Sushi Ontology Report

The goal of this ontology is to create an intelligent menu using the "Yo! Sushi menu".

The ontology development process is discussed in this report. This includes competency questions, knowledge acquisition techniques, areas that are modelled and used axiom patterns.

# Competency Questions (CQ)

The competency questions helped in forming an understanding of the areas needs to be modelled in the ontology. They, also, helped in determining the scope and content of the ontology. Moreover, CQs can be used to evaluate the ontology. For example, one can write a query to list the vegetarian dishes; then checks whether all the vegetation dishes are listed.

Here are some of the CQs that helped developing the ontology:

 What are the ingredients from which sushi are made?

 What are the characteristics of those ingredients?

 What are the types of sushi, and from which ingredients are they made?

From those CQs, the following can be derived about the ontology:

- Scope: the sushi part of the "YO! menu".

- Contents: sushi ingredients, sushi dishes, sushi types, prices, calorific contents, etc.

- Relationships: between sushi dishes and their prices or calorific contents, etc.

# Knowledge Accusation (KA)

The first step in the knowledge acquisition was to extract key terms from the menu and use them to build a list of the sushi ingredients; but having ingredients as a list was not sufficient to answer the given CQs. For an instance, the ontology will not be able to list seafood dishes. Therefore, the following KA techniques were applied:

1. **Card sorting:**

This technique was used in classifying the whole list of ingredients into seven main classes (Fruit, Vegetable, Grains, Meat, Seafood, Seasoning and Dressing) . Those classes were added as subclasses of the main sushi ingredient class; grouping the related ingredients as subclasses of them, as shown in *figure1.*

√ This helps in answering some of the CQs, such as listing sushi containing seafood.

*Namedsushi and hasIngredient some Seafood*

This process was repeated for each subclass of the sushi ingredients in order to be able to answer CQs in a more sufficient way. *Figure 1* shows the final ingredients hierarchy.

