[Date]

Social Food Network GRUB

Ontology Model

Eoin Dalton
20070289

Chapter 1 Contents

Chapter 2 Introduction

For this Module we are asked to create an ontology. The name of the Module is called knowledge systems engineering and an ontology is a way to describe knowledge about a domain or concept which is why we are asked to develop an ontology. For this particular project we were giving an ontology called FindACara. This is a simple ontology that describes knowledge about a person. That person has a brother or sister and is from a country. What we were asked to do was extend or adapt this ontology to incorporate a social networking application.

In this document it will take you through the process of developing an ontology. It will follow a number of steps that was shown to use in the Lecture notes. The steps will look at scope/domain that the ontology is being build on. It will look at ontology reuse. This involves reusing existing ontologies that have being previously been defined. Other steps involved will be encoding and class hierarchy, defining properties, axioms and the creation of individual/instance of the different concepts. You will be able to see graphical tool used to visualize what the ontology looks like. There will be a file attached that will contain all the OWL code that is generated from Protégé which will be fully commented.

Chapter 3 Scope

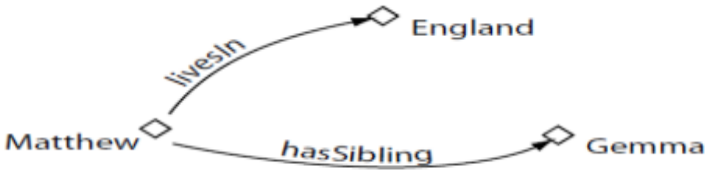

Figure 3.1: FindACara Ontology

To the left in figure 3.1 is a basic ontology called FindACara. This is the starting point for this project in which the focus will be on extending and adapting the FindACara ontology to suit some form of social networking application. After reviewing the FindACara ontology seen in figure 3.1, seen is two people and a country. For a start there is two concepts, which are concept of a person and the concept of a country. The question that was posed is how this can be adapted to suit a social network of our choice.

A follow student and friend by the name of William Wall who is currently in the 4th year of a software development course here in Waterford Institute of Technology is in the process of developing his 4th year project. What he is developing a social media web application that is focused around food. The core processes focus on:

- Creating
- Updating
- Deleting
- Searching

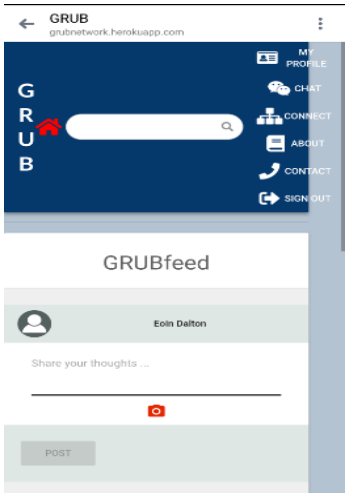

Figure 3.2 : GRUB social network App.

Users can create, update, delete their user accounts. They can search for friends and family members and add them as friends. Users can add posts, video and pictures and all of these processes are centred around food. Shown in figure 3.2 is the main profile of a user account in the GRUB app and you can see some of the functionality that was just mentioned.

That brings us onto this project. We are targeted with building an ontology for a social media app modelled on the FindACara ontology. So, this project is going to try model the ontology for the GRUB app that William is building this will be the scope, trying to model the processes involved this social food

networking app. For the purposes of this project there will be some processes added such as rating for restaurants and a favourite food section.

3.1 The range of users

As this is a social networking app for food there will be a wide verity of users. But unlike a university where it not difficult to list users such a students and lecturers and admin, a social media app is different because there may be such a wide verity of users to many concepts to define. For the purpose of this project there is just one generic user, this will be represented as one class called person. When instances are created later the use of object properties will define what this user are. For example, a person me be a sibling, spouse, parent and friend they could be one of these properties or all of these properties depending on the person status, but they will always be a person.

3.2 Purpose of Ontology

The purpose of this ontology is to model how the GRUB app will function and what are the relationships between the different functions. By mapping the functions into an ontology, it allows developer to see the relationship between the different classes. There are languages such as Apache Java, JQuery and Json that allows the concepts and axioms to be extracted and manipulated so that a user interface may be built on top of the model. This will enable greater interoperability and knowledge to be discovered by both user and machine.

3.3 Questions the ontology should answer

- What relationship does the person/user have with other people/users?
- What are the concepts that needs to be defined for the app?
- What relationships exists between concepts?
- What data properties needs to be stored?
- What is the interaction between different concepts?

Chapter 4 Reuse

The reuses of ontologies, classes and axioms is one the major benefits to building on ontology model. There are terms that have already been defined by professionals and are available to be reused by whoever needs them. For the GRUB application there will be a person which will represent users. When registering an account, users will be asked their full

address. Users can add their favourite food. There are three ontologies identified that can be reused for some of the needs of this application.

Person class = friend of a friend (foaf).

Found at: xmlns.com/foaf/spec/index.rdf (FOAF, 2014)

Food class = food ontology.

Found at: https://moodle.wit.ie/mod/folder/view.php?id=2794194 (chistyakov, et al., 2015)

Place class = place ontology.

Found at: https://www.youtube.com/watch?v=MbauHV2-XYw (Bikakis, 2019)

The first step in the reuse of an ontology is to identify what is needed, sometimes the whole ontology may be merged with the persons existing ontology or in some cases a person may only need some classes or axioms and terms. In protégé this can be done in 2 ways. The first is merging the entire ontology and the other is to copy the parts of the ontology that is required. The ontologies that were chosen were quite comprehensive and there were a lot of parts identified that of no use for this ontology. That meant that is was a case of copy the parts of the ontology that was need.

The first step in doing this if you refer to figure 4.1 is to navigate to Refactor on the top menu bar and select the copy/move delete Axioms that the arrow is pointing to in figure 4.1.

