Malaysian Food Composition Ontology Evaluation

Article · October 2019

DOI: 10.18178/ijmlc.2019.9.5.861

2 authors:

Norlia Yusof
International Islamic University Malaysia

Shahrul Azman Mohd Noah
Universiti Kebangsaan Malaysia

Malaysian Food Composition Ontology Evaluation

Norlia M. Yusof and Shahrul Azman M. Noah

***Abstract*—The Malaysian food composition ontology (MyFCO) models a dietitian's knowledge in designing dietary menu planning. Ontology Development 101 is the method of ontology modeling. The objective of this paper was to share the experience in evaluating MyFCO. It specifically focused on the validation activity using OOPS! tools. The results obtained from OOPS! showed that MyFCO was free from critical error; however, it had three important and six minor pitfalls. Four pitfalls were repaired, whereas the others were remained. The integration between automatic and conventional validation` approaches enhanced the quality of ontology being modeled. The tools improved the conventional approach with faster, easier, and less subjective of a diagnosis activity. Whereas for repair activity, it recommended solutions for the pitfalls.**

***Index Terms*—Food ontology, ontology evaluation, ontology validation, ontology diagnosis, ontology repair, OOPS!.**

I. Introduction

This paper is an extended version of an earlier work published in [1]. Previously, the ontology evaluation phase was explained in brief. This paper will discuss the topic more in depth because it is crucial for the ontology modeling process. It allows the common modeling errors to be avoided and to check the technical quality of the ontology being modeled.

As defined in [2], ontology evaluation refers to the activity of checking the technical quality of an ontology against a frame of reference. Ontology evaluation consists of ontology validation process and verification activity. An activity is an action to be performed, including its required input and output information, whereas a process is a group or set of ordered activities. The definition of activity and process are literally taken from [2]. Ontology validation process can be divided into an activity of ontology diagnosis and repair. The first activity must be carried out before the latter. Ontology diagnosis activity identifies common modelling errors, whereas ontology repair activity solves the errors. Ontology verification activity involves the comparison between ontology and a frame of reference, which is obtained during the ontology requirement specification activity.

In ontology development 101 (OD 101) method, evaluation involves four types of references, i.e., competency questions, application-based, modeling guidelines and expert domain. The first, third and fourth type of reference are used during ontology modeling, while the second is used with the application. In this paper, we only considered the ontology evaluation during ontology modeling. Thus, the application-based reference is out of scope.

Competency questions are used as reference for verification activity. However, MyFCO does not create competency questions and the reason is explained in [1]. Hence, verification activity was not carried out in this paper. For validation process, OD 101 suggests two frames of references, i.e., expert domain and modeling guidelines. However, expert domain may lack time, may be uncooperative, and it is usually scarce. The modeling guidelines in OD 101 use manual approaches, hence to support and improve them, the integration of an automatic approach within this activity was proposed. The tools approach offers easier, less subjective, more complete, and faster evaluation [3].

The latest ontology evaluation tool, which is the OntOlogy Pitfall Scanner! (OOPS!) was chosen to validate the Malaysian food composition ontology (MyFCO) content. It was chosen due to its ability to perform both ontology diagnosis and repair activity. The objective of this paper was to share the process of evaluating MyFCO by using OOPS! (http://oops.linkeddata.es/). The paper is organized as follows: Section II briefly explains the research method for validation process in MyFCO, while Section III discusses the implementation of the validation process. Section IV summarizes the results and finally, Section V concludes the findings.

II. Research Method

This section briefly explains the activities that were carried out for ontology validation process. Ontology diagnosis activity started with a rough first pass of ontology modeling. It checks if the ontology content conforms to the guidelines. This is the part where the modeling guidelines were claimed to be used to evaluate ontology during ontology modeling. The ontology content will be repaired if it does not conform to the guidelines. After manual validation process, the ontology modeling continues with the automatic approach by using the OOPS! tools. Fig. 1 shows the validation process of MyFCO.

III. Implementation

This section discusses the diagnosis and repair activity implemented by the modeling guidelines and OOPS!. Sub-section A outlines the guidelines for class hierarchy to conform to during the diagnosis activity. The guidelines were taken from OD 101 and are shown in italic font. Sub-section B analyzes the results of MyFCO after being diagnosed by OOPS! and Sub-section C explains the repair activity for

Manuscript received October 10, 2018; revised June 12, 2019. This research is partially supported by the Universiti Kebangsaan Malaysia under the Grand Challenge Fund DCP-2017-002/3.