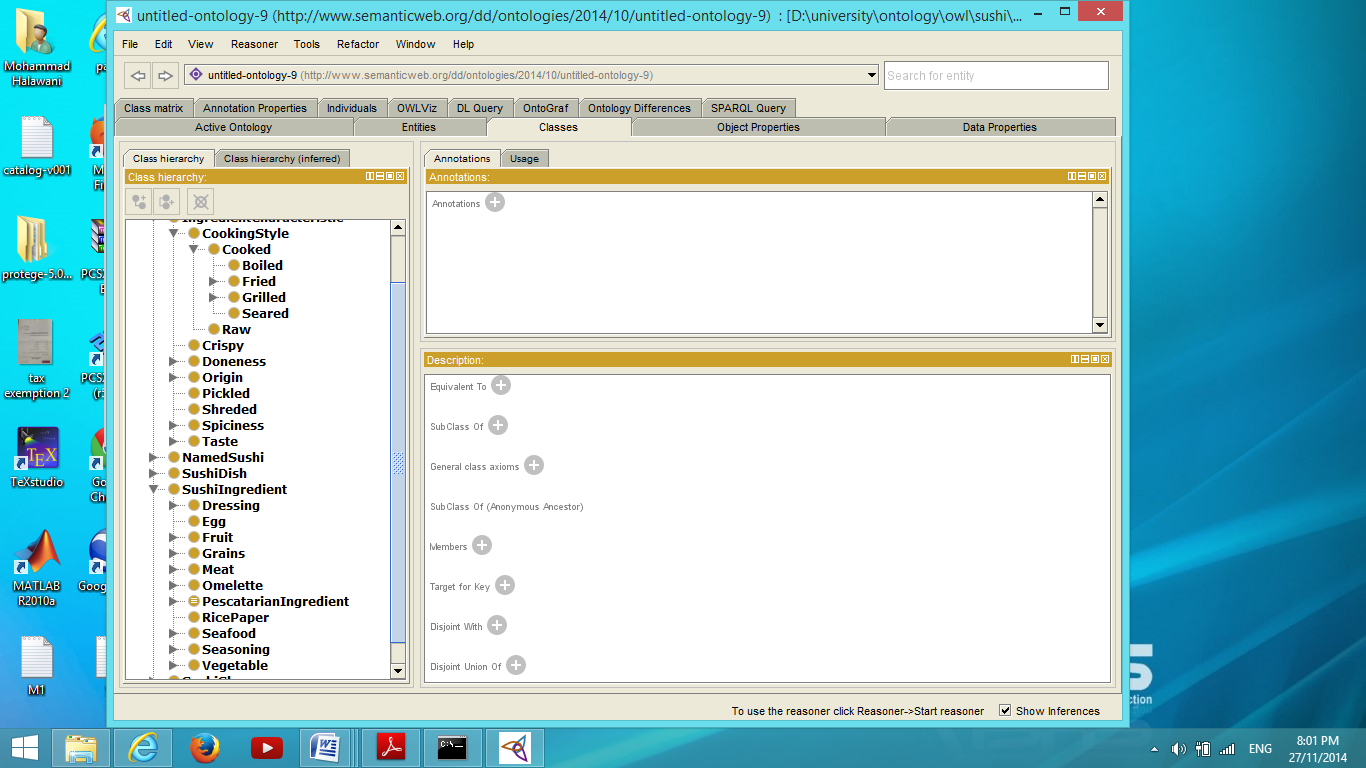


Figure1: Sushi ingredients

1. **The ‘3 Cards Trick’:**

After using the ‘3 Cards Trick’ technique on three of the subclasses of the Seafood class (Nori, Kaiso and Crabstick), one could realise that Nori and Kaiso are sharing some properties that differ from Crabstick. Both of them are, actually, seaweeds. Thus, a new class ‘Seaweed' has been introduced as a subclass of Seafood and as a superclass of both of the Nori and Kaiso classes.

This helped in defining vegetarian ingredients class in a more robust way. Thus, after defining pescatarian ingredients class, vegetarian ingredients class can be defined as follows:

*PescatarianIngredient and (not (Seafood and not (Seaweed)))*

instead of the following definition:

*PescatarianIngredient and (not (Crabstick or Fish or FishRoe or SalmonSkin or ShellFish or YellowfinTuna))*

It worth mentioning that definable classes are made to make the query writing process more convenient. For example, instead of writing one of the previous codes to query vegetarian ingredients, one could just write:

*VegetarianIngredient*

# Areas modelled

As previously mentioned, the CQs helped in determining the ontology's scope. Within this scope the following needs to be modelled:

* Self standing entities:
* SushiType: contains the seven sushi types: Nigiri, Sashimi, ISO, Maki, Futomaki, Handrolls and Gunkan; with Futomaki as a subclass of Maki. Each sushi type has a certain shape.
* SushiDish: contains all dishes listed in the menu. Each sushi dish consists of a number of sushis and has a color.
* NamedSushi: the piece of sushi itself. Each named sushi contains some ingredients and has a type.
* SushiIngredient: the ingredients hierarchy.
* DishColour: contains six dish colours orange, light blue, dark blue, pink, green and grey. Those colours indicate the dish prices. Thus, it helps in answering cost related questions. √ For example: what are the dishes that cost less than £4?

SushiDish and hasDishColour some (DishColour and hasCost some decimal [ < 4.0 ] )

* SushiShape: some those shapes are: Cone, Sliced, ISORoll, and FatRoll.

This class helps in answering shape related questions.

√ For example: what are the sushis that have cone shape?

NamedSushi and HasType some (SushiType and hasShape some Cone)

* Modifiers:

All ingredient modifiers were put under a class called *IngredientCharacteristics.*

Those classes help in answering CQs related to the ingredients characteristics.

√ For example:

- what are the sushis that contains seared meat?

NamedSushi and hasIngredient some (Meat and HasCookingStyle some Seared)

- what are the spicy sushis?

NamedSushi and hasIngredient some (SushiIngredient and hasSpiciness some Spicy)

*Figure 2* shows some of those classes and their subclasses.