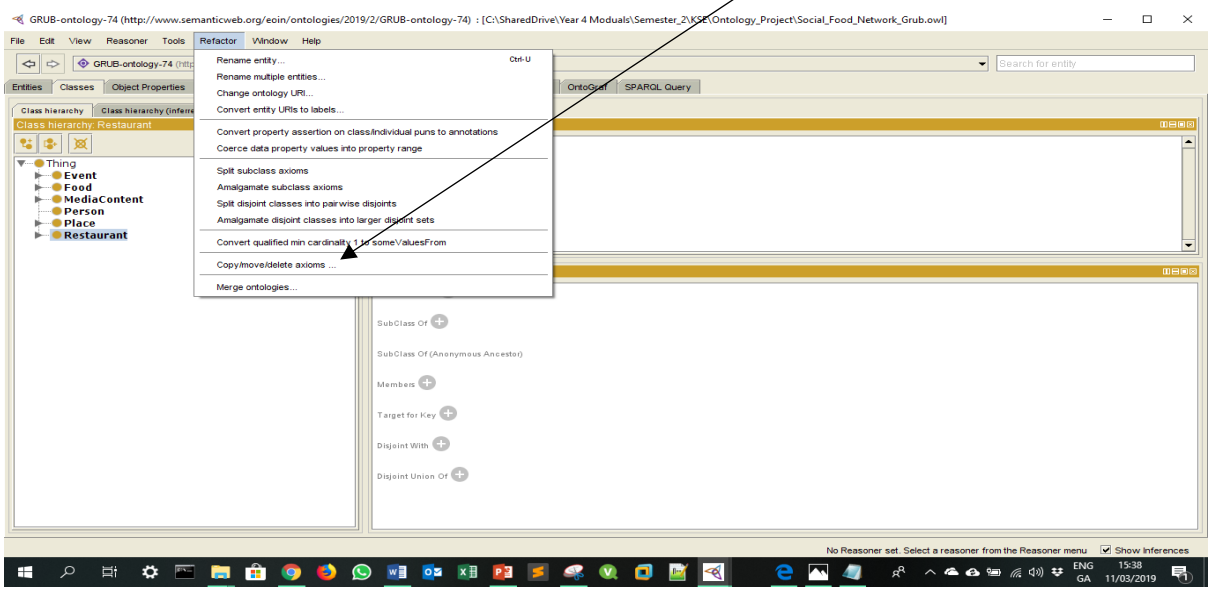

Figure 4.1: Copy Classes from Ontology.

After that the wizard will take you through a series of steps to complete the process. Referred to in figure 4.2 Protégé offer a section of methods in which you may want to copy/move or delete axioms. In this case the definition of the axioms was required as these definitions are agreed definition and will help with interoperability.

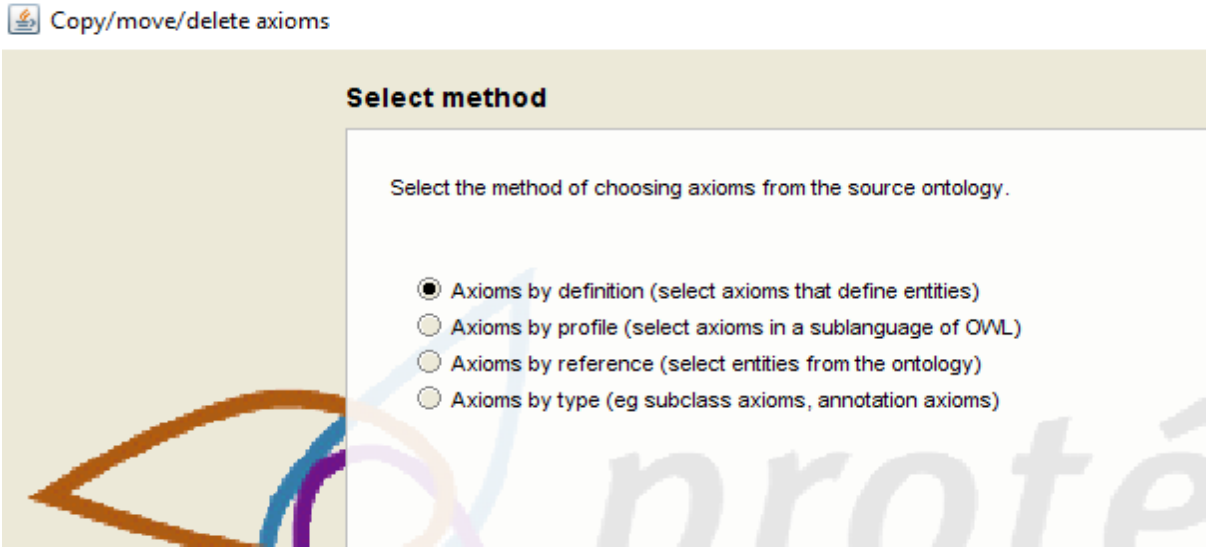

Figure 4.2: Selection method.

The next step involves choosing the axioms that are required as referred to in figure 4.3. Users have the option to choose from classes, object and data properties. Users may also choose individuals, datatypes and annotation properties.

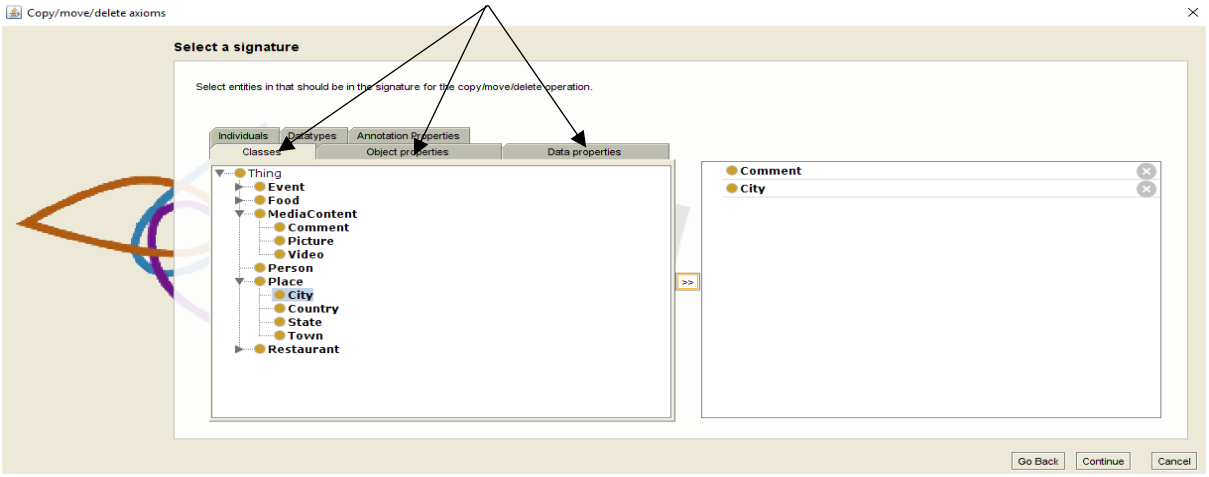

Figure 4.3: Choosing axioms.

The next step as referred to in figure 4.4 is whether the user wishes to copy, move or delete axioms and in this case the option was to copy axioms.

