THE MANCHESTER SUSHI FINDER

Progress Report
COMP60990 Research Methods and Professional Skills
(June 6, 2014)

Name: Al Abbas, Hani Ahmed
Project Supervisor: Sean Bechhofer

School of Computer Science
The University of Manchester

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT:

1 INTRODUCTION:

As the trend nowadays to try making machines more intelligent, sharing knowledge of information instead of sharing the raw information in the web is becoming more desirable. Ontologies are considered the main pillar of the semantic web [1]. Semantic web is all about sharing knowledge ,that is understandable for machines, on the web [1]. Creating ontologies is a way to capture our knowledge of the world [1].

Computers do not understand information stored on the web as xml and html. They are just codes to the machine and they display it to users regardless of what knowledge needed. So, ontologies came along for machines to make sense of information. Ontologies are a way to represent knowledge and make inferences from that knowledge using machines computational capabilities and some description logic.

An intelligent way of representing knowledge needs an intelligent way of browsing it. There are a lot of intelligent browsers that is ontology driven user interfaces. Browsing and constructing queries through such user interfaces would be easy and save time. This is because that the user interface guides the user to construct the wanted query with no worries about previous knowledge of the domain. Ideas like manuals and the help in the menu bar of a user interface would same absolute comparing to the self-guided user interface.

Additional technique to make the user interface smart is to use faceted browsing. The idea behind faceted browsing is to personalize the search and get more specific results by suggesting some filters. Faceted browsing is very related to ontology driven interface since both provide some information about the query while been constructed [2].

There are systems, that are ontology driven user interface, exist such as Transparent Access to Multiple Bioinformatics Information Sources (TAMBIS) and SEmantic Webs and AgentS in Integrated Economies (SEWASIE). TAMBIS is system that gather and analysis bioinformatics information from different sources through one

interactive user interface [3]. While SEWASIE meant to access multiple sources of data and help user through constructing exact query needed [4].

The idea of ontology driven interface is not new. In this project, will try to build an application that has some functionalities of interactive interface.

1.1 Project Aims And Objectives:

The aim of this project is to investigate and demonstrate the benefits of using OWL ontologies within ontology driven interface [5]. This will be shown be implementing a configurable and flexible application, so that most configuration will lay in the ontology file itself and could browse other ontologies that contain some specific configuration [5].

To achieve this aim, project was divided into number of objectives. The objectives of this project are:

1. Research ontology driven interfaces.
2. Research faceted browsing.
3. Research OWL API.
4. Look at existing applications that are similar in the basic functionalities.
5. Gather project requirements.
6. Develop a project plan.
7. Design user stories and tests evaluations.
8. Determine the tools that will be used in the project.
9. Start implementing and divide and add functionalities one at a time.

1.2 Project Scope:

This project is going to demonstrate the use of ontologies using an interactive user interface. The functionalities that will be included in the interface are:

1. The user interface will be built using java.
2. User interface will be configurable for generic use.
3. Flexible query building.
4. Preferences of the user will be saved as configuration for reusability.
5. The sushi finder will be a desktop application.

6. The sushi finder should work for any ontology with standard annotations defined in them (Configuration will be stored in the ontology).
7. User can load ontologies one at a time.
8. User can decides which thing to query about (Sushi-Sushi dishes).
9. User can query for specific sushi or sushi dish type based on wanted or unwanted ingredients.

1.3 Report Structure:

2 BACKGROUND:

2.1 OWL API:

Application Programming Interface (API) is a set of protocols that make sure the software components interact with each other in the right way [6]. API could take many forms in different areas [6]. It is used in the web as a set of Hypertext Transfer Protocol (HTTP) [6]. Also, it has heavy use as libraries of programming language [6]. API is used in different forms such as libraries of programming languages. For example, Java APIs[6]. In object-oriented languages like java, the API is a set of classes and methods to be accessed and used [6]. Basic examples would be like using the inputting and outputting classes e.g. (BufferedReader and BufferedWriter classes in java) [6]. Since this project will be built using java-programming language, the API used is a java API which is called OWL API. OWL API is a set of classes and methods that facilitate the access to Web Ontology Language (OWL) ontologies. It creates objects that represent ontologies objects and manages the interactivity between them and any other program. OWL API consists of two terms, so farther clarification is needed. The following two subsections talk about OWL and OWL API separately.

