erlend-rdf

Why RDF?

When I started on the MLR(ISO/IEC 19788) and MLO(EN 15982) work I tried to better understand what we would benefit from a "semantic web" approach. From this I also realized what new possibilities and challenges we would face by following this approach, and what role we as standardizers should focus on, and what we should leave to the specific domains or stakeholders or user groups or future requirements or changes in technology or new things we cannot envisage today.

And as with my view on contex I would apologize for my lack of consistency with the terms I'm using on property and concept (it is confusing for others, and sometimes for me as well).

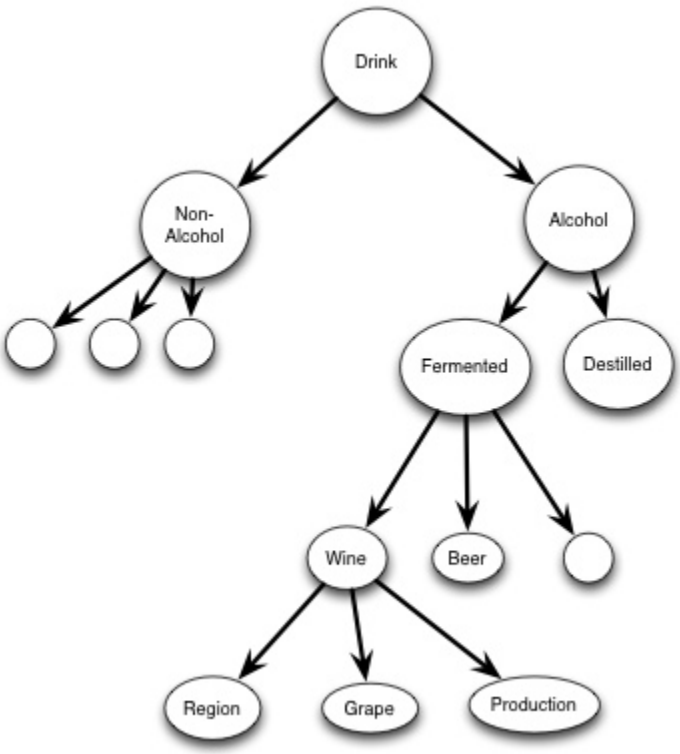
The example

Our challenge is to identify a well-defined concept, and then to identify the different classifications (sub-concepts) this could consist of. And then provide a flexible way of using the information model.

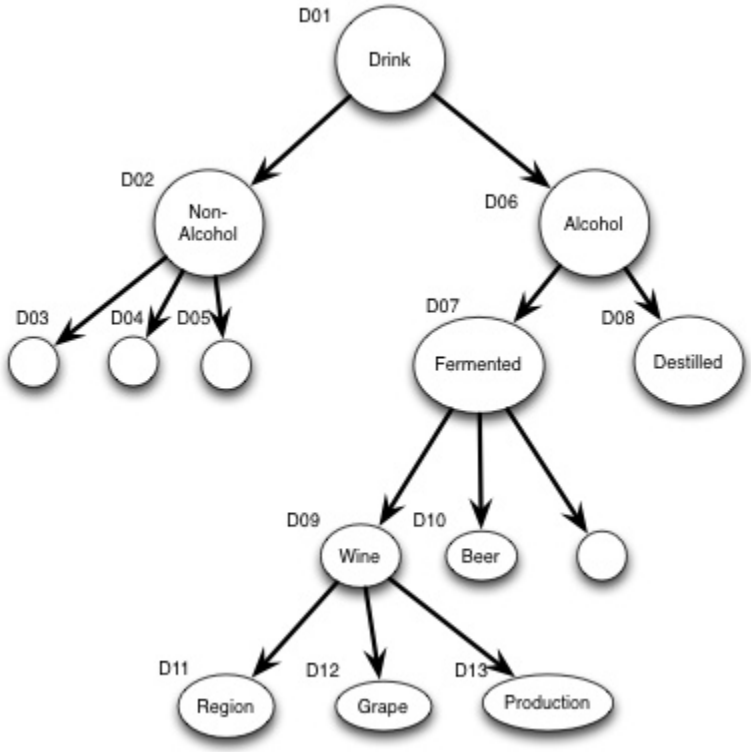
The concept

Drink: something fluid, that are consumable by humans
With this as the concept, the question is then how could we provide sub-concepts, or a classification model for this concept.