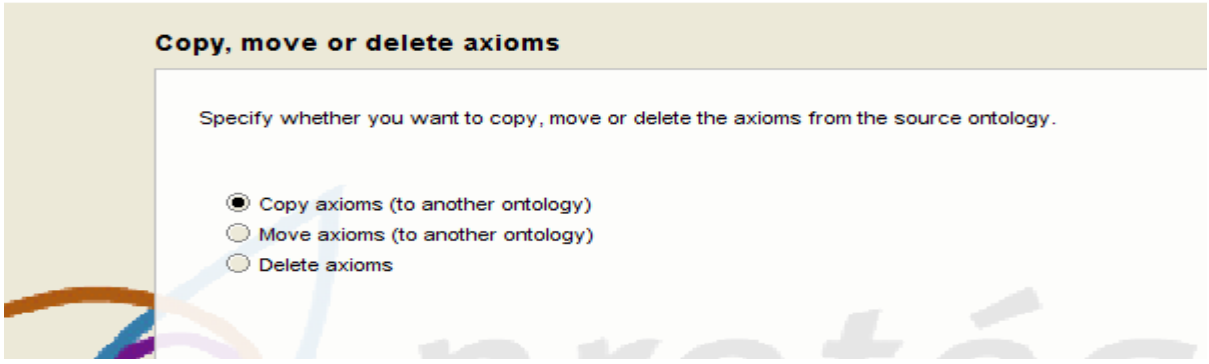

Figure 4.4: Copy, move or delete axioms.

One of the last steps as referred to in figure 4.5 was to choose whether to copy into an existing or new ontology. If the ontology is already created, it must be open in the same window as the ontology that you are copying from. That option is chosen when opening the ontology in the main interface. If you refer to figure 4.6 below this is the option if a user selects to copy to existing ontology.

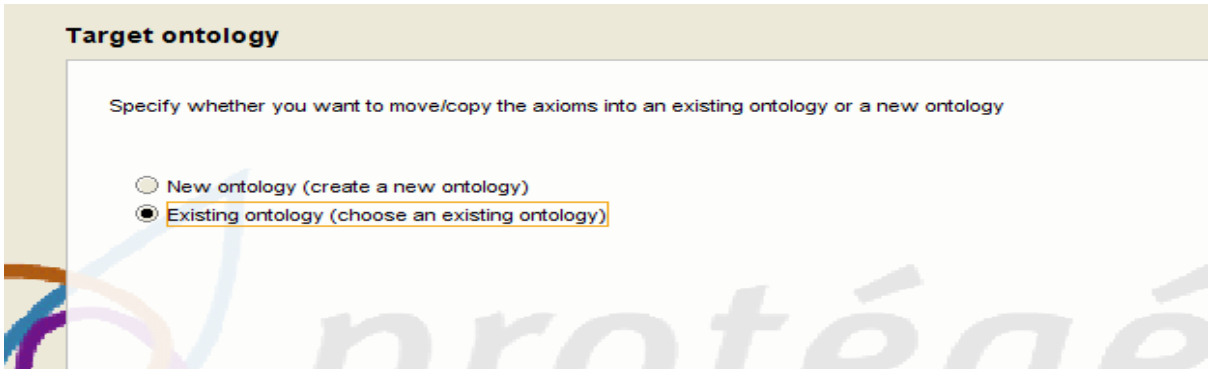

Figure 4.5: Choosing new or existing ontology.

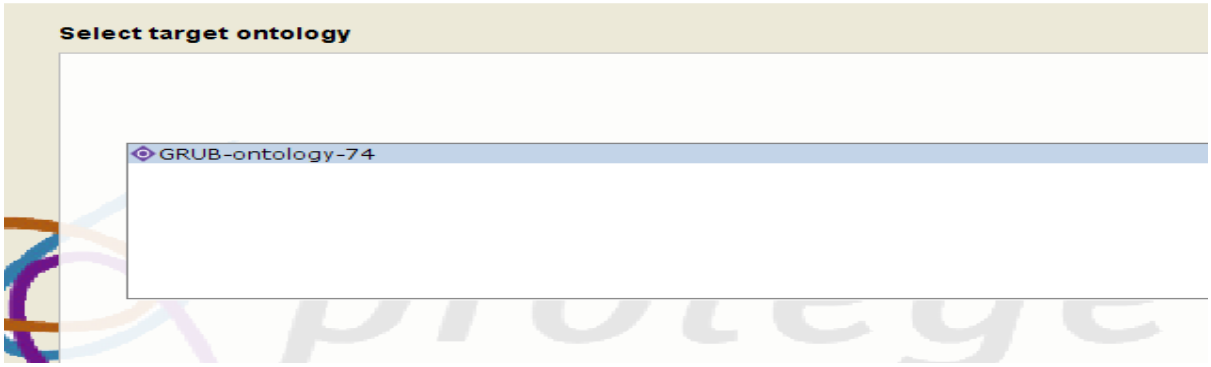

Figure 4.6: Choosing existing ontology.

The process that was just documented was the process that was used to copy all the axioms and classes for the ontology for the GRUB application. The food ontology is merged with the wine ontology and for this application wine wasn't require. The user requirements

were only focused on food types. All that was needed from the foaf ontology was the person class. The place class that would contain all the address detail of the person and this was created from a tutorial found on YouTube.

Chapter 5 Capture/Enumerate

Shown in the table 1 below is a list of concepts and terms. On the left-hand side of the table is a list of the concepts that need to be defined. On the right-hand side is a list of terms that have be agreed by the developer of the application. This is done so that all people involved have a clear interpretation of all the concepts that are needed for the ontology.

Concepts	Terms
event	Event
Dinner Party	DinnerParty
food	Food
desert	Desert
fowl	Fowl
Fowl with dark meat	DarkMeatFowl
Fowl with light meat	LightMeatFowl
fruit	Fruit
meat	Meat
Red meat	RedMeat
Non red meat	NonRedMeat
pasta	Pasta
seafood	Seafood
fish	Fish
Shell fish	ShellFish
social media content	MediaContent
post	Comment
picture	Picture

video	Video
user	Person
place	Place
city	City
country	Country
region	State
town	Town
restaurant	Restaurant
Chinese restaurant	Chinese
Indian restaurant	Indian
Steakhouse restaurant	SteakHouse

Table 1: Concepts and terms.

Chapter 6 Encoding

The standard being used to encode the concepts will be web ontology language (Owl). The ontology will be built using a graphical tool called Protégé and this application uses its own terms to encode the concepts. Below is a list of defined terms that are used in Protégé to encode concepts that will be defined.

Entities = Concepts

Classes = Frames

Object & Data properties = Slots

Axioms = Facets

Individuals = Instances

Chapter 7 Define Hierarchy and Classes

The first step in the section is to work out what concepts are going to be used. If you refer back to chapter 5 a highly informal ontology was built where all the terms well selected reviewed and agreed upon by the developer. The next step after the was to build a Simi-informal ontology as referred to in figure 7.1. You will notice all the concept and the relations between the concepts this is a very high-level view of the ontology.