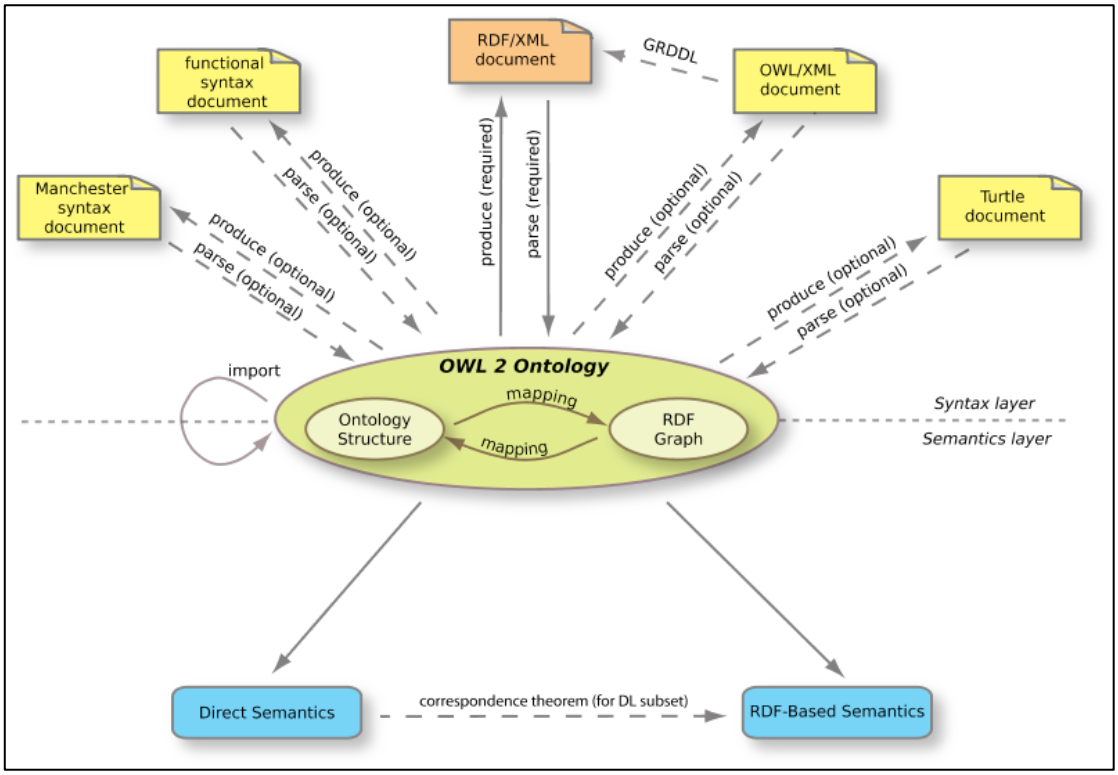
2.1.1 OWL

OWL is a semantic web language that represents things about the world, group of things, and the relations between them [7]. It is a way to represent knowledge such that it is a representation of the world and our knowledge of it and it is accessible to programs and can be used [8]. It is able to represent explicit and implicit things [7]. The intention behind creating OWL is to be used not only by human but also by applications [9]. OWL can be access from machines because it is based on computational logic so that the machine using some software can reason over them [7]. There are two versions of OWL: OWL and OWL 2 [7]. OWL is a W3C recommendation since 2004, and then OWL 2 was published in 2009, followed with a second edition in 2012 [7, 10]. OWL 2 is just an extension and revision of the original OWL publish in 2004 [7]. OWL has several defined syntaxes including Functional Syntax, RDF/XML, OWL/XML and the Manchester OWL Syntax [7, 10].