One possible way of classifying a "Drink" could be in Alcohol and Non-Alcohol drinks.
And note that the three structure only implies the relationship and inheritance of concepts, and not how the transferred information package that are sent between systems would look like.



And if we add identifiers to this we would have the possibility to address the different concepts, and to determine the relationship between them.
These identifiers should be globally unique, and machine-readable and machine processable.
It cannot be stressed and emphasized often enough that there is a difference between what the machines see, and what the human user see. Ideally a human (end user) would never - ever see any of the stuff we are discussing here, the user interface tools will generate this information, and the system will process and act based on this information.

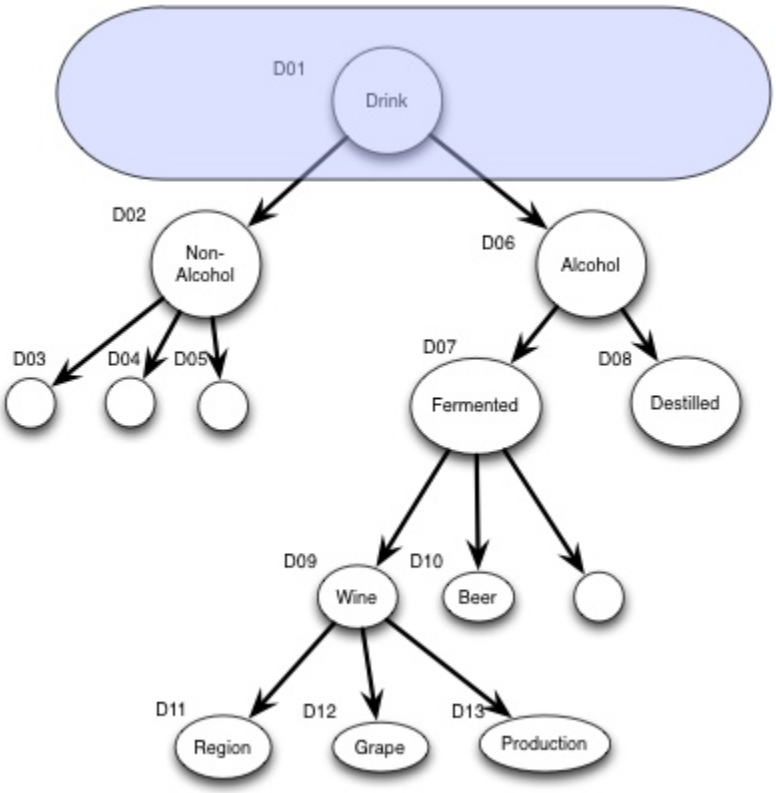


What to standardize?