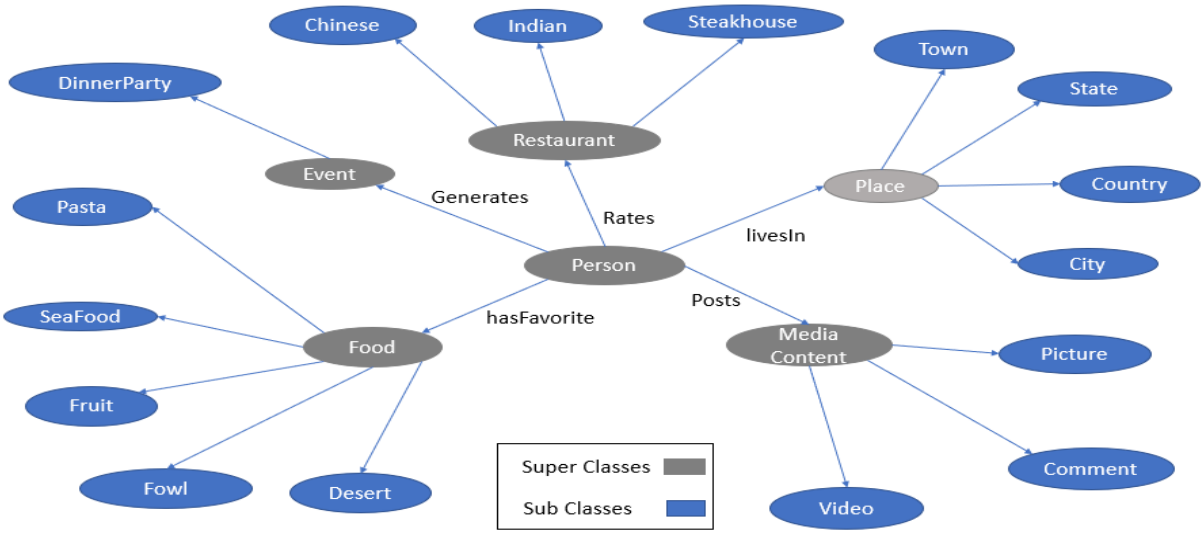

Figure 7.1: Simi-informal ontology.

By developing this high-level view like this it gives the developer of the ontology an overview of the classes and sub-classes and the relationships between concepts. This will simplify defining the concepts for the semi-formal stage which is done using Protégé. If you refer to figure 7.2 it shows the hierarchy of classes in Protégé you can see both the top down and bottom up approach as referred to is the class notes .

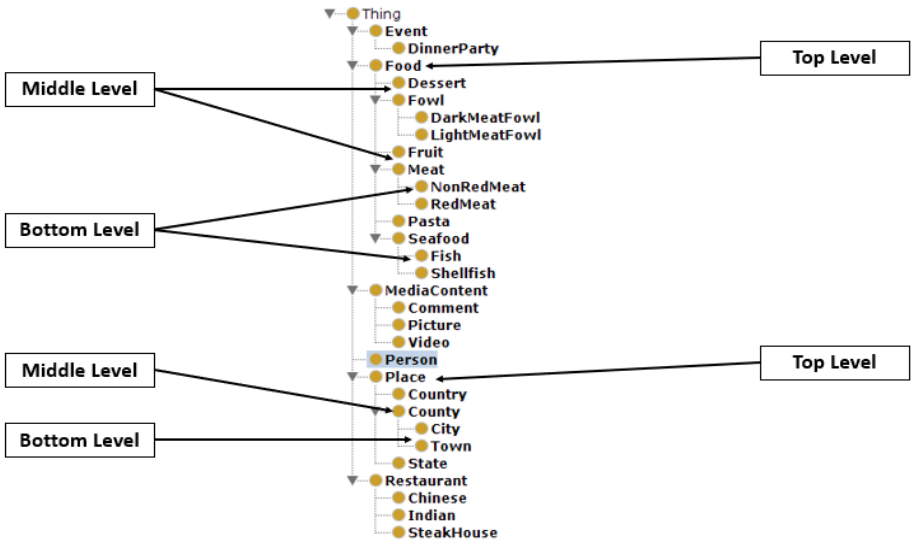

Figure 7.2: Hierarchy of Classes

As refer to in figure 7.2 and stated in the last paragraph there is two approaches that can be applied. The top down approach will start at the top down and start with the most general concept. The bottom up approach starts with the most specific term and work into more general terms (Stapelton, 2019). The two approaches are shown in figure 7.3.

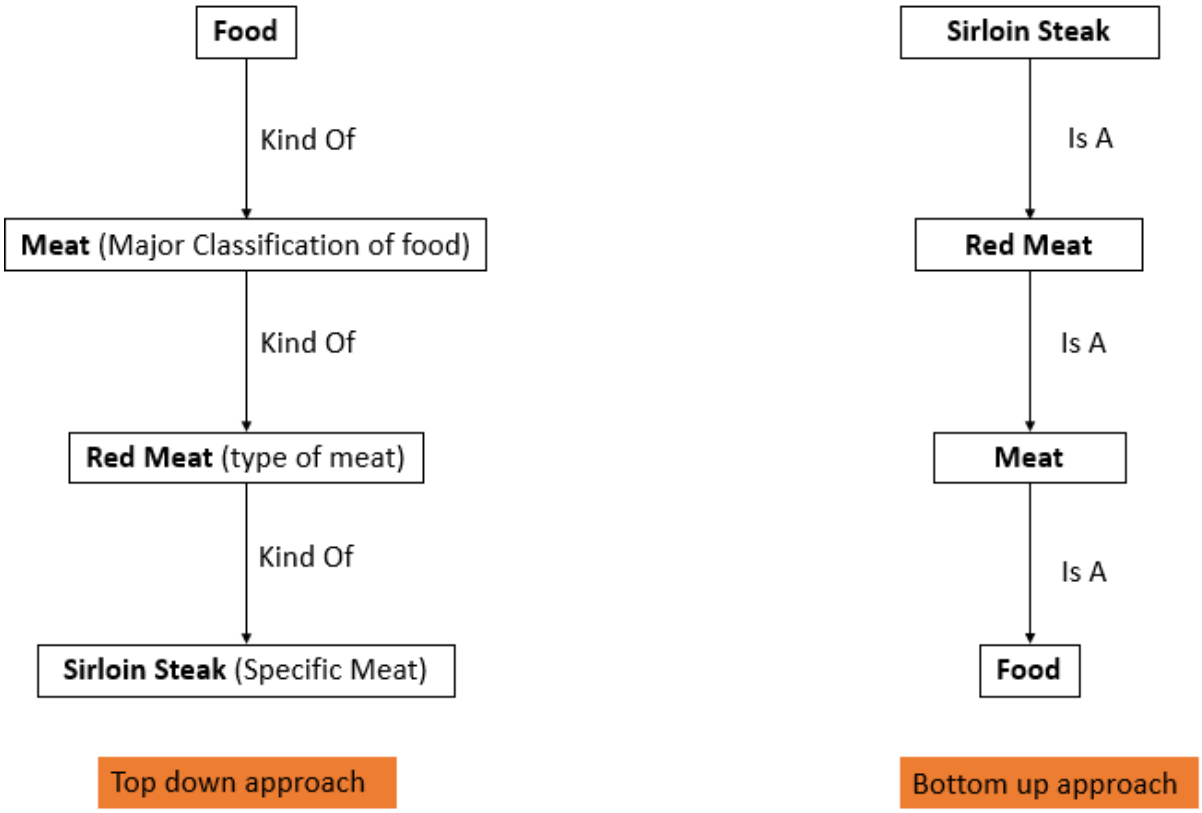

Figure 7.3: Hierarchal approaches to defining classes (Stapelton, 2019).