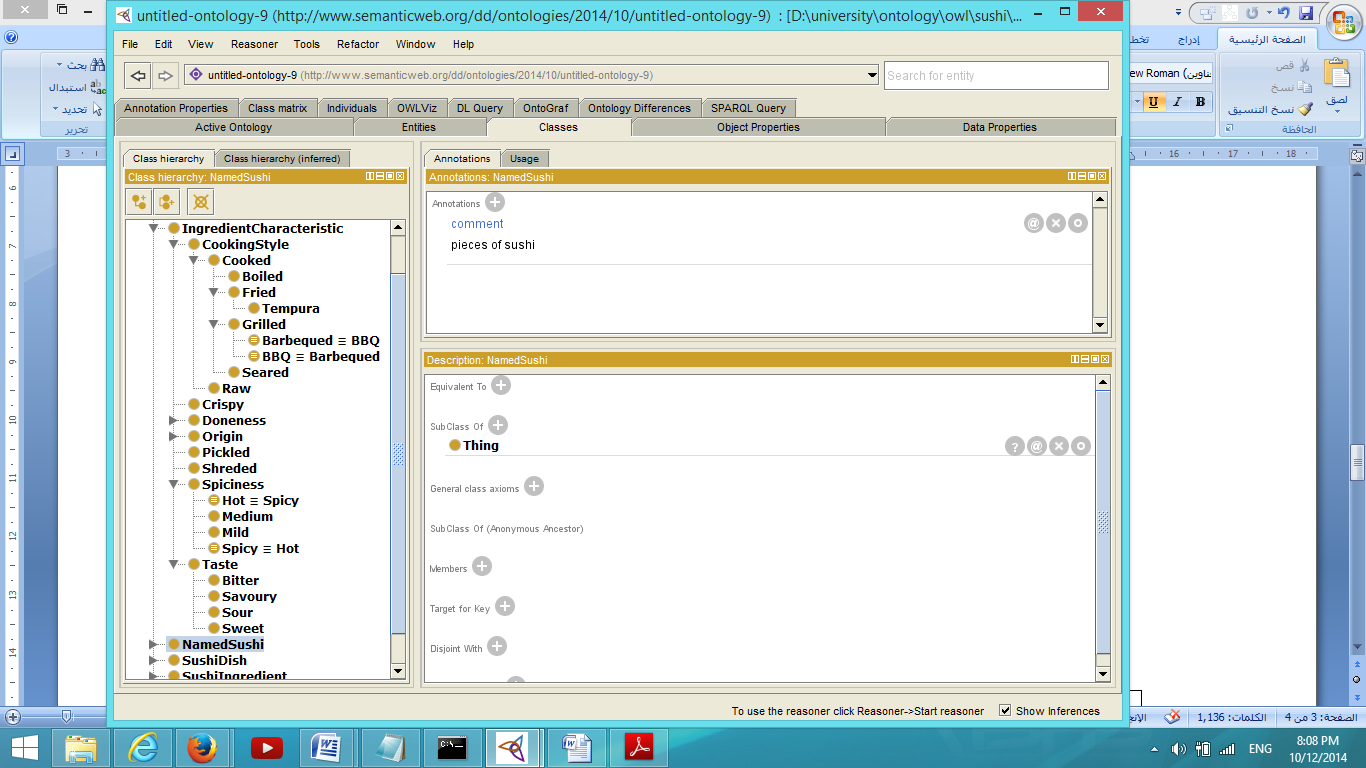


Figure 2: IngredientCharacteristic Class that contains the modifiers

* Relations:

The relations between classes are represented using the object properties. Some of those properties and their domains and ranges are shown in the following table.

|  |  |  |
| --- | --- | --- |
| **Object property** | **Domain** | **Range** |
| hasDishColour | SushiDish | DishColour |
| hasContent | SushiDish | NamedSushi |
| hasType | NamedSushi | SushiType |
| hasCookingStyle | SushiIngredient | CookingStyle |
| hasSpiciness | SushiIngredient | Spiciness |
| hasTaste | SushiIngredient | Taste |
| hasIngredient | - | SushiIngredient |
| hasShape | - | SushiShape |

It worth mentioning that the 'hasCookingStyle', 'hasSpiciness' and 'hasTaste' are sub-properties of the 'hasIngredientCharacteristic' property which has more sub-properties. Moreover, the 'hasIngredient' property is a super-property of the 'hasFilling', 'hasTopping' and 'hasCrust' properties which represents how the ingredients form a piece of sushi (i.e. is it a filling or topping?). Furthermore, the domains and/or ranges of some object properties, such as ' hasIngredient' and ' hasShape', are not specified because of the fact that they might have more than one domain and/or range.

In addition to the object properties, the following two data properties are used to represent the costs and calorific values of each sushi dish.

|  |  |  |
| --- | --- | --- |
| **Data property** | **Domain** | **Range** |
| hasCalorificValue | SushiDish | Integer |
| hasCost | DishColour | Decimal |

Although the 'hasCost' property's domain is ' DishColour ', the cost of a dish can be determined. For example, one can write a query similar to the following.