The authors are with the Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia (e-mail: norlia@ siswa.ukm.edu.my, shahrul@ukm.edu.my).

doi: 10.18178/ijmlc.2019.9.5.861

important pitfall, according to the recommendations made by OOPS!.

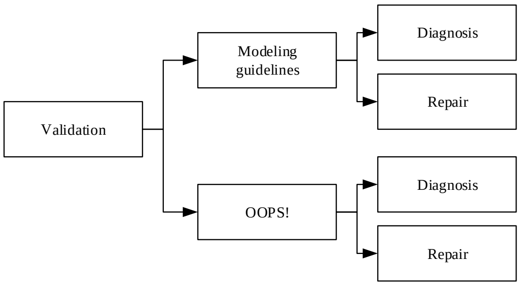


Fig. 1. The proposed validation process.

A. Modeling Guidelines

After the first pass modeling, the class hierarchy was checked to ensure its conformance to the guidelines. Among the guidelines that are adhered to in this study is "*If a class has only one direct subclass, there may be a modeling problem or the ontology is not complete*". It was applied to the last class of macronutrients, i.e., fats. Since it only had one food group, i.e., oils and fats, it could not be assigned as the subclass of the fats class. Therefore, the subclass needed to be dropped. With this guideline, the modeling error was identified and the ontology engineer could correct them. Fig. 2 shows the class hierarchy produced after the first pass modeling. The fats class did not contain its food group; in contrary, it contained only its class category, i.e., monounsaturated (MUFA), polyunsaturated (PUFA), and saturated fat. Meanwhile, carbohydrates and proteins classes contained their food group, respectively. At this stage, the class hierarchy had already defined the eight food groups as the subclasses to their associated macronutrients, which was the superclass. Some of the guidelines that were applied so far in MyFCO will be discussed in the next section. The guidelines provided by OD 101 are to prevent any modeling error. Each of the numbered paragraphs represents the modeling decision that guides a novice ontology engineer in modeling the ontology.

1) Ensuring the class hierarchy correctness

The relation between the subclass and its superclass can be seen as `is-a` (see Fig. 2). For example, starches food group is a carbohydrate; therefore, `Starches` class is a subclass of `Carbohydrate` class. With this guide in mind, the common modeling error, i.e., *singular class is a subclass for plural class* can be avoided. Obviously, starch food group is a starches or `Starch` class is a subclass of `Starches` class is wrong. This error can be avoided by using naming conventions. An ontology engineer was suggested to be consistent when using singular or plural class names throughout the whole ontology. For MyFCO, the plural option was opted since it was thought to be more natural to represent `Starches` class that encompasses all starches food groups.

Another way to check for the correctness of the class hierarchy is to remember that the subclass relationship is transitive. It means that *if B is a subclass of A and C is a subclass of B, then C is a subclass of A.* In MyFCO (see Fig. 2); for example, `Carbohydrates` is a subclass of `RawAndProcessedFoods,` and `Vegetables` is a subclass of `Carbohydrates`, then `Vegetables` is a subclass of `RawAndProcessedFoods.` Thus, the class hierarchy of MyFCO is correct because it conforms to transitive relationship.

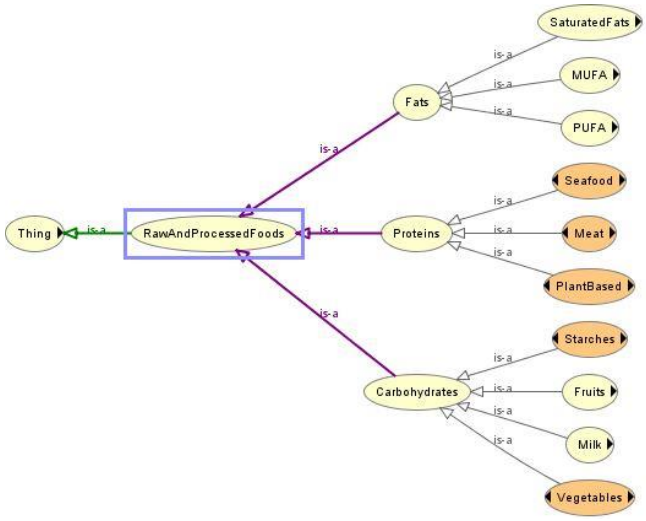


Fig. 2. The first pass class hierarchy.