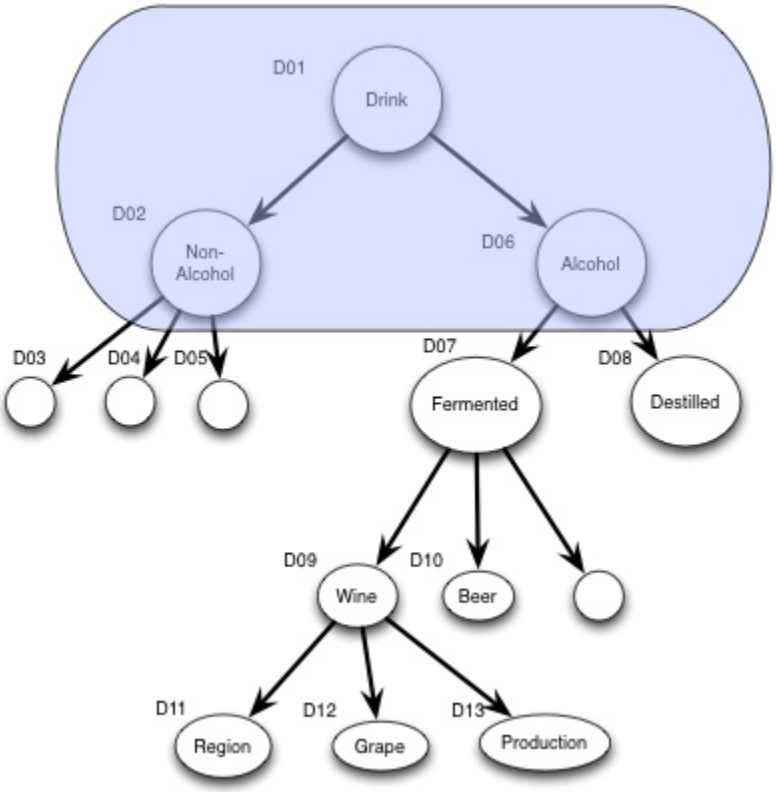
In this rather arbitrary example, I have not used any concepts that are related to the task at hand, there are two reasons for that.
1) I'm sure that we all have the capacity to do some abstract thinking
2) If I had used concepts from AfA/24751 or others we would not discuss the principle, but the actual concepts. As a result we would not have the discussion we need to have.

It is important to distinguish between what is globally relevant, and what is application/vendor/locally relevant. I would argue that we should focus on concepts that are globally relevant, and leave to the different applications/vendors to extend what is globally to meet their specific needs. Combining this would provide a model that is sustainable, and that will have relevance even if there are huge changes in technology and paradigms for how we use technology.
If we find (trough a registry) that all applications/vendors use the same concept in the same way, we might consider to promote that to the standards level.

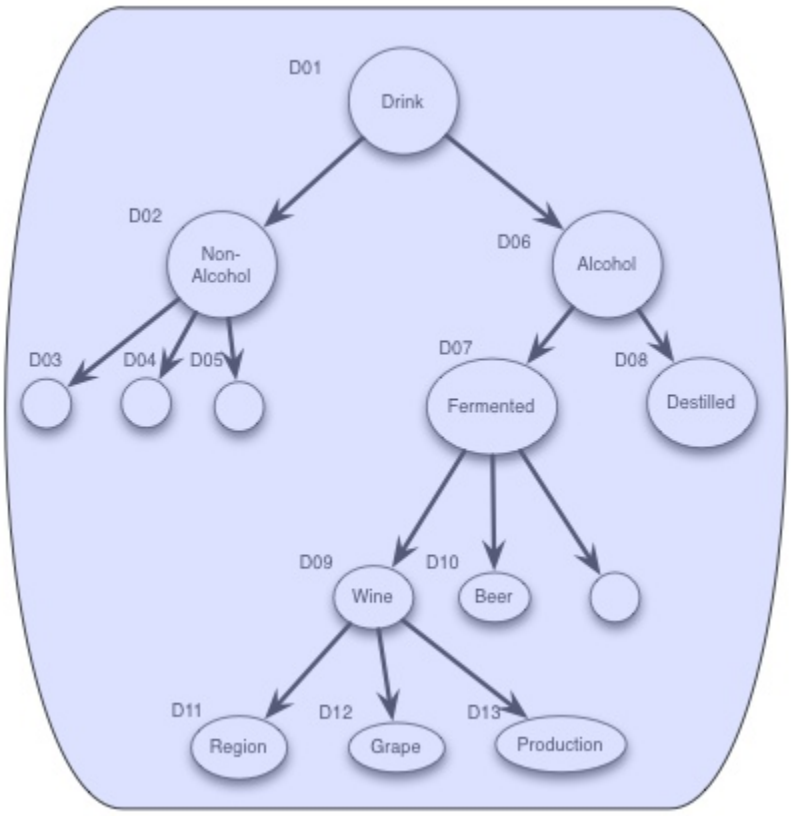
So we could only standardize the **core concept**



