difficulty of displaying the natural dynamics of some subject areas, and suggests ways to overcome this problem. It is proposed to supplement the OWL language with some constructions, which will transform the ontology from static to dynamic and even to self-modifying.

Keywords: Ontology, Self-Modifying Ontology, Transformations in Ontology, Dynamic Ontology, OWL development

I. Introduction

A thought, once spoken, is a lie [1]. This quote is often used by neuroscientists (for example, Alexander Kaplan) when they want to emphasize the difference between the verbal and intracerebral representation of some object. That is, according to expert opinion, the essence of an object and its verbal description are not the same thing. The essence of things is of keen scientific interest, but not everything is clear about it either. Being somewhat familiar with the worldview of yogis, Buddhist monks and all those who practice deep meditation professionally and for decades, one can assume that their understanding of the essence of things differs from that of ordinary people. In the absence of metrics of essence, because there is still no standard for a unit of essence, one has to be content with a rough description of objects from any domain through the distinctive features (properties) of objects and their connections with other objects. These ideas of ours about the world were transferred to the computer, information world, or rather to the artificial information world, where ontologies became a means of describing different domains, the sum of which makes up the world as we understand it.

The concept of ontologies, as part of the Semantic Web, replaced the concept of text markup, the purpose of which was to present information beautifully and attractively. With the ever-increasing volume of information on the Web, it became clear that the old concept of information design no longer meets the needs of users, which is why ontologies began to be used, which are created in a set of constructions of the OWL (Web Ontology Language). The use of ontologies potentially makes information on the Web machine-readable, but the very representation of domains faces some limitations. It is not that there are areas of knowledge that cannot be displayed using ontologies in general, but sometimes it is simply quite difficult to do this with the current set of ontology language tools (hereinafter the words ontology, ontology language, OWL, Web Ontology Language are used as synonyms). Difficulties may arise when describing the variability of objects, such as, for example, the decay of some radioactive elements of the periodic table. It is not so bad if the decay period of radioactive elements lasts many hundreds of years. Indeed, in this case, this phenomenon can be neglected, taking radioactive elements as stable and not decaying. What, if the decay period lasts a fraction of a

second? How can this physical phenomenon be displayed using existing means of the OWL language? The decay of radioactive elements is not the only problem for mapping through ontology. For example, there is a whole class of chemical reactions that occur in an oscillatory mode (for example, the Belousov-Zhabotinsky reaction [2]). In this case, a non-trivial approach is also needed if you use ontologies for mapping. Often, much more everyday things in the world around us need a mechanism for describing the variability of a domain or part of it. So, we need a mechanism for describing spontaneous (and not only) variability of the domain in ontologies and it is discussed in the following sections.

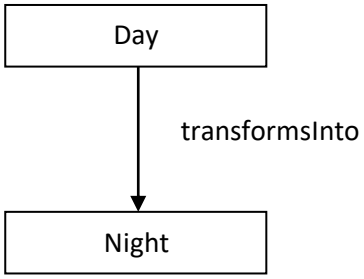
This paper is organized into several sections. The next section describes the ways to define transformations in ontologies and the third section sketches a future dynamic Semantic Web. Finally, the conclusion summarizes the results of this research.

II. Description of Variability in Ontology

The variability of something attracts our attention, and therefore our interest, much more strongly than constancy. This is logical, because historically, in conditions of dangers in the surrounding world, it is much more effective to pay attention to changing objects and phenomena than to what is constant. That is why a human instinctively neglects what does not change over time or changes only slightly. However, it is precisely the being manifested stable constancy in the surrounding reality that causes genuine surprise. For example, the narrow boundaries of the temperature regime of the human body [3] or the Earth seem unimaginable, taking into account the wide range of possible temperatures, from the temperature of outer space to the temperature of the Sun. Although constancy and stability are the exception rather than the norm, this is taken as the basis for modeling the surrounding reality. In this case, if any transformation needs to be displayed, then it is encoded as a transition between states in the state space [4]. But in general, it would be more logical to rely on the variability of the surrounding world and the objects included in it, taking into account the immeasurably greater amount of changeable than constant and stable in the surrounding world.