The class hierarchy correctness can also be ensured by avoiding class cycles. It happens when a class refers to itself through generalization or specialization relationships. Reference [4] referred this situation as circularity error and classified it under inconsistency error. In MyFCO, this cycle exists between `RawAndProcessedFoods` and `CookedFoods` subclasses because they are equivalent. This situation was explained in the previous paper. For example, Fig. 3 shows the class cycles between `Starches` and `CerealBasedDishes` class.

2) Analyzing siblings in a class hierarchy

According to OD 101, the ideal number of subclasses is between two and a dozen direct subclasses. Therefore, two guidelines that were stemmed from this rule are: 1) *if a class has only one direct subclass, there may be a modeling problem or the ontology is not complete*; and 2) *if there are more than a dozen subclasses for a given class, then additional intermediate categories may be necessary*. The first rule was applied to the `Fats` class during the first pass modeling (see Fig. 2). As can be seen from Fig. 2 and Fig. 3, the number of siblings was between two and five.

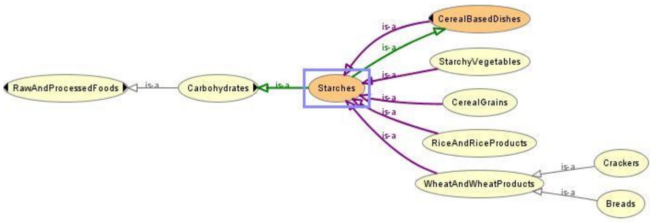


Fig. 3. The class hierarchy between `RawAndProcessedFoods` and `CookedFoods` subclasses.

3) Disjoint classes

Whenever possible, disjoint classes should always be specified. Disjoint subclasses can be specified when there are no common instances between them. This specification helps a reasoner to verify the ontology better. In [4], the disjoint issue was classified under inconsistent and incompleteness errors. The former is referred to as a partition error. It can occur when common classes or instances are defined in disjoint decompositions. For example, class `TropicalFruits` consists of `BananaFruits`, `FifteenSeedFruits`, `OneStoneFruits`, `SegmentedFruits`, and `TenSeedFruits` subclasses. `Guava` is defined as an instance in `FifteenSeedFruits`. However, when `Guava` is defined in `BananaFruits` subclass, the reasoner gives a reminder as shown in Fig. 4. The latter error occurs when disjoint classes are missing. This happens when an ontology engineer models the subclass, but does not define the disjoint character where it should. This error is the most common modeling error found in OWL ontology [5]. In real practice, the reason why disjoints should always be specified is because OWL classes tend to overlap. Thus, to separate a group of classes, it must be declared as a disjoint explicitly.

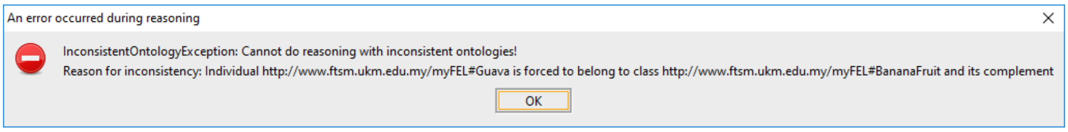


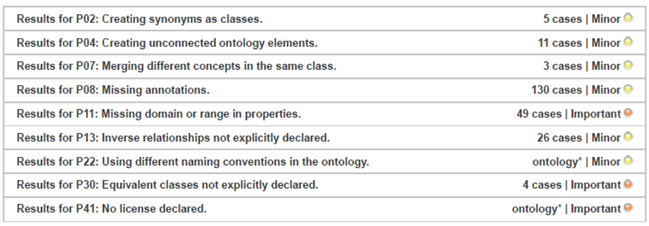
Fig. 4. Disjoint reminder.

4) Naming conventions

The last guideline is to define a naming convention and adhere to it consistently. Some of the advantages of the convention are making the ontology easier to understand, and more importantly, helping to avoid some common modeling errors. There are several things to consider when defining the naming convention. Firstly, the choices of capitalization and delimiters. It is common to assign capital letters for classes and individuals, and lower case for property names. If the class name consists of more than one word, e.g., cooked foods, there is a need to delimit the words. Among the possible options for delimiters are putting a space, combining the words together and capitalizing each new word, and using an underscore or dash to join words. This study followed the recommendation from [6] , where the second option was applied. This option is also known as the camelback notion (see Fig. 3). For Protégé, the space delimiter is no longer available.