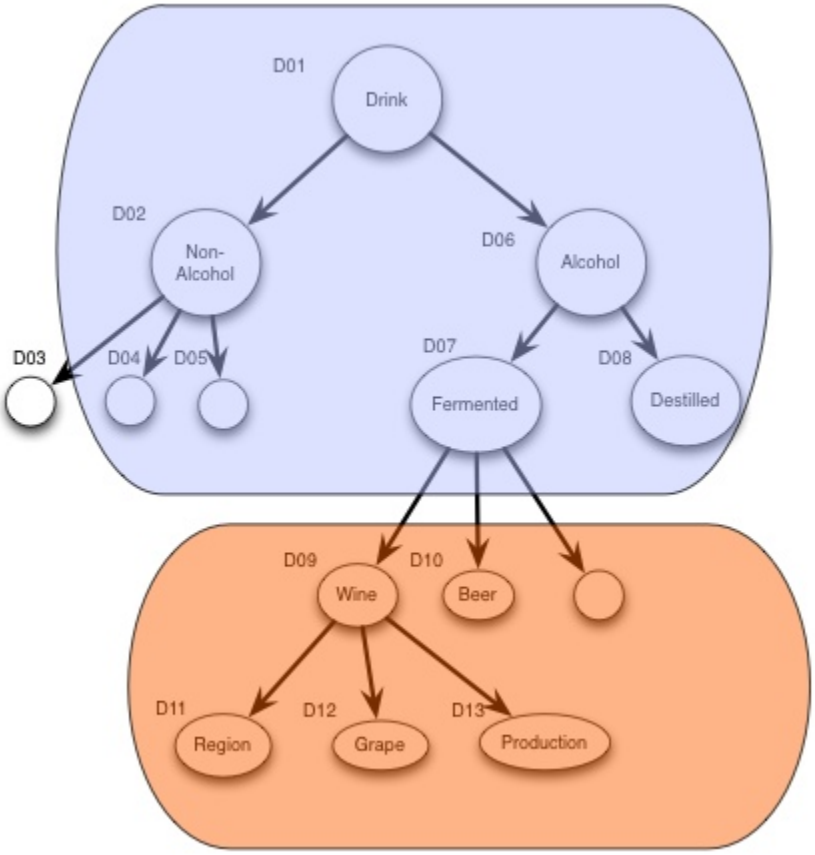
Or we could standard the some of the main classifications:



Or we could standardize everything, and claim this as a fixed model.



I would argue that we should standardize common concepts, and leave to the application/vendors to specify a model that meet their requirements, that are based on the standardized model.



What to make common would always be an important discussion, however it

> ***NOTE***
>
> The syntax is not correct, and at best it will work as an illustration.

Based on this we will then have a model that looks something like this:
ID's should be expanded

id		
D01	owl:class	Drink
D02	owl:class	Non-Alchohol
D02	owl:subClassOf	D01
D03	owl:class	CO2 added
D03	owl:subClassOf	D02
D04	owl:class	CO2 added
D04	owl:subClassOf	D02
D05	owl:class	Still
D05	owl:subClassOf	D02
D06	owl:class	Alcohol
D06	owl:subClassOf	D01
D07	owl:class	Fermented
D07	owl:subClassOf	D06
D08	owl:class	Destilled
D08	owl:subClassOf	D06
D09	owl:class	Wine

D09	owl:subClassOf	D07
D10	owl:class	Beer
D10	owl:subClassOf	D07
V01	owl:class	Region
V01	owl:subClassOf	D09
V02	owl:class	Grape
V02	owl:subClassOf	D09
V03	owl:class	Production
V03	owl:subClassOf	V09

Note: the data model will only be three columns (triplet), and if we need to add more complex models, we do not need to change the database model. This guarantees for robustness and flexibility.