There is a lot of information scattered on the web [9]. This information could mean something for humans but not for machines [9]. So, the semantic web gives explicit meaning for this information [9]. As a result, integrating and processing the

information would be easier for machines [9]. There are three sublanguages of OWL, each one suits some needs [9]. They are just subroutine of the full OWL [9]. The three languages are: OWL Lite, OWL DL, and OWL Full[9].

In 2012, OWL 2 has been introduced [11]. It is not different than OWL, it could be seen as an extension of OWL with some additional features [11]. OWL 2 have several syntaxes and semantics, usually a developer needs only one syntax and one semantic [11]. Figure 1 shows the structure of OWL 2 [11].



OWL files or documents are called ontologies [7]. The purpose of these ontologies is to make it easier of machine to access information in the web and preform reasoning on them. These ontologies can be put into the web or into a local computer depending on the need. One of the advantages of ontologies in the web is that they can be referenced from or reference to other ontologies [7]. Ontologies station in a local computer can be used locally in the same level of the local machine.

2.1.2 OWL API

OWL API is an Application Programming Interface for the purpose of specifying how to interact with OWL Ontologies [10]. OWL ontologies can be created, manipulated,

and reasoned over using OWL API [10]. It has been available since almost the same time of OWL [10]. OWL API went through several revisions following the development of OWL [10]. OWL API has the ability to parse and serialize OWL ontologies to different syntaxes such as Functional Syntax, RDF/XML, OWL/XML and the Manchester OWL Syntax [10].

OWL API comes with free java implementation that takes out the burden of parsing and serializing OWL ontologies from the developers back [10]. OWL API comes also with loading and saving ontologies capabilities [10].

OWL ontologies being accessed using OWL API only through OntologyManager interface [10]. OntologyManager interface manage all changes in ontology as seen in Figure 1 below [10].

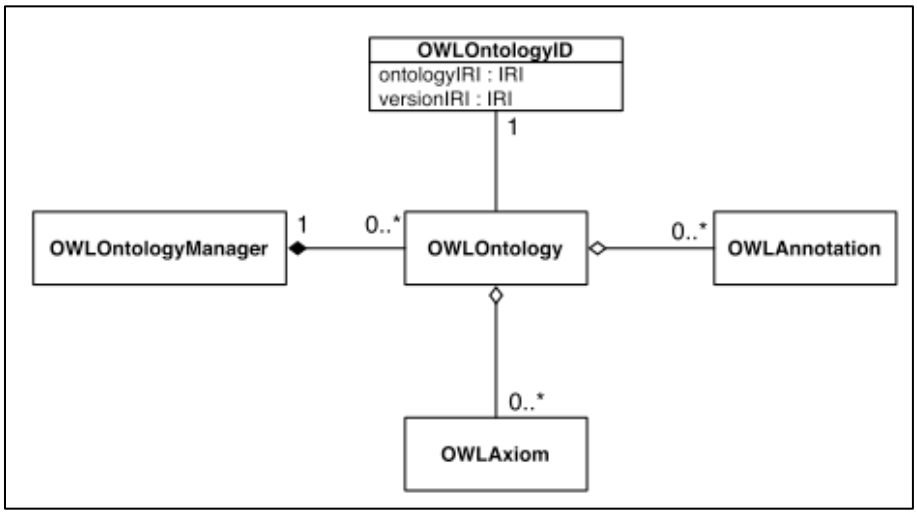


Figure 1: UML diagram showing the management of ontologies using OWL API

Inference is applied on the OWL ontologies using OWLReasoner interface [10]. This interface provides some useful check like consistency, checking computation of class and axiom entailments [10]. Since the reasoning functionality is separate, developers either can use the available or can provide their own implementation [10]. There are some exist implementations of reasoners such as FaCT++, HermiT, and Pellet [10].

As for Query using OWL API, it does not offer any query mechanism [10]. However, it provides some sort of basic querying which is based on entailment checking functionality [10].