Now let's look at some formal definitions of classes. Shown in figure 7.4 these classes that are been generated in web ontology language (Owl). This form of Owl is based upon the mark-up language XML. In figure 7.4 you will notice two types of declarations highlighted by arrows. One declaration uses a URL and the other doesn't. The reason why they are different is the reuse of ontologies. As stated above there is three ontologies that are being reused foaf, place and food. What those URL's are doing is pointing to classes that have been defined already and published online this is common practice when building ontologies. As the saying goes is "no need to reinvent the wheel". The classes that are newly defined don't have a URL's. In figure 7.4 super classes are been declared. Shown in figure 7.4 is the declaration of sub classes.

```
    <Class IRI="#Town"/>
</Declaration>
<Declaration>
    <Class IRI="#Video"/>
</Declaration>
<Declaration>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#City"/>
</Declaration>
<Declaration>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Country"/>
</Declaration>
<Declaration>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Food"/>
</Declaration>
```

New classes

Predefined classes by other ontologies

Figure 7.4: Declaration of Classes

```
<SubClassOf>
    <Class IRI="#Video"/>
    <Class IRI="#MediaContent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#City"/>
    <Class IRI="#County"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Country"/>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Place"/>
</SubClassOf>
<SubClassOf>
```

Figure 7.5: Declaration of sub classes.

Chapter 8 Define Properties

Chapter 8 is where the object and data properties are defined. In Resource Description Framework (RDF) standard triples are defined as **subject>>predicate>>object.** In Owl the object property is similar to the predicate. It's the relationship between the classes. Referred to in figure 8.1 is the defined object and data properties. On the left side is the object properties. All object properties will be a subclass of topObjectProperty. On the right side is the data properties. All data properties will be a subclass of topDataProperty.

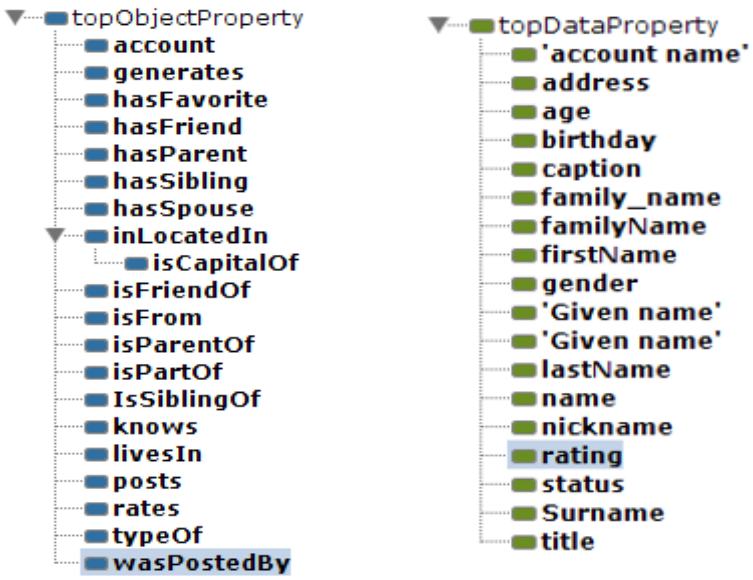

Figure 8.1: Object & Data properties.

Most of the properties shown in figure 8.1 have been defined by the other ontologies that are being reused. The new object properties defined in the ontology can be seen in figure 8.2. This is the formal declaration of the object properties. Because they are new object properties there is no URL associated with them.

```
</Declaration>
<Declaration>
    <ObjectProperty IRI="#IsSiblingOf"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#generates"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#hasSpouse"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#isFriendOf"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#isFrom"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#isParentOf"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#livesIn"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#posts"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#rates"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#wasPostedBy"/>
</Declaration>
```

Figure 8.2: New object properties.

Figure 8.2 is showing the object properties that were copied in from ontologies that have already been defined. The object properties such as hasFriend is defined and can be found at www.semanticweb.org highlighted by the red arrow. At the bottom you will notice an object property called holdsAccount and the is defined in the foaf ontology.

```
<Declaration>
    <ObjectProperty IRI="#wasPostedBy"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#hasFavorite"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#hasFriend"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#hasParent"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#hasSibling"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#inLocatedIn"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#isCapitalOf"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#isPartOf"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="http://xmlns.com/foaf/0.1/holdsAccount"/>
</Declaration>
```

Figure 8.3: Predefined object properties.

The last two figures focused on the formal declaration of object properties. Shown in figure 8.4 is the formal declaration of data properties. The two data properties referred to in figure 8.4 is caption and accountName. The first data property called caption was a new declaration for the ontology. The data property caption is used when a user uploads media content and want to leave a comment attached to the content. The second data property, accountName is defined in the foaf ontology and is being referenced by a URL.

```
<Declaration>
    <DataProperty IRI="#caption"/>
</Declaration>
<Declaration>
    <DataProperty IRI="http://xmlns.com/foaf/0.1/accountName"/>
</Declaration>
<Declaration>
```

Figure 8.4: Declaring data properties.

Chapter 9 Define Axioms

(Oxford Dictionaries, 2019) state that an axiom is a "statement or proposition which is regarded as being established, accepted, or self-evidently true". This quote is a good way to explain axioms when talking about ontologies. What axioms do is put constraints and rules on the properties and classes, this is done so that machines can interpret the terminology better. One of the rules that can be used is to say that a class is disjoint with another class. What this mean is plain English is that some concept can't be the same as other concept.

An example of this can be seen if figure 9.1. The statement say that a picture is disjoint with a video and a comment. This statement is true because a picture can't be a video, or a comment and a video can't be a picture. In figure 9.1 looks at new disjoint rules set up for the purpose of this ontology. Rereferred to in figure 9.2 is predefined disjoint classes found in the food ontology. The statement in figure 9.2 say that pasta can't be a seafood and vice versa. Humans know that pasta is not the same as seafood, but machine can't interpret the same as human and by using disjoint statements we are tell the machine to they can interpret the same as us.

```
<DisjointClasses>
    <Class IRI="#Comment"/>
    <Class IRI="#Picture"/>
    <Class IRI="#Video"/>
</DisjointClasses>
```

Figure 9.1: Disjoint classes defined for this ontology.

```
<DisjointClasses>
    <Class IRI="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#Pasta"/>
    <Class IRI="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#Seafood"/>
</DisjointClasses>
```

Figure 9.2: Predefined disjoint classes.