The use of ontologies is at the heart of all Semantic Web applications [5]. The Semantic Web is the name of a new concept of the World Wide Web that aims to shift from the presentation of information in the Web to its semantics. Ontologies are needed to formally describe, or in other words, specify any part of the surrounding world , that is, the domain. Despite the huge number of various sciences and engineering branches, all sciences and engineering branches are a kind of communicating vessels, where the same principles are common to seemingly different scientific fields. The use of permanent, stable objects as basic units for ontology implementation is a general principle that brings ontology development closer to other types of design. However, even the most stable objects or phenomena are not like that in the strict sense of the word. That is why it is necessary to be able to describe variability using the language of ontologies. Among the languages for describing ontologies, a leader has long stood out, which is the Web Ontology Language, or OWL for short. Unfortunately, this language does not have explicit means for specifying variability in a domain. Therefore, it is necessary to adapt the available OWL tools to the variability of the domain. For example, if there was a need to display the classes “Day” and “Night”, as well as the fact that day transitioned or turned into night, then we could do it like this:

Visual representation



OWL code

```
<owl:Class rdf:ID="Day">
</owl:Class>
<owl:Class rdf:ID="Night">
</owl:Class>
<owl:ObjectProperty rdf:ID="transformsInto">
   <rdfs:domain rdf:resource="#Day"/>
   <rdfs:range rdf:resource="#Night"/>
</owl:ObjectProperty>
```

Fig.1. Displaying the transformation of one class into another.

The transformation of day into night is not the only example of natural variability that requires means of display. However, even this simple example hides pitfalls such as cyclicity. That is, the transformation of day into night and back occurs constantly, and this fact should be reflected in the ontology so that the model would be more complete. Unfortunately, there are no built-in means to display the cyclical nature of something, so something needs to be invented.

Let us look at another example. In nuclear physics, such a phenomenon as radioactivity is known. Radioactivity is the physical phenomenon by which the nuclei of unstable atoms (called radionuclides or radioisotopes) spontaneously transform themselves into other atoms by simultaneously emitting radiation, that is to say particles of matter [7]. The key point in this definition is that some substances are converted into others. Apparently, it would not cause noticeable difficulties if the period of transformation (decay period) of some substances into others took many thousands of years. This actually happens with some substances, like Pu-239 (half-life is 24100 years) [8]. But there are quite a lot of radioactive substances whose lifetime lasts only hundredths of a second. In this case, of course, it is no longer possible to neglect the instability of the substance, because the transformation of the substance (change in its quality) occurs extremely quickly. An example of this kind of radioactive decay reaction, occurring in a fraction of a second, is the decay of moscovium [9]:

$$^{286}\mathrm{Mc} \rightarrow {}^{282}\mathrm{Nh} + \alpha \qquad (1)$$

The radioactive decay reaction of moscovium occurs in 20 milliseconds, while other similar radioactive decay reactions can occur in other times. Therefore, for a full-fledged model, it is very important to specify the reaction time. The lack of an explicit opportunity to indicate the time of occurrence of a radioactive reaction, and, more generally, the lifetime of ObjectProperty, is the second significant obstacle to make the model specified by means of the OWL adequate. However, in some cases it is vital to set the duration of some action, so workarounds should be sought for this.

The radioactive decay reaction of moscovium demonstrates the unidirectional and single-cycle transformation. This means that moscovium turns into nihonium only in this direction and no further transformation occurs, however this is not always the case. There are many radioactive decay reactions, where transformations occur in a chain manner. And even this does not qualitatively complicate the overall picture as much as cyclic processes, in which, with the transition from one state to another state, the reaction does not die out, but continues further, again transitioning to the original state, etc. Such reactions, in particular, are Belousov–Zhabotinsky reactions that occur in an oscillatory mode [2]. Displaying such reactions in the ontology is problematic if you use exclusively standard OWL language expressions. Apparently, viewing transformations as processes can help with mapping these transformations into ontology. Mapping processes in ontology is a fairly well-developed topic. This topic is fully or partially devoted to such works as [10], [11], [12] and others. It seems that displaying both processes and objects without distinguishing them is a logical dissonance. This logical dissonance would not exist if everything displayed in the ontology were only objects, or, on the contrary, everything displayed in the ontology were only processes. In the case of mixed content in the ontology (both objects and processes), in order to avoid logical dissonance, one should create one common super-class “Process”, which will contain the main characteristics of any process. Schematically, the "Process" class might look like this (Fig.2):