Secondly, singular or plural for a class name. This convention has been discussed in the first guideline. It is important to be practiced as it prevents the ontology engineer from making common modeling errors. Thirdly, to distinguish between classes and properties element, prefix and suffix conventions are two common practices. They are combined with the property names. The first convention addition has a prefix, while the latter is an addition of suffix. The current study followed the recommendation by [6], where a prefix `is` or `has` is combined with the property name. This convention is an intuitive solution for ontology engineers because the purpose of the property is clearer. Hence, this approach helps them to distinguish immediately between a class and a property.

B. OOPS!



Results for P02: Creating synonyms as classes.	5 cases | Minor
Results for P04: Creating unconnected ontology elements.	11 cases | Minor
Results for P07: Merging different concepts in the same class.	3 cases | Minor
Results for P08: Missing annotations.	130 cases | Minor
Results for P11: Missing domain or range in properties.	49 cases | Important
Results for P13: Inverse relationships not explicitly declared.	26 cases | Minor
Results for P22: Using different naming conventions in the ontology.	ontology* | Minor
Results for P30: Equivalent classes not explicitly declared.	4 cases | Important
Results for P41: No license declared.	ontology* | Important

Fig. 5. The OOPS! validation results.

The diagnosis results obtained from OOPS! were manually revised in this sub-section. It classified the modeling error(s) into three levels: critical, important, and minor levels. This indicated how serious the error(s) was. The first two levels were mandatory to be corrected. Priority was given for the critical level first. The last level was not mandatory since it was not counted as a problem; however, by doing so, it will improve the ontology performance. Fig. 5 shows the validation results for MyFCO. It achieved three important and six minor pitfalls. The important pitfall is discussed first. The repair recommendations are literally taken from [7].

Fig. 6 shows an excerpt of the first important pitfall, i.e., *P11: Missing domain or range in properties*. Forty-nine cases were detected for this pitfall as they represented 26 object properties and 23 datatype properties without domain and range. OD 101 provided the guidelines regarding this property's facet. The repair recommendation by OOPS! is to declare properties with domain and range constraints for a more complete definition. On the other hand, researchers [5], [6], and [8] advised against the need to specify the domains and ranges of the properties. They do not behave as constraints to be checked; instead, they are axioms for reasoning. It may cause two possibilities, i.e., unexpected classification result where the classes are coerced to be subsumed by another class, or unsatisfiable among classes. This situation can be more difficult in large and complex ontologies. The effect of range and domain constraints as axioms is the most common problem in OWL [5],[8]. In MyFCO, the domain and range of properties are not assigned to avoid the above problems. Thus, no ontology repair action was carried out for this pitfall.

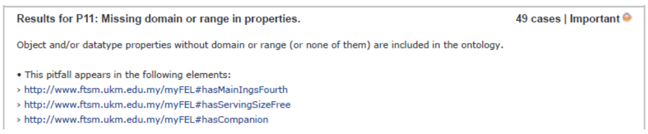


Fig. 6. An excerpt from the first important pitfall.

Fig. 7 shows the second important pitfall, i.e., *P30: Equivalent classes not explicitly declared*. OOPS! detected four pairs of classes that supposedly had the equivalent class between them. They were detected due to the duplicated concepts that exist among them. In OD 101, the guideline regarding this matter falls under the synonym class. Synonyms for the same concept do not represent different classes.

OOPS! offered two repair recommendations, each for different namespace and same namespace. The MyFCO used the same namespace because it did not import and integrate with any existing domain ontology. Thus, the repair recommendation was to remove one of them and to attach its label annotation properties to the remaining class.

However, in the MyFCO, this pitfall emerged due to the nature of food classification in the real world. For example, the fruits food group is a carbohydrate, thus, `Fruits` is a subclass of `Carbohydrate` class. The duplicated concepts lie in `MUFA` class, which represents an individual of `Avocado`. Avocado is a fruit and avocado is also a MUFA. Thus, a fruit is a MUFA. Or, `Fruit` class is a subclass of `MUFA` class. Even though they are synonyms, they are representing different classes. The class of `Oils` and `Oil`, and `Nut` and `Nuts` are duplicated due to the similar situation as the first case.

