Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho

Vol. 7, No. 3, pp. 95–100, September.-2019

p-ISSN: 2663-628X
e-ISSN: 2663-6298

DESIGN AND IMPLEMENTATION OF E-CAMPUS ONTOLOGY WITH A HYBRID SOFTWARE ENGINEERING METHODOLOGY

Karwan Jacksi

Department of Computer Science, Faculty of Science, University of Zakho, Zakho, Kurdistan Region, Iraq - karwan.jacksi@uoz.edu.krd

Received: Jul, 2019 / Accepted: Sept., 2019 / Published: Sept.,2019 ***https://doi.org/10.25271/sjuoz.2019.7.3.613***

ABSTRACT:
Semantic Web according to the vision of the W3C is the future of WWW (or Web). It is an extension of the current Web through standards by the W3C. Data of the Semantic Web has well-defined meanings, can be understood by devices and allows machines and people to work in collaboration. Ontologies are vital components of the Semantic Web infrastructure and are more often recognized as the backbone of the Semantic Web. Although numerous developments occur in the field of developing ontologies along the lines with the Semantic Web implementation, but standardizing the process models, tools and methodologies need to be improved in the future. In literature, experts in ontology engineering have stated that setting a methodology for developing ontology applications with support of integrated tools is an essential task for ontology engineering to be succeeded. In this paper, an e-campus ontology for educational purposes is designed and implemented, and mainly focused on the learning hierarchy of C-sharp programming language. A hybrid methodology based on software engineering approaches for developing ontologies is presented. Finally, the developed methodology is applied on the implemented ontology.

KEYWORDS: e-Campus, e-Campus Ontology, C#, C-Sharp Programming, Methodology, Software Engineering, Semantic Web, Ontology Engineering.

1. INTRODUCTION

The Semantic Web (SW) aim at providing information as formal, well defined meanings, compatible, sharable knowledge base, and can be processed by machines (Jacksi, Zeebaree, and Dimililer 2018). Ontology acts an important role in the SW technology as it's famous as of the backbone of the SW structure, and is the vital element of SW infrastructure (Jacksi, Dimililer, and Zeebaree 2015). Web Ontology Language (OWL) and Resource Description Framework-Schema (RDFS) are the recommendations of the World Wide Web Consortium (W3C) for data representation models so as to deliver foundations for the ontology descriptions (Jacksi, Dimililer, and Zeebaree 2016).

Ontology is a collection of semantically related concepts built on a limited number of predefined relations and terms of a domain. These terms and concepts can be represented visually so as to ease the representation for both syntactic and semantic data (Fensel 2002). Ontologies provide distinct descriptions in their information, as a result, they are used in numerous fields and applications since its knowledge representation is understandable and processable by software agents and systems (AL-Zebari et al. 2019).

In Web, once abstract data is distributed across several knowledge bases, ontologies are the solely resolution as commonplaces to interpret the mutual senses of the domain key terms. Hence, significant concerns seek the development of ontologies. Several reasons make the mission of developing ontologies a challenge, unavailability of standardized methodologies with support of integrated tools is the most common reason (Akerman and Tyree 2006).

So far, there is no unified approach for formal representation of information on the Web according to the Software Engineering aspects (Abbas 2016). However, it is clear by some means that some overlaps still occur between fields of software engineering and recognized works in systems, and some of the new schemes accept distinction in hybrid approaches to developing systems merging the technologies of Semantic Web with the methods of well-known development formality and standardized modeling languages like Unified Modeling Language (UML) (Bhatia, Kumar, and Beniwal 2016; Happel and Seedorf 2006).

The previous literature indicates that the results of well-established methodologies and their support of modeling languages and standards for specifying software-intensive systems, prepared the maturity of Software Engineering. Despite deploying a lot of techniques and methodologies for ontology engineering, but still there is a gap between ontology engineering and software engineering, because there is no unique standard method to model a domain, subsequently no unique standard methodology to develop ontologies, even though ontology engineering has similar characteristics with software engineering (Kim and Choi 2007). Henceforth, developing a standardized methodology with support of integrated tool for domain modeling will build a major distinction in bridging such gap (Gaševic, Djuric, and Devedžic 2006).

In this work, electronic campus ontology for educational purposes for the University of Zakho (UoZ) has been created. It is built using Protégé, which is a free, most popular tool and open source ontology editor developed at Stanford University. Protégé supports the ability to integrate new tools and utilities which makes developing ontologies easier. Then the focus went mainly on the learning part of the ontology where the C# (pronounced C sharp) programming language is taught.

The rest of this paper is organized as follows. Related works is presented in section 2 to study the topic background, section 3 discusses the engineering approaches of the proposed methodology. Section 4, presents the application of the proposed

methodology applied on the built ontology. Finally, suggestions and feature works are concluded in section 5.

2. RELATED WORKS

In order to develop an ontology in educational domain, research works have suggested many approaches. A C-programing ontology has been developed by (Sosnovsky and Gavrilova 2006) for educational purposes for designing a visual ontology. In their development they followed by a five-step algorithm. In the ontology development, the visualization technique effects on both analyzing and synthesizing processes. Later on, an ontology for Java language has been developed using the same five-step algorithm as well (Ganapathi, Lourdusamy, and Rajaram 2011). There are five main phases as a core of the proposed method: "Glossary development, Laddering, Disintegration, Categorization, and Refinement". Most of the published works were focusing on the knowledge structuring during the ontology design, because it is more appropriate to be applied in teaching systems where the user can understand it much more than factual details. In (Lee, Ye, and Wang 2005) an ontology of a Java Learning Object (JLOO) has been presented in a framework, where this ontology has been used to organize and develop the learning objects in a pre-course for Java language in a learning system. The computing curriculum CC2001/ACM and IEEE/CS were used in the JLOO ontology. During the ontology development, JLOO followed an oriented model.