2.2 Information Retrieval

Information retrieval is the method in which some information is retrieved from a source or multiple sources contains needed data. Every retrieval system needs some mechanisms to retrieval relevant needed information. Nowadays, there is more information probably resides on the cloud than it was in the recent years. As the amount of data on the web grows dramatically, the need to find and retrieve relevant information becomes more important. Getting the wanted results is becoming more problematic because of the amount of the data on the web and the technique used. Usually, in a search engine the results range between relevant and irrelevant. It would be nice for a user to query for something and get the most relevant result that meets his/her needs.

There are different methods of retrieving data from the web. Keyword-based search method could be the oldest among the others.

2.2.1 Keyword-Based Information Retrieval

Keyword-based search method use specific words call "Keyword" that are linked with database records [12]. Keyword-based search method would be the default and the usual choice to use in search, since it has been used over long time. This method seems easy to use, as it is resemble natural language which could be understandable by humans but not by machines [12]. Because of the human factor that exists in writing the search query, things inevitably could go wrong. Simplicity and ease come with a cost, keyword-based search method suffers from some serious issues [12]. (1) One of these issues is the lack of accuracy and recall because of all of the synonyms and the homonyms which are based on memorizing terms rather than concepts [12]. (2) Another major issue is that using keyword-based search add more ambiguity, when the user want just to browse around to find out what is there or the user does not know the right term used in specific content [12].

According to [12], there are solutions for both issues. The lack of precision and recall can be treated by ontology-based information retrieval method [12]. The growth in the ambiguity issue, would be solve be using multi-faceted search method which would

guide the user during constructing the search query [12]. So, the use of knowledge base and concept base would be more desirable than just providing arbitrary information. In addition, a sense of Artificial Intelligence is also felt since machines can make inferences based on some rules.

In this project, keyword based method won't be used as I am trying to show out the benefits of using ontology driven user interface.

2.3 Ontology Based User Interface

In general, developing user interfaces hindered by the knowledge of the user [13]. Letting the user known what can he ask for and constructing a meaningful search query using the user interface is the major issue [13]. To remedy this issue, solutions have been proposed. Some of those solutions would be making every option in the user interface available to reduce the recall issue. Another solution would be writing manuals to the user to follow. These solutions might be providing more complexity and other problems. The former solution could overwhelm the user with all of the options available whether needed options or not. The latter solution could increase the load on the user to study and spend time on something that needed to be recalled eventually. Ontology driven user interface would be the most suited solution, since ontologies are based on conceptual model rather than just terms [4, 13]. This conceptual model gives a map for the user to follow upon constructing queries [13].

Ontology based user interface is a user interface that allows the user to construct and manipulate queries based on some domain concept stored in ontology [4, 13]. The domain concept drives the user interface [13]. There is no need for manuals or shove all available options in the user interface, since the ontology based one which should act as a guide for the user [4, 13]. It depends on recognizing knowledge instead of memorizing keywords [13]. It allows the user to build a complex and meaningful queries and return the needed results [13]. In addition, it offers the user the option of browsing around to find out what he/she needs [13]. The user does not have to any thing about the underlying conceptual knowledge [13]. TAMBS give the illusion of retrieving from single source while it read from multiple sources and convert selected options to appropriate query languages that match sources' [13].

The user interface offers choices and some scenarios for the user, so that the user would be guided toward constructing meaningful queries that return the intended results [4, 13]. Query expressions are Description Logics (DLs) expressions and they are incremental and compositional [4, 13]. Users would not face the no-result status after running queries. DL is a way for knowledge representation used by the conceptual model [14]. DL model is not easy because of the need knowledge about the DL syntax along with understanding it, so a friendly user interface need to build to separate the user from dealing with DL [13]. DL provide hierarchal model based on conceptual model that represent classes of specific domain and the relationships between the instances of those classes [4, 13].

There are two kind of concepts that the DL model support [13]. The concepts definitions and the assertions made on the concepts definitions, like the subsumption relationship between two classes [13]. In a way, assertions on the original concepts considered as defining new concepts definitions [13]. Compositional concept can be formed using some services provided by DL [13]. Reasoning about the concept definitions is done through the services provided by DL [13]. These services are [13]:

- **Satisfiability**: make sure that the concepts are consistent.
- **Subsumption**: create composite concepts definition from assertion made on the original concepts definition.
- **Classification**: make new classification hierarchy based on the subsumption relationship.
- **Retrieval**: retrieve any individual that is part of concept definition whether it is original or generated from the subsumption relationships.