`Shellfish` and `Mollusc` are not synonyms because the latter is the subclass of the former. In the real world, shellfish consists of crustacea and mollusk. Thus, it is classified that `Shellfish` is the superclass of `Mollusc`. Therefore, it can be concluded that all the four cases are not synonyms. Thus, no repair actions are needed and this pitfall is remained in MyFCO.

Results for P30: Equivalent classes not explicitly declared. **4 cases | Important**

This pitfall consists in missing the definition of equivalent classes (owl:equivalentClass) in case of duplicated concepts. When an ontology reuses terms from other ontologies, classes that have the same meaning should be defined as equivalent in order to benefit the interoperability between both ontologies.

• The following classes might be equivalent:
› http://www.ftsm.ukm.edu.my/myFEL#Fruits, http://www.ftsm.ukm.edu.my/myFEL#Fruit
› http://www.ftsm.ukm.edu.my/myFEL#Shellfish, http://www.ftsm.ukm.edu.my/myFEL#Mollusk
› http://www.ftsm.ukm.edu.my/myFEL#Oils, http://www.ftsm.ukm.edu.my/myFEL#Oil
› http://www.ftsm.ukm.edu.my/myFEL#Nut, http://www.ftsm.ukm.edu.my/myFEL#Nuts

Fig. 7. Second important pitfall.

P41: No license declared is the last important pitfall in MyFCO as shown in Fig. 8. The pitfall concerns the ontology metadata aspect, of which does not have any guidelines in OD 101. The repair recommendation by OOPS! was to include a statement containing the license information using any of the following properties: `dc:rights`, `dcterms:rights`, `dcterms:license`, `cc:license` or `xhv:license`.
This activity will be discussed further in the following sub-section.

Results for P41: No license declared. **ontology* | Important**

The ontology metadata omits information about the license that applies to the ontology.

*This pitfall applies to the ontology in general instead of specific elements.

Fig. 8. Third important pitfall.

Results for P02: Creating synonyms as classes. **5 cases | Minor**

Several classes whose identifiers are synonyms are created and defined as equivalent (owl:equivalentClass) in the same namespace. This pitfall is related to the guidelines presented in [2], which explain that synonyms for the same concept do not represent different classes.

• This pitfall appears in the following elements:
› http://www.ftsm.ukm.edu.my/myFEL#PlantBased
› http://www.ftsm.ukm.edu.my/myFEL#Meat
› http://www.ftsm.ukm.edu.my/myFEL#Seafood
› http://www.ftsm.ukm.edu.my/myFEL#CerealBasedDishes
› http://www.ftsm.ukm.edu.my/myFEL#VegetableDishes

Fig. 9. First minor pitfall.

MyFCO has six minor pitfalls. The description of pitfalls that were not provided with screen shots was taken directly from the OOPS! interface. Fig. 9 shows the first minor pitfall, i.e., *P02: Creating synonyms as classes*. They are similar with the second important pitfall, except that they have an equivalent relationship among the synonym classes. Five cases were detected under this pitfall. It represented the number of equivalent classes between several subclasses of `RawAndProcessedFoods` and `CookedFoods` (see Fig. 10).

In the real world, foods are classified into two general categories, i.e., raw and processed, and cooked food. From eight food groups, five of them have these two categories, i.e., starches, vegetables, plant-based protein, seafood, and meat. The other three food groups, i.e., fruits, milks, and fat, belong to the raw and processed foods category. The food groups with two general categories use almost similar concept names; for example, `meat` and `meatDishes`. However, they are not synonyms. They have an equivalent relationship because in the real world, dietitians would design the dietary menu planning according to the food groups in raw and processed foods only. A detailed explanation regarding this issue was discussed in [1].

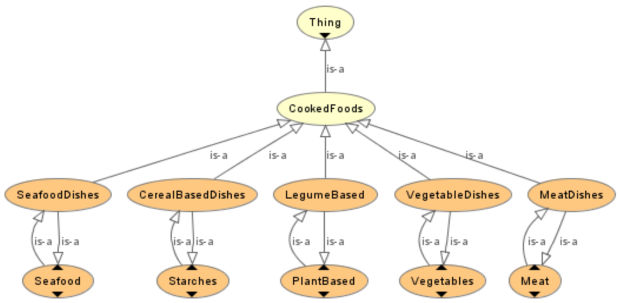


Fig. 10. The class hierarchy of `CookedFoods` class.