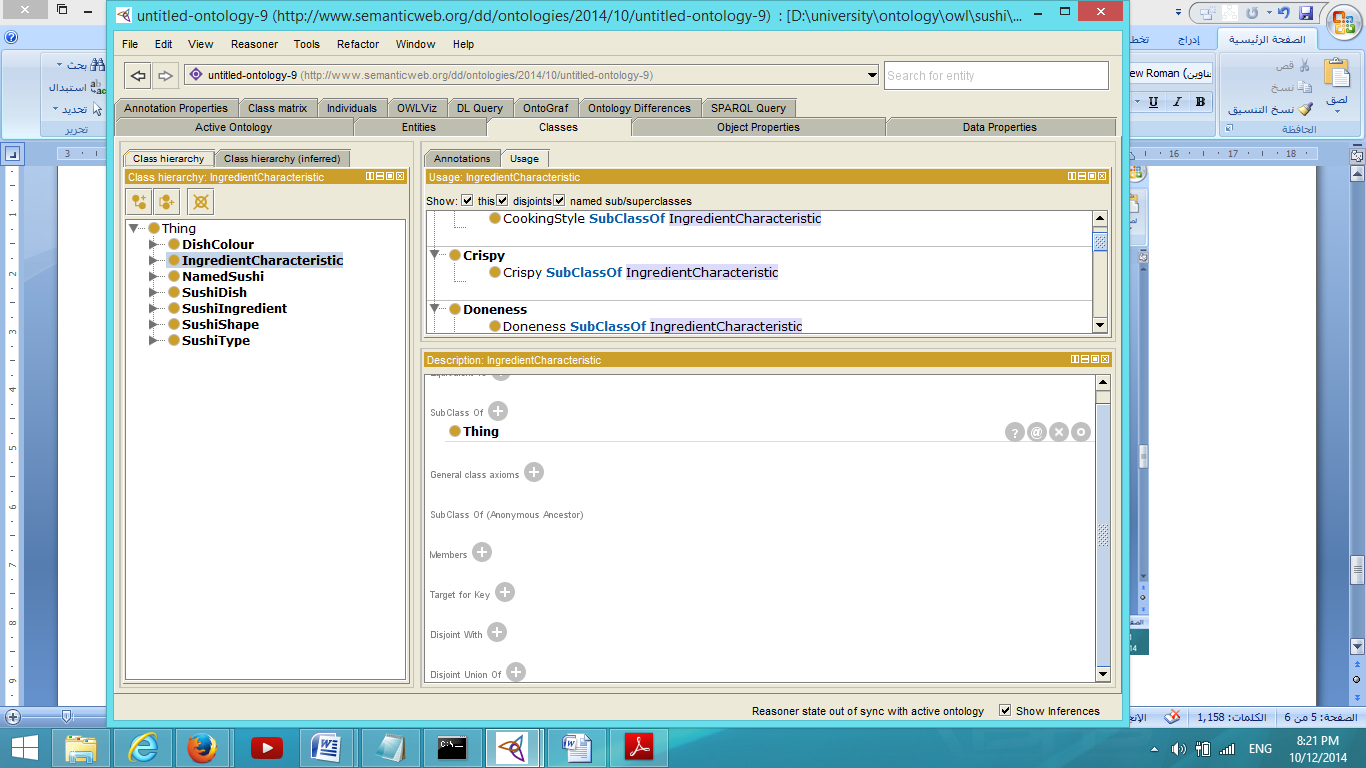
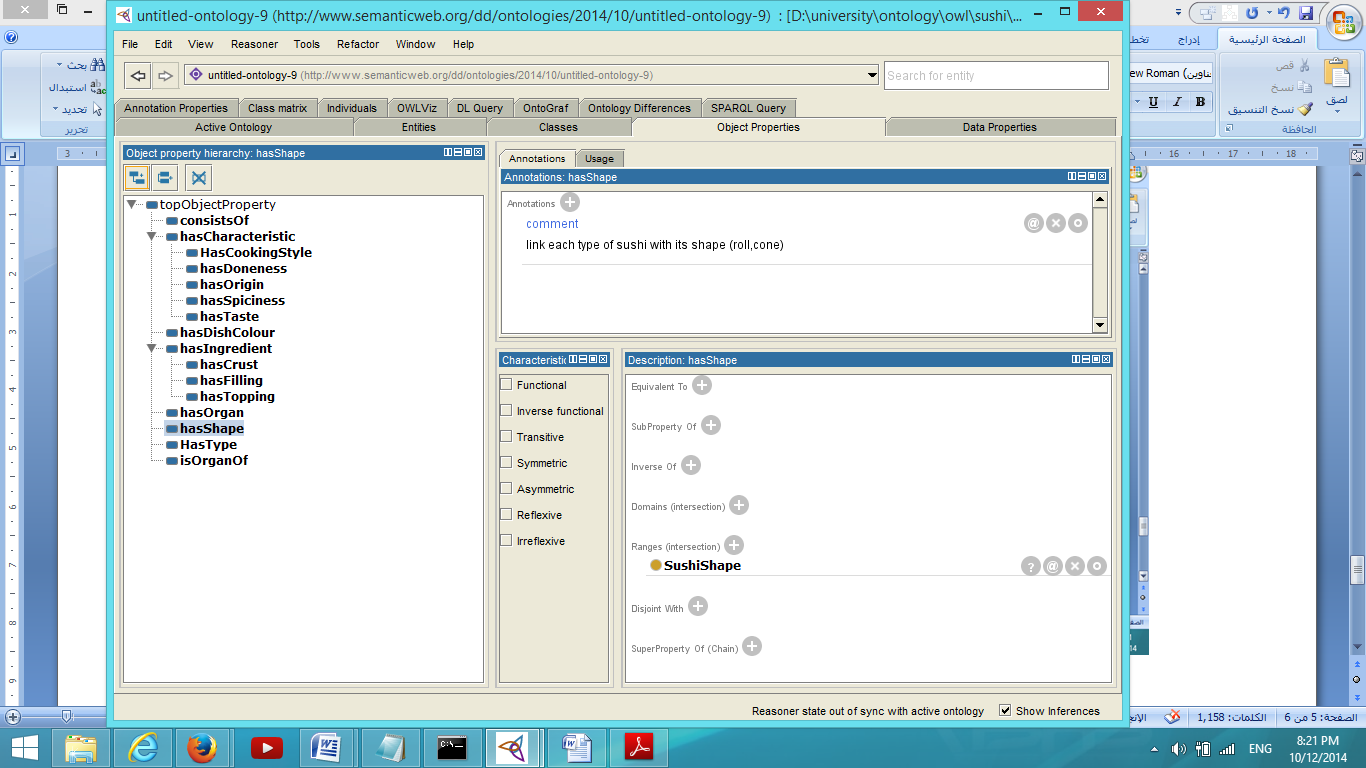