Ontologies support creating annotation properties within themselves and associate them with entities. Annotations properties consist of the name and the value. Annotations within ontologies play a major role in driving the user interface. Annotations would form some set of rules for the user interface to follow and interact based on. A tool was developed based on some Animals ontology and annotations would be a good example of what could annotations do in term of user interface interactivity [15].

In this project, I am trying to develop an application to find out different information about sushi. The application is ontology based. Annotations play major role in driving the user interface to make more flexible. This application is based on The Manchester Pizza Finder application except this one would have some enhancements.

2.4 Faceted-Based Search

Ontology based user interface only provides the taxonomy and conceptual hierarchy and broad search capabilities. With the ontology conceptual hierarchy, user still can get broad search results. Transition from general to more specific results needs some kind of smart retrieval mechanism. Facet-based search along with ontology based user interface would guide the user toward constructing valid search queries and personalizing the search queries to suite the user needs. As using ontology in user interface development eliminates the recall element, using faceted-based search eliminates the ambiguity constructing the query and gets the intended results. So, ontology helps in returning relevant results. But faceted-based search assists in taking those relevant results and returning the most exact results. Ontogator is a system that combines the two methods: ontology driven interface and faceted based search [12]. The intent of Ontogator to search of particular image with specific annotations [12].

Ontology support faceted classification system as it provides a taxonomic order. Taxonomic order allows multiple way of viewing results rather that pre-determined one [12, 16]. Faceted search based on faceted classification system where information element are dimensions called facets [16]. Faceted search is an intelligent and efficient retrieval mechanism that allows the users to filter a collections of information based on some facets [16, 17]. Faceted search is known also as “faceted navigation” or “faceted navigation” [16]. Users can get more accurate and relevant results by applying filters (Facets) [16]. Facets here are derived from the ontology itself based on some annotations as metadata [12].

There are faceted browsing and faceted search. Faceted browsing is constructing search queries by selecting some provided filters (Facets) [12]. In faceted browsing, user is provided with choices to select from to form the valid search query [12]. The query language is hidden from the user, so the burden of knowing the syntax is lifted.

On other hand, faceted search is more in personalizing the search result to suit what is needed. Faceted search is used heavily in e-commerce websites like Amazon[1] or eBay[2]. Figure 2 shows the use of faceted search in Amazon.

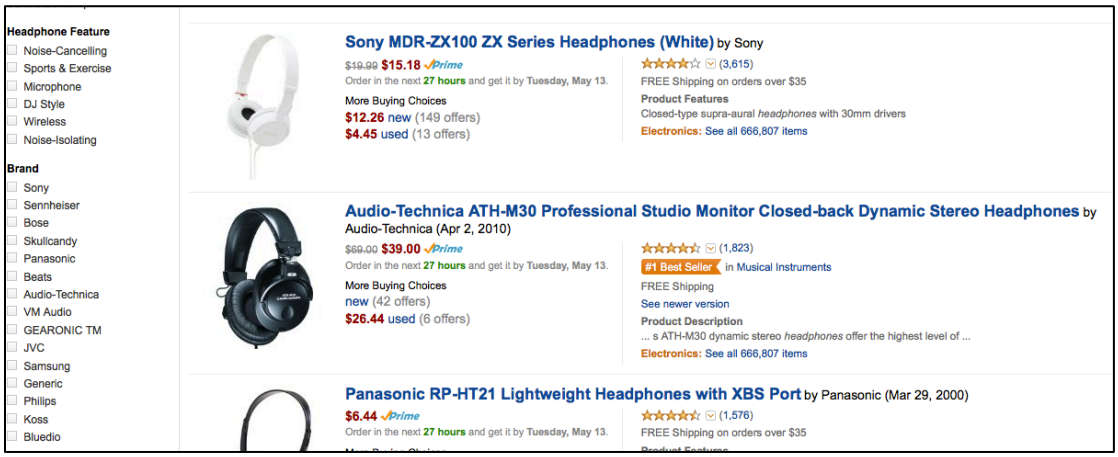


Figure 2: The use of faceted search in e-commerce website (Amazon)