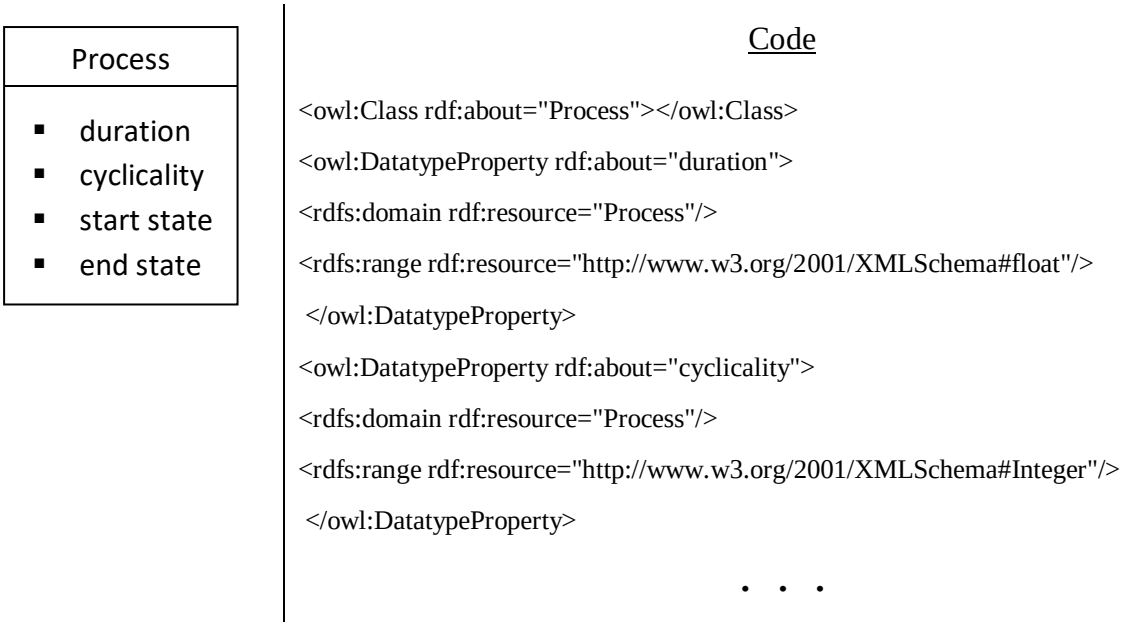


Fig.2. The “Process” class.

The “start state” and “end state” are properties (dataproperty in OWL) that contain the names of the classes between which the transformation occurs. In the case of a multi-stage transformation, instead of the initial and final states, it is better to use a collection of names of the classes participating in the transformation. The downside to using a separate Process class is that it has to be re-created every time the process needs to be mapped to the ontology, so you should look to other options.

In general, it is possible to make do with a limited number of means of expression and solving any problem with limited resources is akin to art. However, more capabilities usually provide more flexibility. More capabilities can only be provided by adding new, specialized means of

expression for displaying transformations in ontologies. This direction in the development of ontologies transforms them from static to dynamic, and it is this issue that will be discussed in the next section.

III. From Static to Dynamic Ontology

A well-known statement says that a plan cannot be either not partially fulfilled or exceeded, because a plan is a plan. However, real life always makes its own adjustments to any endeavor conceived by a person, be it a plan or something else. That is why precise calculation and scrupulous adherence to precise calculation carries with it the grain of failure and the value of the tools involved is more useful to evaluate from the point of view of the variety of possible alternatives and workarounds. OWL's capabilities are not limited to displaying only objects and their connections, because you can use comments to store anything. Comments are a kind of reserve, or an unstructured field for describing new qualities not provided for by the standard OWL syntax. Actually, using comments for purposes other than their intended purpose is nothing new. For example, comments in ontologies have already been used to store the values of membership functions [13]. Certainly, there are other examples of inappropriate use of comments in ontologies, but we will not consider them all.

In contrast to the structured nature of the main ontology elements, ontology comments are not structured. However, they also need to be structured to become useful. Unlike the official structuring of the main elements of the ontology, the structure described in the comments of the ontology is subject to agreement. This agreement may vary from application to application, i.e. each application or handler program can have its own type of perceived comment structure. Thus, comments of the ontology can act as an intermediate level between information about the ontology in free form and an established structure with a given standard number of constructs of the OWL language. A structure that has proven its worth may eventually migrate from comments to the standard set of OWL language.

Associative container is one of the data structure that can be used to store additional information about the ontology elements in the comments of the ontology. Associative container has different names in different programming languages. It is map in C++ [14], dictionary in Python [15] and so on. Regardless of the name, this data structure stores the value in key:value pairs. This is enough to store information about the transition of one object to another with some cyclicality. For example, for two classes “Day” and “Night” (Fig.1.) additional information about the cyclic transition from one object or state to another stored in ontology comments may look like this:

[‘Day’: ’Night’, ’Night’: ‘Day’, cyclicality: -1] (2)

The example (2) shows that ‘Day’ goes into ‘Night’ but ‘Night’ goes into ‘Day’ and these transitions happen endlessly. The positive value for the ‘cyclicality’ property means a finite number of transitions.