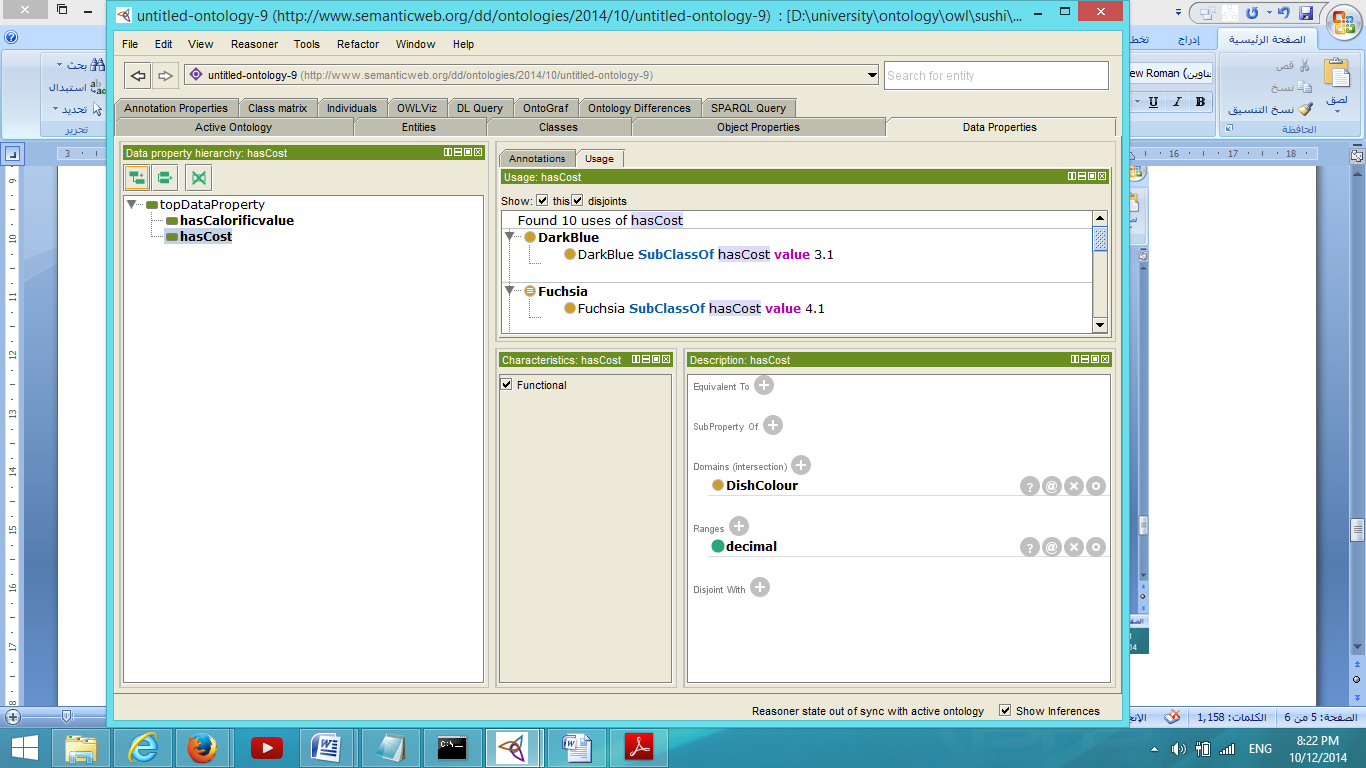
SushiDish and hasDishColour some (DishColour and hasCost some decimal [< 4.0] )