The repair recommendation for this pitfall was to create one class with different labels (`rdfs:label`), one for each synonymous term. As ontology reflects the real world, it must adhere to the classification in the domain. Therefore, the current researchers did not make any corrections and this pitfall remained in MyFCO.

The second minor pitfall is *P04: Creating unconnected ontology elements*. The description of this pitfall is that ontology elements (classes, object properties, and datatype properties) are created in isolation, with no relation to the rest of the ontology. Eleven cases were detected, which represent the classes at the root of the hierarchy in MyFCO (see Fig. 11).

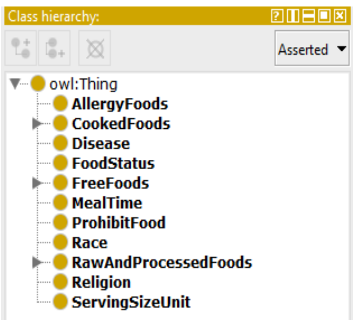


Fig. 11. The superclass hierarchy.

Results for P07: Merging different concepts in the same class. **3 cases | Minor**

A class whose name refers to two or more different concepts is created.

• This pitfall appears in the following elements:
› http://www.ftsm.ukm.edu.my/myFEL#RiceAndRiceProducts
› http://www.ftsm.ukm.edu.my/myFEL#WheatAndWheatProducts
› http://www.ftsm.ukm.edu.my/myFEL#RawAndProcessedFoods

Fig. 12. Third minor pitfall.

Actually, they are not isolated. They are direct subclasses of the most general class in an ontology, i.e., `owl:Thing.` Hence, no correction was made for this pitfall. OOPS! detected this pitfall because the class is not part of any hierarchy (using `rdfs:subClassOf`). This detection is applicable for explicitly hierarchical relationship. Meanwhile, the classes at the root of the hierarchy are implicitly a subclass of class `Thing`. OOPS! noticed the limitation. This is the reason the pitfall must be revised manually because not all of the pitfalls are factual errors.

The third minor pitfall is *P07: Merging different concepts in the same class* as shown in Fig. 12. The repair recommendation for them as to create one class for each concept represented in the affected class. This study merged two different concepts in the same class to reflect the real world food composition. Food either originates from nature, or is produced by man. Since ontology is a reflection of the real world, thus no correction was made for this pitfall.

The fourth pitfall is *P08: Missing annotations*. The description of this pitfall was in creating an ontology element, human readable annotations have failed to be attached to it.

The repair recommendation was to include label annotation properties (`rdfs:label`) and description annotation properties (`rdfs:comment`). These are the two most commonly used annotation properties, besides `owl:versionInfo` [9]. This pitfall will be repaired for further reuse.

The fifth minor pitfall is *P13: Inverse relationships not explicitly declared*. The description of this pitfall was when any relationship (except for those that were defined as symmetric properties using `owl:SymmetricProperty`) did not have an inverse relationship (`owl:inverseOf`) defined within the ontology. OOPS! listed all of the object properties in MyFCO, which did not have the inverse relationship (see Fig. 5.). OD 101 provided the guidelines regarding inverse relationships. Reference [6] stated that the specification of the inverse properties is needed for completeness. This is because all relationships, apart from the symmetric ones, could have an inverse relationship in principle [7]. The inverse relationships for the object properties will be specified in future works.

The final minor pitfall is *P22: Using different naming conventions in the ontology*. The description of this pitfall was ontology elements are not named according to the same convention (for example, CamelCase or use of delimiters as "-" or "_"). As this pitfall applied to the ontology in general, thus, no specific elements were given. OD 101 provided guidelines on naming conventions. It emphasized consistency with the chosen naming conventions. The benefits from the consistency help to avoid modeling mistakes, improve readability, and ease the understanding of ontology. In MyFCO, the naming convention adherence for the class and individual name are capitalized whereas the property name has no capitalization. The pitfall occurred in the object property name in which the current researcher accidentally assigned one of them with a capital case, i.e., `IsServedWith.` It is supposed to be written as `isServedWith.` As the pitfall was quite simple to repair, renaming was conducted directly.

C. Ontology Repair Activity

The license declaration pitfalls will be repaired as recommended by OOPS!. Two questions that will guide the license declaration are, which predicate can be used for rights declaration? and which license can be used in the rights declaration? [10]. The first is known as work property, while the latter is known as license property [11]. The following paragraph will provide answers for both questions.