The next piece of code shown is figures 9.3 and 9.4 looks at rules for domains and ranges. The domain and range are rules that limit the scope of the statement. The Owl code below say that domain is county and that the range of county in country and the object property is locatedIn. The full statement would read a county is located in a country. That object property can only be associated with the domain and that range.

```
<ObjectPropertyDomain>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#inLocatedIn"/>
    <Class IRI="#County"/>
</ObjectPropertyDomain>
```

Figure 9.3: Object property domain.

```
<ObjectPropertyRange>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#inLocatedIn"/>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Country"/>
</ObjectPropertyRange>
```

Figure 9.4: Object property range.

Shown is figure 9.5 is a functional data property. This is a property that can only contain one value. (Protege, 2019). The Owl code below is a function data property stating that a birthday can only have one value. And that functional data property can is defined in the foaf ontology. This make sense as a person can't have to dates of birth.

```
<FunctionalDataProperty>
    <DataProperty IRI="http://xmlns.com/foaf/0.1/birthday"/>
</FunctionalDataProperty>
```

Figure 9.5: Functional data property.

Referred to in figure 9.6 is an inverse object property. The inverse property happens when you switch the order of the relationship. For this GRUB app there was a statement that said Eoin Dalton has a sibling Sorcha Dalton and the inverse of this is Eoin Dalton is a sibling of Sorcha Dalton. This is show below is figure 9.4. isSiblingOf is a new object property and hasSibling is a predefined term that has been reused for this ontology.

```
<InverseObjectProperties>
    <ObjectProperty IRI="#IsSiblingOf"/>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#hasSibling"/>
</InverseObjectProperties>
```

Figure 9.6: Inverse property.

(Protege, 2019) states that "annotation properties allow arbitrary information to be added to selected portions of an OWL ontology" and an "assertion is an axiom that declares something about one or more concrete objects in the class hierarchy". Shown is figure 9.7 is a mix of both references. What that statement is saying is the birthday is defined by the birthday found in the foaf ontology.

```
<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:isDefinedBy"/>
    <IRI>http://xmlns.com/foaf/0.1/birthday</IRI>
    <IRI>http://xmlns.com/foaf/0.1/</IRI>
</AnnotationAssertion>
```

Figure 9.7: Annotation assertion.

Chapter 10 Create Instances

As part of the brief for this project on of the aims was to create at least 18 individuals with 10 of those individuals being people linked together as this is focused around social media which is an application for linking people to people. In order to show the full ontology instances such as places, restaurants, types of food and media content also needs to be created as shown in figure 10.1. To show how the ontology is functioning it will focus around one individual (Eoin Dalton). This individual will have a spouse, four friends, three siblings and a parent. The individual will be able to post media content such a video, pictures and write comments. The individual in question lives in a town. The town is located in a county, the county is located in a state and the state is part of a country, but the individual is from a townland. The individual has a favourite food and can give ratings to restaurants they have eaten in. There is also some more detail added on some of the other instance. By doing this it enables us to be gaining knowledge on the individual but also shows how that individual links to other individual.

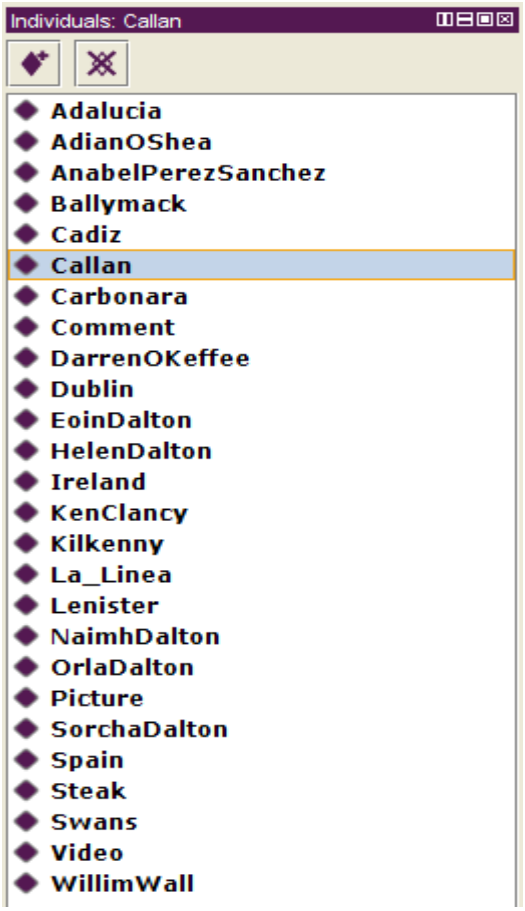

Figure 10.1:Instances

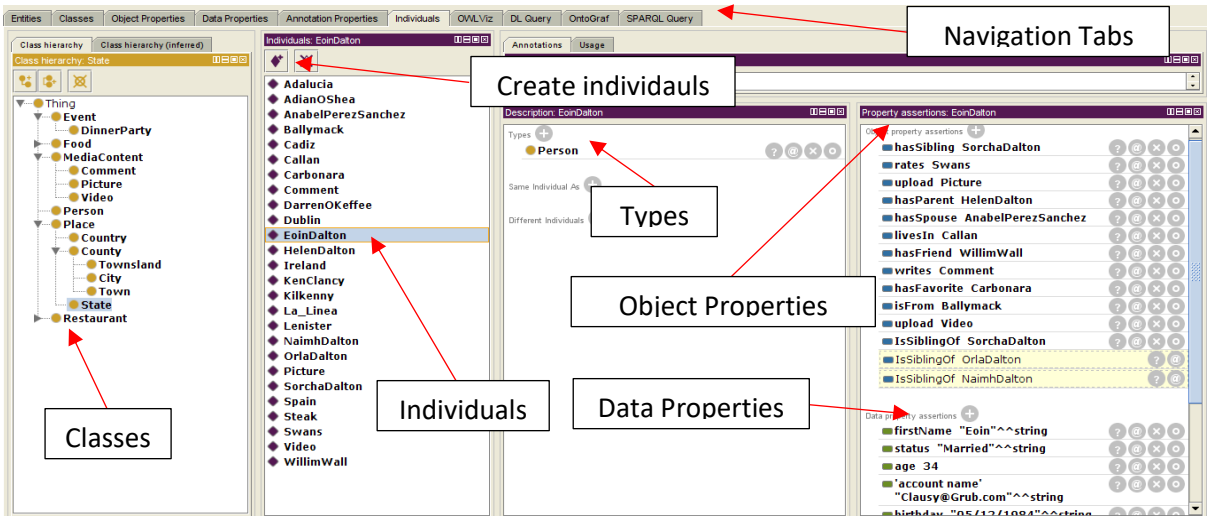

Figure 10.2: Creating individual in Protege.