* Definable Classes

As previously mentioned, in this ontology definable classes are made to make the query writing process more convenient. The following are some of those classes:

PescatarianIngredient, VegetarianIngredient, VeganIngredient, PescatarianDish, VegetarianDish and VeganDish.

Finally, this is an overview of the classes and properties in the sushi ontology:

# Axiom Patterns

The current sushi ontology is using the following patterns:

* **Normalization:**

This pattern was used in the start of building the ontology when the sushi ingredients class hierarchy was built. In the sushi menu, the modifiers were put in different classes than the real ingredients (i.e. dimensions were separated). For example, in the menu, there is an ingredient called seared beef. For this ingredient, we put the beef as a subclass of sushi ingredient class; and the seared (modifier) was put as a subclass of the cooking style class. Under cooking style class, the cooking styles where normalized to the xxx-ed form, such as the fried and seared classes (as shown in *figure3*).

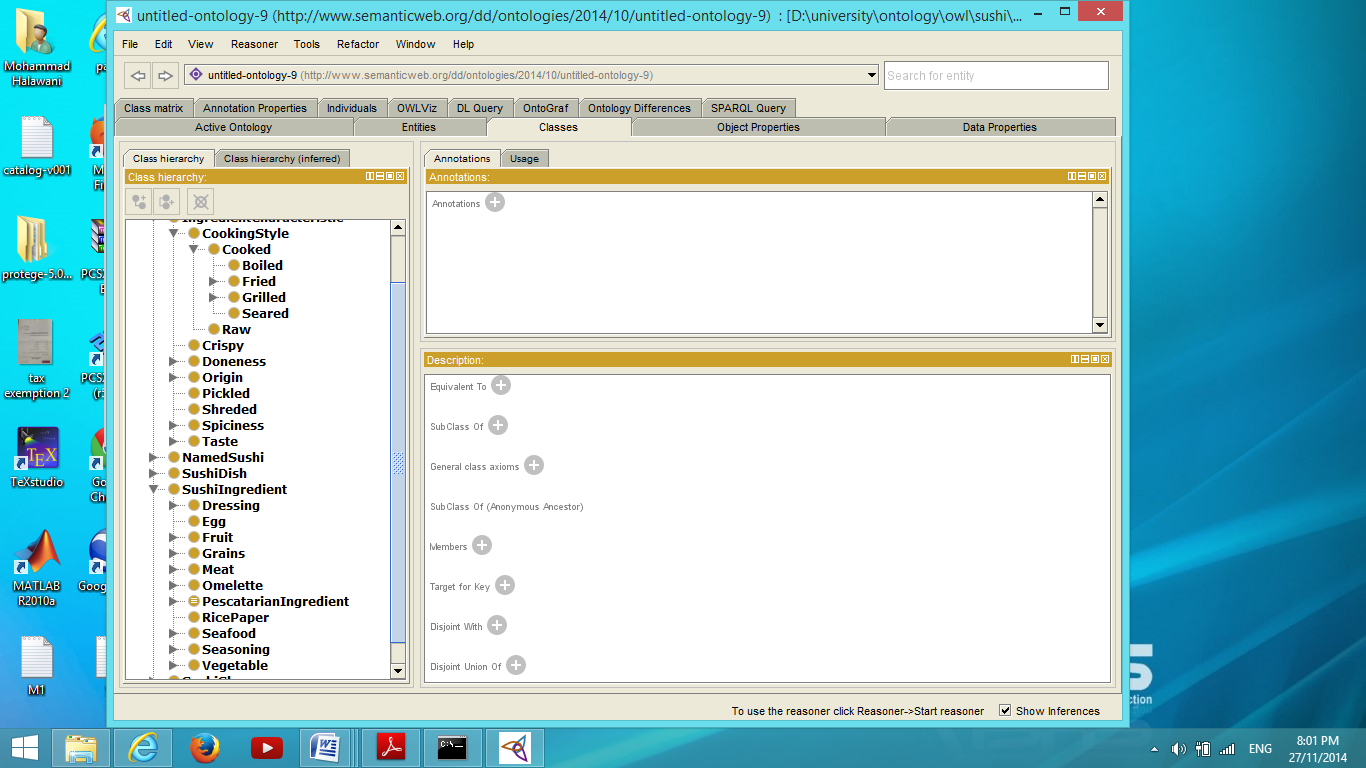
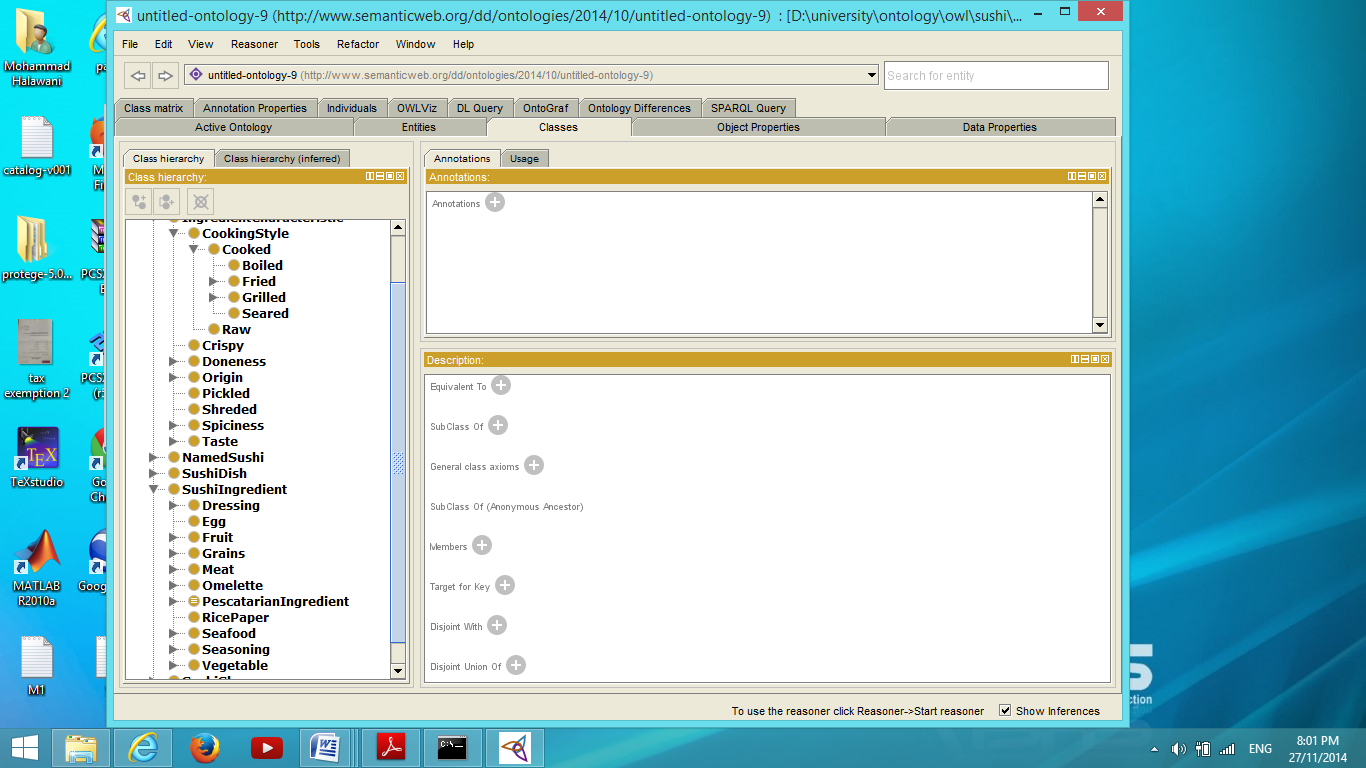