The Dublin Core Metadata Initiative (http://dublincore.org) is a metadata standard for a wide range of resources including physical resources such as documents and artworks; and digital resources such as images, videos and web pages. The annotation property that is suitable for license is `rights`. It is defined as the *information about rights held in and over the resource.* It can be used with `dc:` (http://purl.org/dc/elements/1.1/) and `dcterms:` (http://purl.org/dc/terms/) namespaces. Another property of Dublin Core that is related to intellectual property license is `license` (A legal document giving official permission to do something with the resource). This property gives rise to creative common metadata (http://creativecommons.org/ns#), i.e., `cc: license`. Apart from the Dublin Core vocabulary, XHTML vocabulary also provides license property (https://www.w3.org/1999/xhtml/vocab) with the namespace of `xhv:`.

It is important for a researcher to determine the most appropriate license. A study by [10] classified the most common data license according to its restrictiveness. Among the types of licenses are public domain, attribution, share-alike, with restrictions, closed, and other. Since the purpose of ontology vision is to enable domain knowledge reuse, the open license is assumed to be the most relevant for its rights declaration [12]. It is also similar to the attribution license. Taken from [10], the license waives all the possible rights, requiring only a mere attribution. The most popular open license is Creative Commons Attribution License (CC BY) [12], [13].

In Protégé, the metadata annotations are under the ontology header view. Fig. 13 shows the interface of metadata annotations where the license of the ontology is declared. The predicate for the license declaration of MyFCO was taken from the `dcterms:license` and assigned to the CC BY license.

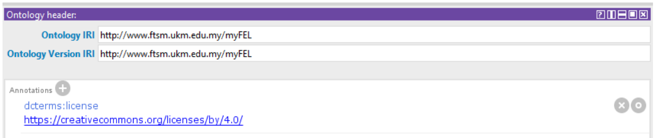


Fig. 13. License declaration.

To check whether the repairing activity was implemented correctly, the ontology had to be reevaluated. OOPS! also recommended the reevaluation because it enabled the discovery of hidden errors. Fig. 14 shows that the MyFCO is able to repair the pitfall (P41) that is no longer listed in the evaluation result. It also shows that another pitfall is being repaired (P22) that is also no longer listed as the modeling error in MyFCO.

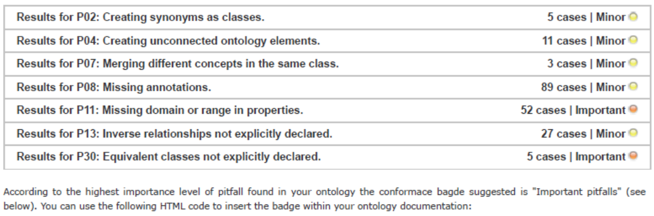


Fig. 14. Reevaluation results.

IV. Results and Discussion

The diagnosis results obtained by MyFCO were free from any critical pitfalls. This showed that the modeling guidelines had helped the ontology engineers to model the ontology in a systematic manner. Pitfalls can be categorized in two types, which are remain and repair. The first category is mostly due to the real world factors. Since ontology is a model of the real world, it must reflect that. The pitfalls under this category were P30, P02 and P07. On the other hand, the remaining pitfalls, P04 and P11 were due to OOPS! limitation and the guidelines outlined by other researchers, respectively. Not all pitfalls in OOPS! are considered as factual errors. They depend on the modeling decision or requirement. This shows that OOPS! is not a rigid tool.

P41, P08, P13 and P22 fell under the repair category. The activity started with the easiest pitfall to be repaired, which was P22, followed by the level of seriousness. Thus, P41 became the second pitfall to be repaired. The other two remaining pitfalls were taken as future works. Two pitfalls, namely P13 and P22 happened due to the careless mistake made by the current researchers. OD 101 had provided the guidelines, however, they were overlooked. On the other hand, P41 and P08 happened because the current researchers did not have any knowledge about them as OD 101 did not give any guidelines concerning them. Both pitfalls were related to ontology annotation, which started to gain attention from the community in 2005 [14] whereas OD 101 was published in 2001.

OOPS! plays a significant role in ensuring the ontology is free from the common pitfall by double checking the modeling guidelines provided by OD 101. For example, it supports the latest common modeling errors that are not listed in OD 101, such as the annotation issue. The main advantage of OOPS! is the repair recommendation made by it. It shows how the ontology element can be repaired to improve the ontology technical quality. In MyFCO, the evaluation results from OOPS! have improved the inferencing, understanding, clarity and metadata aspects. Nevertheless, OOPS! has a limitation. It still needs to be revised manually in some cases of the pitfall.