Protégé is relatively straight forward for creating instance. Show in figure 10.2 is the user interface for Protégé. To the left side you will see all the classes. Across the top is the navigation tabs. In the middle is the individuals and where users will declare the individual type and to the right is where the properties are defined. The use case individual (Eoin Dalton) is selected as shown by the arrow in the individual section. Because this is the use case there is a lot of object and data properties assertations made shown in the right side of figure 10.2.

Referred to in figure 10.1 was the semi-formal implementation of individuals. Now the report will look at the formal implementation of some of the individuals. Shown in figure 10.2 is the initial declaration of individual. The individual (Eoin Dalton) that the ontology is demonstrated on is shown as well as a comment individual of type comment and carbonara individual of type pasta.

```
<Declaration>
    <NamedIndividual IRI="#Carbonara"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Comment"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#EoinDalton"/>
</Declaration>
```

Figure 10.3: Declaration of individuals.

When creating individuals in a semi-formal was as shown is figure 10.2 assertions were mentioned. Class, object and data property assertions can be declared. Show is figure 10.4 is a class assertion. In plain English it is saying that the individual Eoin Dalton is a type of person.

```
<ClassAssertion>
    <Class IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#Person"/>
    <NamedIndividual IRI="#EoinDalton"/>
</ClassAssertion>
```

Figure 10.4: Class assertation to show a type

There were two other assertions mentioned above and one of them was object assertion. Shown in figure 10.5 is an object assertion. What this assertion is saying is that individual Eoin Dalton has a spouse called Anabel Perez Sanchez. In figure 10. Below again is a object assertion but the different is the property isLocatedIn is all ready defined is the place ontology and it is using a URL to point to that property. What that assertion is saying that the county Cadiz is located in Andalucía and that Callan is located in Kilkenny.

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#hasSpouse"/>
    <NamedIndividual IRI="#AnabelPerezSanchez"/>
    <NamedIndividual IRI="#EoinDalton"/>
</ObjectPropertyAssertion>
```

Figure 10.5: object assertion.

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#inLocatedIn"/>
    <NamedIndividual IRI="#Cadiz"/>
    <NamedIndividual IRI="#Adalucia"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.semanticweb.org/eoin/ontologies/2019/2/untitled-ontology-74#inLocatedIn"/>
    <NamedIndividual IRI="#Callan"/>
    <NamedIndividual IRI="#Kilkenny"/>
</ObjectPropertyAssertion>
```

Figure 10.6: Defined object assertion.

The second assertion looks at data properties. Shown in figure 31 is a data property assertion. The is using the data property age from the foaf ontology and is assigning the data property to individual Eoin Dalton. In English this reads Eoin Dalton is aged 34.

```
<DataPropertyAssertion>
    <DataProperty IRI="http://xmlns.com/foaf/0.1/age"/>
    <NamedIndividual IRI="#EoinDalton"/>
    <Literal datatypeIRI="&xsd;integer">34</Literal>
</DataPropertyAssertion>
```

Figure 7: Data property assertions.

Chapter 11 Visualization of Ontology with OntoGraf

Protégé comes with many plugins for the creation of ontologies. One of the tools allows users to visualize the ontology. The hierarchy can be expanded to the lowest level. One of the problems with this is the graph can be quite hard to follow when all classes and individuals of classes are expanded shown below in figure 34. In figure 32 is OntoGraf show the super classes. All classes are sub-classes of Thing. In figure 33 OntoGraf is showing all super classes and sub-classes and underneath figure 33 is figure 34 the shows all the individuals and classes. If you notice individual Eoin Dalton has many arrows pointing to him and other individuals, this is because must of the relationships and knowledge is focused around this individual.

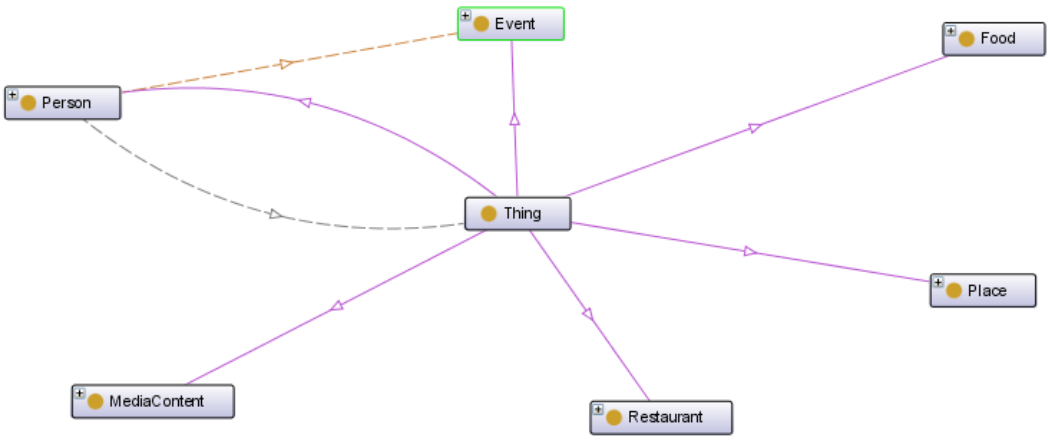

Figure 8: OntoGraf showing super classes.

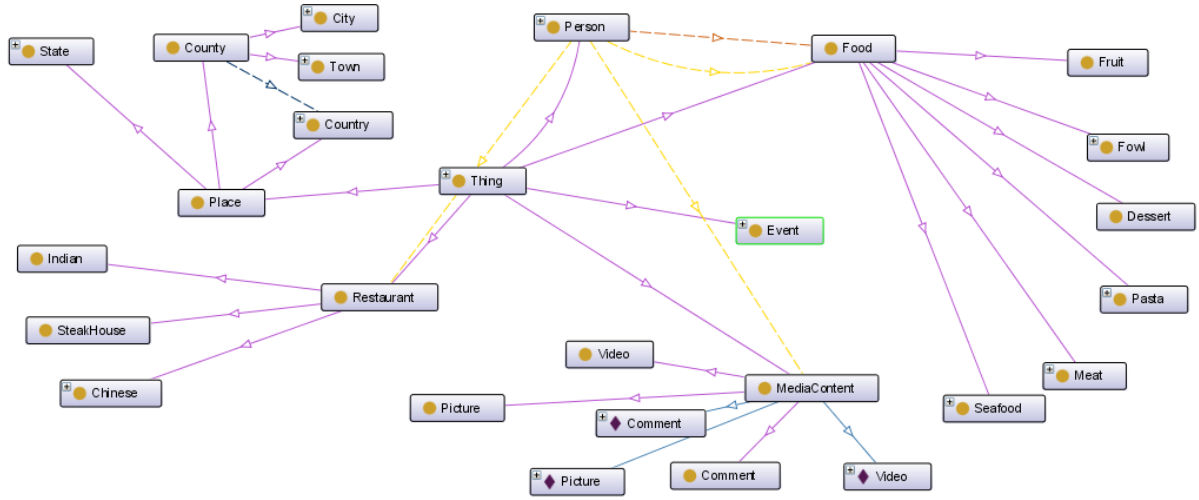

Figure 9: OntoGraf super classes and sub-classes.