Faceted search can be applied in two ways, either unidirectional or bidirectional. In unidirectional way, either applies it beforehand on a collection of selection that is browsed to construct the query or on the result of a query so it can be refined more. In the bidirectional way, it is to apply it both beforehand and afterward. Both serve the same purpose which is to personalized the search and make it easy to suit the user's needs.

Facets represent categories selected [12]. Each category consists of subcategories [12]. For example, in the pizza finder application user may choose Spicy Ingredient to get all pizza that are spicy but the category "Spicy Ingredient" could have subcategories like "Hot Pepper". Query in DL format would be [12]:

$$(S_{1,1} \vee \ldots \vee S_{1,k}) \wedge (S_{2,1} \vee \ldots \vee S_{2,k}) \wedge \ldots \wedge (S_{n,1} \vee \ldots \vee S_{n,k})$$

Where $(S_{1,1} \vee \ldots \vee S_{1,k})$ is the whole facet and S is the subcategory within that facet [12].

There are some benefits of using facets [12]:

- **Guidance**: facets guide the user toward constructing valid search queries.

[1] http://www.amazon.com

[2] http://www.ebay.com

- **Transparency**: facets give the user idea of what is available and help in browsing the content.
- **Lucidity**: facets help in removing ambiguity caused by synonymous and homonymous query terms.
- **Relevance**: facets help with pre-compute partial results on selecting choices.

2.5 Ontology Visual Querying

The idea of visual querying is constructing a search query visually using drag and drop instead of the traditional way. Same idea can be applied on ontology-based interfaces that would be more powerful because of the benefits of the ontology-based applications. The interface would guide the user to build interactive meaningful queries by using ontologies [2, 4]. In addition, another advantage derived from the benefits of ontology-based applications is constructing only exact queries [2, 4].

According to [2], visual querying is not new. It has been there since almost the beginning of textual query languages. Almost all visual querying languages have two features in common. The two features are: (1) a model to represent the stated query and (2) a way to of constructing the query. Since visual querying languages invented to query from a data structure, it is only natural for its evolution to follow the development of data structure [2, 18]. A simple example of visual querying would in Microsoft Access.

A major benefit from ontology visual querying is the ease of querying, since user only drag and drop what needed to be queried. User does not have to remember or know the vocabulary, since user can survey the domain [2, 4]. As a result, forming queries for naïve users becomes easier [2]. In ontologies, new concept can be defined either directly like defining class or indirectly like making inference of something. Therefore, creating query is the same as creating new concept such as the TAMBIS system and SEWASIE system [2-4]. Another advantage would be helping users, who not experienced with the system, to create satisfiable queries according to the constraints [2].

2.6 The Manchester Pizza Finder

The Manchester Pizza Finder is an application that finds specific pizza based on some topping choices. User can include and exclude any toppings, and based that the result would satisfy the query. The use of DL reasoner is present in this application, since it generated the filtering criteria (pizza topping) and their categories in the runtime. It is also make sure that the constructed queries and results are consistent. Based on the choices made the DL reasoner retrieve result that fulfill the input query. This application shows the use of ontologies, OWL API, and the power of building ontology-based interface, and faceted browsing.

The Manchester Pizza Finder is a user interface application that makes use of OWL ontology. It uses pre-defined pizza ontology that represents a domain concept of a pizza restaurant menu. For the application to be able to communicate with the pizza ontology, an API needs to be used. OWL API is an important component as any part of the application if not more important. OWL API manages all the communications between the application and the pizza ontology. OWL API is implemented using Java Pizza finder is developed using Java, since OWL API is a java API. This makes the communication between the application and the ontology easier. OWL API have full access to the pizza ontology, it can preform operations on the ontology like make sure it consistent.