Figure 3 : sushi ingredient and cooking style class hierarchies.

* **EPQ:**

As previously mentioned, data properties and object properties were used to represent relationships between classes and data characteristics of those classes.

* **Closure axioms :**

They are used in all of the dishes and the named sushi classes in order to close the open world assumption (OWA) of the ontology. In the following two examples, the closure axioms ensure that there are no other ingredients in the sushi or sushi pieces in the dish than the ones asserted. Thus, the reasoner can close the OWL's open world assumption.

*Class: TamagoNigire*

*SubClassOf: NamedSushi*

*hasIngredient some Rice*

*hasTopping some Tamago*

***hasIngredient only (Rice or Tamago)***

*Class: TamagoNigireDish*

*SubClassOf: SushiDish*

*hasContent exactly 2 TamagoNigiri* ***hasContent only TamagoNigiri***

The following queries to obtain vegan sushis and sushi dishes will return *nothing*, in case of not including closure axioms; since the reasoner, based on the OWA, will assume that there might be other, non- vegan, ingredients or sushis which are not asserted.

NamedSushi and (hasIngredient only VeganIngredient)

SushiDish and (*hasContent* only (NamedSushi and (hasIngredient only VeganIngredient)))

√ Therefore, closure axioms helped answering CQs related to some dietary restrictions.

Furthermore, it worth mentioning that there are some ingredients, and consequently some dishes, which are neither vegan nor not vegan because of this OWA. For an instance, the 'AsianDressing' does not have an asserted egg ingredients; but because we are not sure whether it might has, it is neither considered as a vegan nor not vegan ingredient. Thus, it might be a vegan ingredient, or it might not. The same applies when querying non-spicy, or non-sweet, ingredients. Therefore, since it is not specified for most of the ingredients to be not spicy, or not sweet; the result of the query is *nothing*.

* **Value partitioning pattern and Covering axioms :**

The value partitioning pattern is used in the *Spiciness* class where its values are partitioned to *Spicy, Medium and Mild* classes, as shown in *figure 4*, and then covered by them using the covering axiom *Spiciness subclass of: (Spicy or Medium or Mild).* Using the covering axioms states to the reasoner that these are the only possible subclasses of the spiciness class. Thus, it closes the OWA.