It is necessary to have one more property to represent some transition from one object to another with the formation of additional objects. In addition, it is necessary to display how long this reaction takes place. For example, in the case of moscovium (1) additional information in the comments of ontology may look like as follows:

['Moscovium': 'Nihonium', cyclicality: 1, objects: α, duration: 0.002] (3)

Here (3) moscovium goes into nihonium with cyclicality 1 and α objects (alpha particles) are emitted.

It is assumed that the string (3) is stored in the comment of the moscovium class in the OWL ontology and the string (2) is stored in the comment of the "Day" class in the OWL ontology. Each class can have its own comment that contain informaition transformation, cyclicality of transformation and newly formed particles. In general, transformation can occur in stages. In such a case it is necessary to specify all these stages in a strictly defined order. For example, if there are classes A, B, C and A goes into B and B goes into C then information in comments looks like as follows:

['A': 'B','B': 'C'] (4)

Certainly, the example (4) can be easily expanded to more intermediate stages. Such a way of displaying changes using comments in the ontology is suitable for solving immediate practical problems. However, from the point of view of the philosophy of domain mapping, this method is incorrect. The thing is that now ontology is considered as something fixed, not subject to change, while everything around is changing. Therefore, it would be more correct not to come up with ways to get out of the situation of lack of means to express variability, but to include the ability to display variability as an integral part of the OWL vocabulary. In this case, the ontology turns from a static to a dynamic model of a certain domain. The way to describe domain variability can be borrowed from CSS (Cascading Style Sheets) [16]. CSS provides programmers with a convenient mechanism for specifying animation [17]. For example, this code shows a graphic object (square) that changes its color four times and this 4-stage color change continues for 3 times:

```
@keyframes square {
  0%   {background-color: green;}
  25%  {background-color: blue;}
  50%  {background-color: red;}
  100% {background-color: black;}
}

div {
  width: 75px;
  height: 75px;
  background-color: green;
  animation-name: square;
  animation-duration: 5s;
  animation-iteration-count: 3;
}
```

The "@keyframes" rule in CSS may have a different appearance. For example, here the background color changes from green to red:

```
@keyframes example {
  from {background-color: green;}
  to {background-color: red;}
}
```

By analogy, the following syntax may be proposed for specifying changes for ontology elements. To demonstrate, here is an example:

```
<owl:Transform rdf:id="Day">
        <owl:to rdf:id=”Night”>
        <owl:iteration rdf:count=”-1”
  </owl:Transform>
```

```
<owl: Transform rdf:id="Night">
        <owl:to rdf:id=”Day”>
        <owl:iteration rdf:count=”-1”
</owl: Transform>
```

In the example shown above, the nature of changes is specified for ontology elements called “Day” and “Night”. In addition to specifying the direction of the transformation, the example indicates that the transformation will continue indefinitely (“-1” means infinitely). A more concise syntactic example would look like this:

```
<owl:Transform>
        <owl:from rdf:id=”Day”>
        <owl:to rdf:id=”Night”>
        <owl:iteration rdf:count=”-1”
  </owl:Transform>
```

If it is necessary to display the multi-stage changes, then it would be done something like this:

```
<owl:Transform>
        <owl:stage rdf:id=”A”>
        <owl:stage rdf:id=”B”>
        <owl:stage rdf:id=”C”>
        <owl:iteration rdf:count=”2”
  </owl:Transform>
```

In the example above A transforms to B, B transfotms to C and these multi-stage transformations continues 2 times. If something new is formed as a result of transformation, then a description of this should be included in the OWL code. For example, α particles are being formed in the result of Moscovium transformation to Nihonium and this reaction takes place in 20 milliseconds. So, this process of transformation may be descripted by means of this OWL code:

```
<owl:Transform>
      <owl:from rdf:id=”Moscovium”>
      <owl:to rdf:id=”Nihonium”>
      <owl:duration rdf:value=”0.002”>
      <owl:newclass rdf:id=”α”>
</owl:Transform>
```

Here two options are possible, and they differ from a philosophical point of view. The first one implies that the class “α” initially exists that is desribed in the ontology and the transformation description block (block that begins with “<owl:transform>”) simply reports that α particle phisically is created here (in the process of moscovium transformation to nihonium). Thus, this option means that all physical forms are known beforehand and their incarnation occurs as life progresses (as life develops). The second option implies that the class “α” does not exist initially (that is, it is not described in the ontology), but this “α” class will occur in the ontology

dinamically. Such an option means that the ontology may change itself. It is necessary to specify the starting point, when dealing with changes to control the situation. Speaking about ontologies, you can always determine what is earlier and what is later, because the description of the ontology itself provides such information: the description of some elements of the ontology comes earlier than the description of others. As a result, we get a transformable, dynamic ontology, which is more consistent with the model of a certain subject area than the usual, static ontology with which we were accustomed to working before.