Pizza finder is considered ontology-based application, since it is the ontology that derives the interface and provides a conceptual hierarchy of a pizza domain. In ontology-based application, user does not need to recall keyword or know query language on querying for specific pizza. The application itself guides the user toward building only valid queries with the ability of making complex meaningful ones. It has the ability to incrementally compose queries. User can browse around to figure out what specific toppings are needed. It based on the knowledge of pizza domain, not on keywords.

Pizza finder personalizes the query construction process by providing some filters (Facets). As a result, the results would suite the user needs. Pizza finder uses faceted-base in query building, so the user will be guided to construct only valid queries. User does not have to recall what keyword to search for something in the domain. User has

the option in querying for broad or specific pizza in the domain based on chosen facet. Figure 3 shows the use of facets in pizza finder, use can query for example for vegetable topping pizzas or can query for more specific thing in vegetable topping category such as Tomato topping.

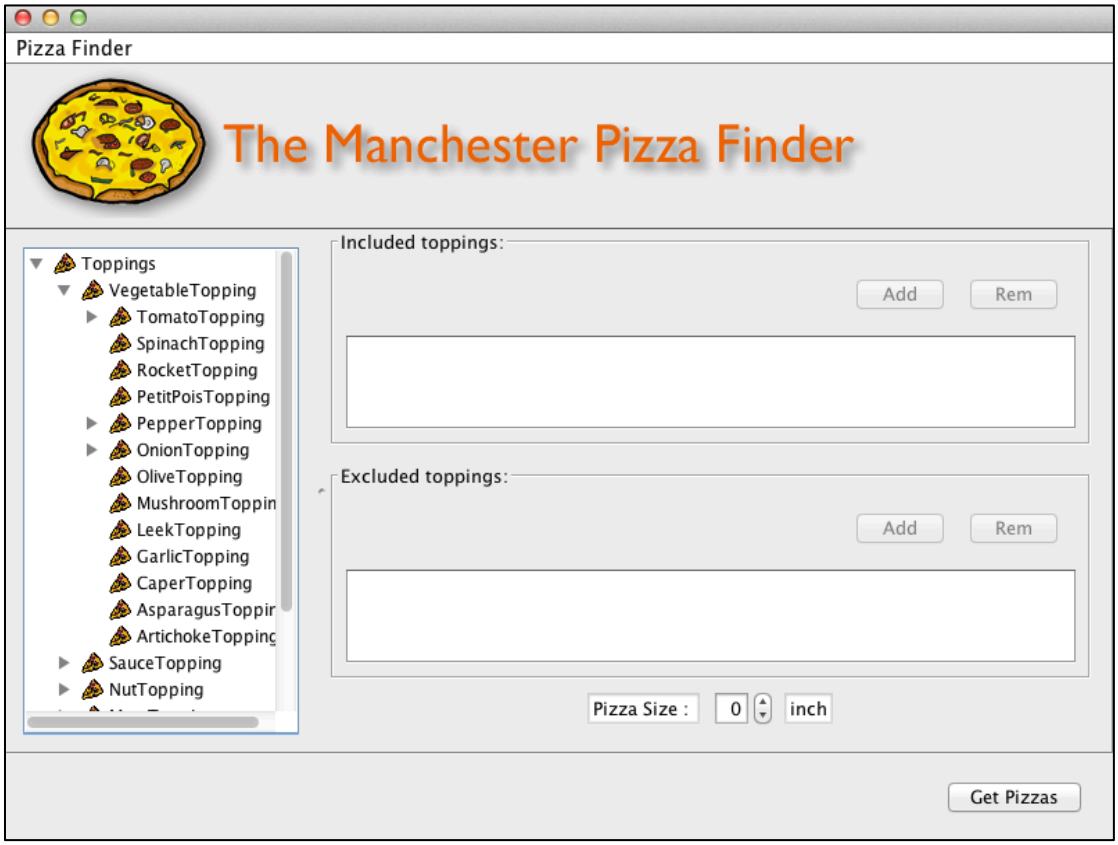


Figure 3: Facets are in the left hand side used to specify what is needed exactly.