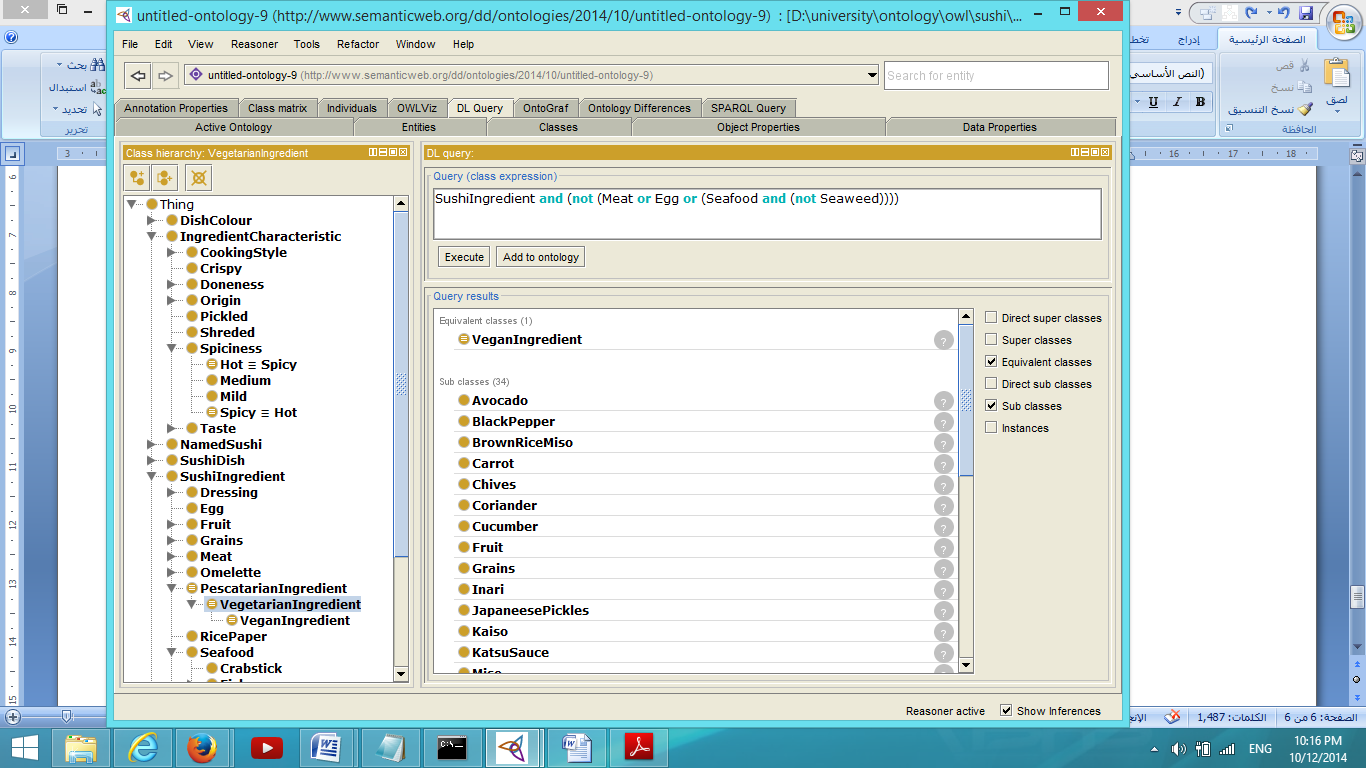


Figure 4. Spiciness value partitioning

# Robustness

For adding new sushi pieces or sushi dishes, one needs to follow their patterns.

**An example**:

Adding a new grey sushi dish that consists of one piece of a new maki sushi called ‘X’ that just has one ingredient ‘Ing’ (the dish’s calorific value is 120).

NamedSushi pattern SushiDish pattern

*Class: X*

*SubClassOf: NamedSushi*

*hasType some Maki*

*hasIngredient some Ing*

*hasIngredient only (Ing)*

*Class: XDish*

*SubClassOf: SushiDish*

*hasContent exactly 1 X*

*hasContent only X*

*hasCalorificvalue value 120*

*hasDishColour some Grey*

In conclusion, the discussed ontology answered most of the competency questions except for some of the religious restrictions ones. For example, since the animals killing method (slaughtering, electrical shock, etc.) is not considered in the modelling world; certain decisions, such as whether the meat is Halal, cannot be inferred. This can be done by adding new class *KillingMethod* ,that includes the killing method as subclasses, as a subclass of *IngredientCharacterestics*, and adding a new object property to link it with the *Meat* class.