In the examples above, we looked at changes that affect something (things, material objects, etc.), which is expressed in ontology as classes. However, it can be assumed that relations between classes themselves may undergo changes over time. For example, in the simplest case, two people, being friends at first, can turn into sworn enemies over time. The proposed method for displaying changes can cope with such a challenge as well. For this purpose, in the "<owl:Transform>" tag you need to indicate the initial and final names of the relations, instead of the class names (as in our last example, these are the class names "Moscovium" and "Nihonium"). Apparently, the considered properties for the "<owl:Transform>" tag, such as "from", "to", "stage", "iteration", "duration", "newclass" will also be valid in the case when we are talking not only about changes to classes, but also about changes that affect relations. To summarize, let's display a list of all new proposed tags for the ontology (table I):

TABLE I. The list of proposed tags for adding to OWL.

Nr p.k.	Tag name	Description
1	<owl:Transform> </owl:Transform>	main tag to display changes to classes or relations
2	<owl:from>	initial state of change
3	<owl:to>	final state of change
4	<owl:stage>	intermediate stage of change
5	<owl:iteration>	number of change cycles
6	<owl:duration>	duration of change
7	<owl:newclass>	creating a new object while changing

These tags (table I) are sufficient to show the basic characteristics of changes in classes and relationships, but it is possible that additional tags will be identified over time.

IV. Conclusion

Apparently variability is one of the most essential features of any viable process in particular and of Life in general. Despite their intangible nature, domains tend to change along with the real objects they represent. The difficulty is that ontologies in their current form do not have built-in tools for displaying the variability of the domain. This paper shows the way how to overcome the difficulty. This paper shows that you can get by with only standard expressive means of ontologies (existing constructions of the OWL language), or supplement the OWL language with such constructions so that the display of variability in the ontology would be simpler and clearer.

The addition of the OWL language standard with new constructs to reflect the variability of the domain leads us to the creation of not static ontologies, as they were before, but to dynamic ontologies. Dynamic ontology, that is, an ontology that changes over time, is not only a description of the domain, but also a kind of model. Thus, using only one such dynamic ontology, you can achieve two goals at once: describe the domain and create a model of some process or phenomenon. It is quite obvious that the dynamic ontology paradigm requires further development and this article is only the first steps in this direction.

Acknowledgments

The author thanks his family and friends for their moral and financial support during the writing of this work.

References

[1] F. Tyutchev, Silentium!

[2] J. L. Hudson; J. C. Mankin, Chaos in the Belousov–Zhabotinskii reaction, 1981.

[3] I. Yefremov, Razor's Edge, 1963

[4] Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd edition, 2009

[5] Davies John, Struder Rudi, Warren Paul, Semantic Web Technologies, 2006.

[6] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), Available online: https://www.w3.org/TR/owl2-syntax/ (Accessed: 17.02.2024)

[7] https://www.orano.group/en/unpacking-nuclear/all-about-radioactivity (Accessed: 5.03.2024)

[8] Physical, Nuclear, and Chemical Properties of Plutonium, Available online: https://ieer.org/resource/nuclear-power/plutonium-factsheet/ (Accessed: 11.03.2024)

[9] Ts. Yu. Oganessian, K. V. Utyonkov, D. N. Kovrizhnykh, et al., New isotope 286Mc produced in the 243Am+48Ca reaction, 2022

[10] Rao Jinghai, Dimitrov Dimitar and others, A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP's Guided Procedures Framework, 2006

[11] Mondorf Ansgar, Herborn Timo, Ontology-based process mediation in the European project BRITE, 2008

[12] Fathalipour Mahmood, Ontology-Based, Process-Oriented and Society-Independent Agent System for Cloud Computing, 2013

[13] Verhodubs Olegs, Adaptation of the Jena Framework for fuzzy reasoning in the Semantic Web Expert System, 2014

[14] https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/ (Accessed: 22.04.2024)

[15] https://www.geeksforgeeks.org/python-dictionary/ (Accessed: 22.04.2024)

[16] https://www.w3.org/Style/CSS/Overview.en.html (Accessed: 26.04.2024)

[17] https://www.w3schools.com/css/css3_animations.asp (Accessed: 26.04.2024)