In figure 34 working from left to rights is the hierarchy of the classes with all individual expanded on the right side. The classes and individual can be moved around to make the ontology more readable to the users.

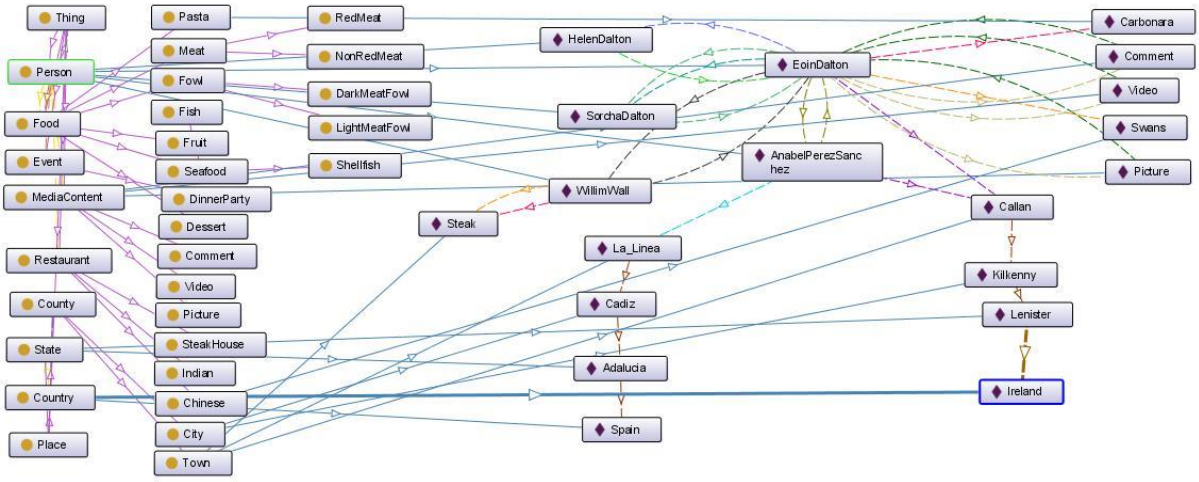

Figure 10: OntoGraf super classes, sub-classes and individuals.

Chapter 12 Conclusion

For this project the class was giving a small ontology model and asked to extent and adapt the ontology to suit a from of social network application. For this particular project it adapted an idea from another 4th year student who is currently in the process of building at food social media application. In order to build the ontology first the different concepts that were needed were taught out. Then the terms were defined so that they matched the application the William was building. After that there was an investigation into other ontologies that have been previously built and could be reused for the purpose of this project. The last step involved building the ontology through Protégé. To test the ontology there were many individuals created and the reasoner was used to see if all the axioms made sense. This project was a good introduction into the use of ontologies.

Chapter 13 Bibliography

Bikakis, A., 2019. *Protege: How To Create Classes And Properties.* [Online]
Available at: https://www.youtube.com/watch?v=MbauHV2-XYw
[Accessed 11 03 2019].

chistyakov, Max, K. & Lapaev, M., 2015. *Github.* [Online]
Available at: https://moodle.wit.ie/mod/folder/view.php?id=2794194
[Accessed 11 03 2019].

FOAF, 2014. *FOAF Vocabulary Specification 0.99.* [Online]
Available at: xmlns.com/foaf/spec/index.rdf
[Accessed 11 03 2019].

Tserpes, K., Papadakis, G. A., Papaoikonomou, A. & Kardara, M., 2012. *An Ontology for Social Networking Sites Interoperability.* s.l., ResearchGate.

Chapter 14 Appendix

14.1 Reflection

Now that it's over, what are my first thoughts about this overall project? Are they mostly positive or negative?

I think this project can only be seen as positive as the concepts are will be very useful in the near future. One of the things I would say is that I have a little advantage over my class mates as my final year project is based on web ontology language and the semantic web. That made this project all the more interesting as I got to great experience using Protégé which will benefit me with my 4th year project.

Describe two things you feel you learned from this activity in general?

1. The first thing I learned was how the reuse and import ontologies into your own ontology. I understood the concept of reusing ontologies but up until this point I was unsure on how to incorporate them into my own ontology. I learned that it can be done in two main ways. The first is to merge an already defined ontology into your own or a newly created ontology. The

second way is to copy the parts of the ontology that you want to use into your own ontology which was the case for my project. This will be a major help to me in the next few weeks when I am putting the final touches on my final year project prototype ontology.

2. The next thing I learned was about assertions and how important they are when defining an ontology. When you think about how we would use the word in an English sentence it makes perfect sense to how they are used for ontologies. To make an assertion about something is like stating a fact. And with ontologies this is how we can gain knowledge. An example is Eoin Dalton has a friend, this is a fact about me because I do have friends.

What were some of my most challenging moments and what made them so?

For me the must challenging aspect is working out the difference between what is a class and what is an object property. For instance, I created a class for person. I was then going to create sub-classes for friend brother, sister mother and father, but I figured out that through the object properties we can define relationships then once we create an individual, we can define what that person is. As an example, Eoin Dalton has sibling Sorcha Dalton. Through the object property has sibling we can assert that Sorcha is my brother, so no need to create a class for this. This for me is the most difficult thing about ontologies.

What were some of my most powerful learning moments and what made them so?

I will be honest here and admit that these moments haven't happened during this project but through your lectures. When we had conversations back before Christmas, I couldn't get a grasp on the power of ontologies and the semantic web. But once we started Knowledge systems engineering it all started to come together. Even now there are a lot of parts I am learning. Coming from a programming background I was only seeing how it could be used as a schema. But now I am seeing the interoperability of ontologies and how the knowledge can be extracted from them. Also, with programming's languages such a Java and JavaScript users may finally be able to manipulate the ontology is such a way that graphical interfaces may be built to add individuals so the ontologies can grow ontologies into knowledge's bases for people to search accurately with the use of the semantic web.

What is the most important thing I learned personally?

The most important thing I learned was navigating Protégé. As I am in the progress of building my 4th year project on this tool, I feel this project in invaluable because it gave me time to actually sit down and use the tool in detail. Up until this point I was getting to spend an hour here and there using it as I am still researching for my 4th year project. I am in trying to get all the CA done to give me two or three weeks at the end of the semester to really focus that project and after doing KSE it gave valuable experience using the tool for my own project.

How will I use what I've learned in the future?

One of the great things about ontologies and the semantic web is that it is only in recent years the computing industry have adapted to using the technologies. This means there will be a need for people with knowledge about these set of standards and technologies. That give us a good advantage when applying for job out in industry. During an interview being able to speak about relatively new technologies can only be a benefit.