Sushi finder is considered an extension for pizza finder. Some enhancements of pizza finder are to be introduced. In Sushi finder, the application is more flexible than pizza one. The application should be able to work for any given ontology regardless of domain, but should follow some standard annotations. Another enrichment would be the use of annotations to drive the user interface sort of dynamically. Keeping the configurations in the ontology itself make it easier to for the user interface to be flexible. The whole application would be configurable in term of labels and languages being used. It configurable within itself, no need for changing configuration files. Facet search is introduced, user can filter the choosing options based on some configuration done in the ontology as annotations. In addition, user can apply filters on the query results.

3 RESEARCH METHODS:

3.1 Project Plan:

3.2 Project Deliverables:

3.3 Project Evaluation Plan:

3.4 Project Tools:

4 PROGRESS:

4.1 Users Stories:

4.2 Acceptance Tests:

4.3 Prototype:

5 REFERENCES:

1. Ding, L., et al., *Using Ontologies in the Semantic Web: A Survey*, in *Ontologies*. 2007, Springer US. p. 79-113.
2. Bechhofer, S. and N.W. Paton, *Ontology Visual Querying*, in *Encyclopedia of Database Systems*. 2009, Springer.
3. Stevens, R., et al., *TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources.* IBM System Journal. **40**(2): p. 532-551.
4. Catarci, T., et al., *An Ontology Based Visual Tool for Query Formulation Support.* ECAI, 2004: p. 308-312.
5. Bechhofer, S. *The Manchester Sushi Finder - Project Page*. 2014 [cited 2014 March 5, 2014]; Available from: http://studentnet.cs.manchester.ac.uk/pgt/2013/COMP60990/project/projectbookdetails.php?projectid=20889.
6. Wikipedia. *Application programming interface*. [cited 2014 April 21, 2014]; Available from: http://en.wikipedia.org/wiki/Application_programming_interface.
7. Group, W.C.O.W. *Web Ontology Language (OWL)*. 2012 [cited 2014 April 19, 2014]; Available from: http://www.w3.org/2001/sw/wiki/OWL.
8. Davis, R., H. Shrobe, and P. Szolovits, *What is a Knowledge Representation?*, in *AI Magazine*. 1993. p. 17-33.
9. Group, O.W. *OWL Web Ontology Language Overview*. 2004 [cited 2014 May 13, 2014]; Available from: http://www.w3.org/TR/owl-features.
10. M, H. and B. S, *The OWL API: A Java API for OWL ontologies.* Semantic Web, 2011. **2**(Number 1 / 2011): p. 11-21.
11. Group, O.W. *OWL 2 Web Ontology Language Document Overview*. 2012 [cited 2014 May 13, 2014]; Available from: http://www.w3.org/TR/owl2-overview/.
12. Hyvönen, E., S. Saarela, and K. Viljanen, *Application of ontology techniques to view-based semantic search and browsing*, in *The Semantic Web: Research and Applications*. 2004, Springer. p. 92-106.
13. Bechhofer, S., et al., *Guiding the User: An Ontology Driven Interface.*
14. Bechhofer, S. and C. Goble, *Classification Based Navigation and Retrieval for Picture Archives*, in *Database Semantics*. 1999, Springer. p. 291-310.
15. Schreiber, A.T.G., et al., *Ontology-based photo annotation.* IEEE Intelligent Systems, 2001. **16**(3): p. 66-74.
16. Wikipedia. *Faceted search*. [cited 2014 April 23, 2014]; Available from: http://en.wikipedia.org/wiki/Faceted_search.
17. Smith, D.A. and N.R. Shadbolt, *FacetOntology: Expressive Descriptions of Facets in the Semantic Web*, in *Semantic Technology*. 2013, Springer. p. 223-238.
18. Catarci, T., et al., *Visual query systems for databases: A survey.* Journal of Visual Languages and Computing, 1997. **8**: p. 215-260.