The ID should be a URI globally unique. i.e. we should replace the **"D"** with http://iso.org/iso/iec/jtc1/sc36/24751/2013/
and the vendor specific **"V"** could be replaced with http://barolo.com/La_Morra/

If now another local vendor in Italy need to express more detailed information e.g. what type of soil the wine are from, he then extend the model to meet his requirements, and finding that soil is an extension of region, a value he finds in the **"V"** extensions in a registry he would then add the following to his system: and the **"I"** would be replaced with http://rioja.com/faustino/

ID		
I01	owl:class	soil
I01	owl:subClassOf	V01

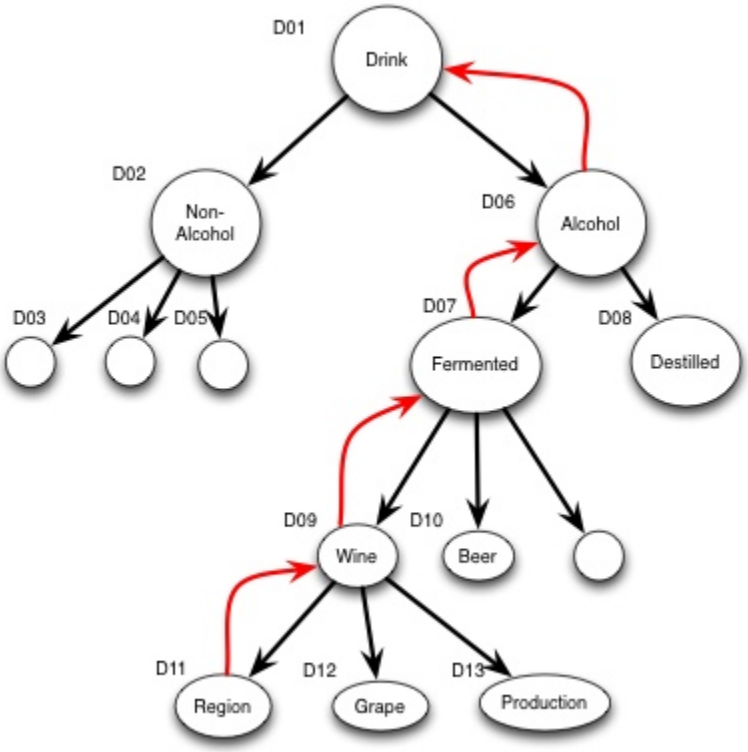
If we choose the MLR approach having DC as a superclass, an entry of data could then look like this in the triple.

E01	owl:typeOf	I01
E01	dc:title	chalk
E01	dc:resource	http://www.bodegasfaustino.com/
E01	dc:title	Faustino wines
E01	dc:description	Wines from a region in Spain - that tastes great...

Graceful degradation

Exchanging information between systems from a vendor or within the same environment, there is usually no problem with interoperability. But what happens if my system receives the triple with chalk as soil?

Fortunately since this is built on the same principles we could have the machine find where the concept "soil" comes from by following the pointers:



So the computer will then follow the **subClassOf** pointer until it finds something it already knows, and that should at least be the concepts we have standardized. So the computer will then know that the "soil:chalk" is related to "alcoholic drink". What the system then does with the information is up to the rules of the system to decide. With this model interoperability will not break, and we will have graceful degradation.

The 24751 case

We will be able to support any request from any vendors for their need to specify complex super-fine details of their specific system, and if a users decides to take the same set of preferences to a different system - the system could still adapt to some extent.

We should do our best to avoid that users need to have preferences for all applications, for all devices for all platforms.
Even if a preference is extremely detailed, it should be able to use a detailed profile on a more generic system without the possibility to meet any of the detailed requirements.

And again - we need to focus on the need of the individual user, and I would assume that the individual user would not specify preferences for all applications, for all devices, for all platforms. We should not develop a model that have one user - one device as an assertion, but more the model that we have one user - many devices. And where one profile should work on as many devices as possible... And that the user shall never have to see the details of the model, but only manipulate these variables from a intuitive simple user interface...