V. CONCLUSION

This paper discusses the manual approach, i.e., modeling guidelines and automatic approach, i.e., OOPS! for validating MyFCO. Its contributions include the mapping of frame of reference in OD 101 onto the sub-activity in ontology evaluation and their suitable time to evaluate. It also proposes the validation process by integrating the manual and automatic approaches. Besides, it shows how to solve important pitfall based on its recommendations. Finally, it produces better quality of food composition ontology.

For future works, apart from the repair activity for the two remaining pitfalls (P08 and P13), MyFCO will be upgraded with linked open data to enable domain knowledge sharing and reuse.

REFERENCES

[1] N. Yusof, S. A. Noah, and S. T. Wahid, “Ontology modeling of malaysian food composition,” in *Proc. Third International Conference on Information Retrieval and Knowledge Management (CAMP’ 16)*, 2016, pp. 149–154.

[2] M. C. Suárez-Figueroa, A. de C. Guadalupe, and G.-P. Asunción, “Lights and shadows in creating a glossary about ontology engineering,” *Terminology International Journal of Theoretical and Applied Issues in Specialized Communication.*, vol. 19, no. 2, pp. 202–236, 2013.

[3] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, “Modelling ontology evaluation and validation,” in *Proc. Third European Semantic Web Conference*, 2006, pp. 140–154.

[4] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web*, Second Edition, Springer-Verlag London, 2004.

[5] A. Rector *et al.*, “OWL pizzas : Practical experience of teaching OWL-DL : Common errors and common patterns,” in *Proc. International Conference on Knowledge Engineering and Knowledge Management*, 2004, pp. 63–81.

[6] M. Horridge, *A Practical Guide to Building OWL Ontologies using Protege 4 and CO-ODE Tools*, 2011.

[7] M. P. Villalón, “Ontology evaluation : A pitfall-based approach to ontology diagnosis,” Ph.D. Dissertation, Department of Artificial Intelligence, Polytechnic University of Madrid, 2016.

[8] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist Effective Modeling in RDFS and OWL*, Second Edition, Morgan Kaufmann, 2011.

[9] J. Hebeler, M. Fisher, A. Perez-lopez, and R. Blace, *Semantic Web Programming*, Indiapolis, Indiana: Wiley Publishing, Inc., 2009.

[10] V. Rodríguez-Doncel, A. Gómez-Pérez, and N. Mihindukulasooriya, “Rights declaration in Linked Data,” in *Proc. Fourth International Workshop on Consuming Linked Data (COLD)*, 2013.

[11] T. Segaran, C. Evans, and J. Taylor, *Programming the Semantic Web*, First Edition, O’Reilly Media, Inc., 2009.

[12] Australian Research Data Commons, *Research Data Rights Management Guide*, 2018.

[13] M. Poblet *et al.*, “Assigning creative commons licenses to research metadata : Issues and cases,” in *AI Approaches to the Complexity of Legal Systems*, 2015, pp. 245–256.

[14] J. Hartmann, R. Palma, Y. Sure, M. C. Suarez-Figueroa, P. Haase, A. Gómez-Pérez, and R. Studer, “Ontology metadata vocabulary and applications,” in *Proc. OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, 2005, pp. 906–915.

Norlia M.Yusof was born in Johore, Malaysia. She received the Bsc with honors in computer from the Universiti Teknologi Malaysia, Johore in 2001 and Msc in intelligent systems from Universiti Utara Malaysia, Kedah, in 2005, respectively. Currently, she is completing her PhD at Universiti Kebangsaan Malaysia, Selangor. Her research interests are case-based reasoning, semantic technology, AI in design and constraint satisfaction. She has published one journal and four proceedings in the related research areas.

Shahrul Azman M. Noah was born in Johore, Malaysia. He received the BSc with honors in mathematics from the Universiti Kebangsaan Malaysia in 1992, MSc and PhD degrees in information studies from the University of Sheffield, UK, in 1994 and 1998, respectively.

He is a professor in the Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia and currently heads the knowledge technology research group. His current research work is focused on information retrieval with special emphasis on semantic search, ontology-based retrieval system and recommender systems. He has published various research articles in these areas. Prof. Noah is currently a member of the IEEE Computer Society and International Association for Ontology and its Applications (IAOA).