Santhosh John in his work (John 2010) emphasized the importance of having a standardized approach with the support of the integrated tool. The modeling languages such as UML and other established designs should get benefit from ontology engineering and software engineering. IBM China Research Lab has developed the Eclipse Modeling Framework (EMF) using ontology engineering and ontology identification model, which facilitates the conversion of models (Pan et al. 2006).

3. PROPOSED METHODOLOGY

The main aim of this work is to fill the existing gap between software engineering and otology engineering through the impact of well-proven way and method models of software engineering field for the development of ontologies. The two aspects such as engineering and philosophy of the proposed methodology were taken from available standards. The two well-proven software process models: Rational Unified Process (RUP) (MacIsaac 2003), and traditional, linear, waterfall approach derived the proposed methodology.

The stages of methodology development of the proposed ontology include lifecycle proposed through Methontology (Corcho, Fernández-López, and Gómez-Pérez 2006), a designed methodology for ontology manufacturing by the FIPA (Foundation of Intelligent Physical Agents), which advocates inter-operability over agent-based implementations. The value of Methontology engineering eturns back to the medium exemplification with regard to various models like stipulation semi-formal-model specification utilizing a group of medium exemplification, imaginary model and characterize model (e.g., description logic ontology UML profile) that is going to be executed in an ontology application language (e.g., OWL).

There are three stages of the proposed methodology; pre-development, development and post-development. The first stage, *pre-development*, is related to the feasibility work of ontology field which contains evaluating the range of the field including all details. The second stage, *development*; its main aim is to produce the fundamental model of the platform. Finally, in the last stage, *post-development*, the application model definition is built. Each of these stages receives a certain product with the general aim of originating functional ingredients in accordance with ontology that can be utilized in various systems.

The last structure of the designed methodology suits the different phases of ontology development into the stages of a gradual, iterative, and constant process of methodology development, Rational Unified Process (RUP). This leads to supply rigorous approach to attributing responsibilities and duties within a development group. The RUP catches a lot of the finest exercise in updated software improvement in a way that is appropriate for the development of ontology as well. All stages of proposed developing ontology methodology in conjunction with their phases are suited into the four phases of RUP development: *inception*, *elaboration*, *construction*, and *transition*; and it shows how the workflows are integrated into these phases.

The primary aim of the inception stage is to fulfill agreement among the interested parties on life-cycle goals for the plan and identify the study feasibility. The goal of the elaboration stage is to investigate the domain issue, set up a strong architectural base, improve the plan of the project, and remove the factors of substantial risks of the project. Throughout the construction stage, almost all ingredients and implementation characteristics are improved and incorporated to the outcome, as well as all characteristics are completely checked. All these aims and objectives are assigned in better shape to ontology definition stage. Lastly, the ontology application is identified towards the transition stage. Figure 1 explains the final structure of the proposed methodology.

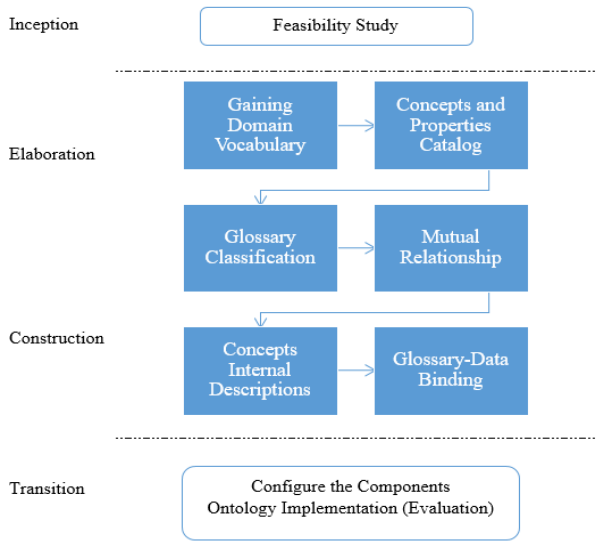

Figure 1. Structure of the Proposed Methodology.

4. METHODOLOGY IMPLEMENTATION

In order to implement the proposed methodology, the educational domain has been chosen, where the reusability of knowledge is important. The increasing volume of electronic and E-learning systems in the educational module have a duty to care the import and export of information and it should be in a standardized way. It is important to use the ontology as a basic foundation for information that will allow a different language to deal with each other in the same language. As mentioned earlier in this paper the ontology that is presented is the University of Zakho (UoZ) ontology which includes a wide range of classes and properties for educations purposes. A part of the ontology is visualized using the OntoGraf plugin of the Protégé as shown in Figure 2. General structure of the developed ontology is shown in Figure

3. The ontology has courses inside and one of the courses, C# programming, is taken to apply the proposed methodology. The main reason for taking the C# language part of the ontology is related to attempts by the industry language to create more effective strategies that is related to learning by combining many perspectives in the domain. The OWL of the C# programming course can be seen in Figure 4.

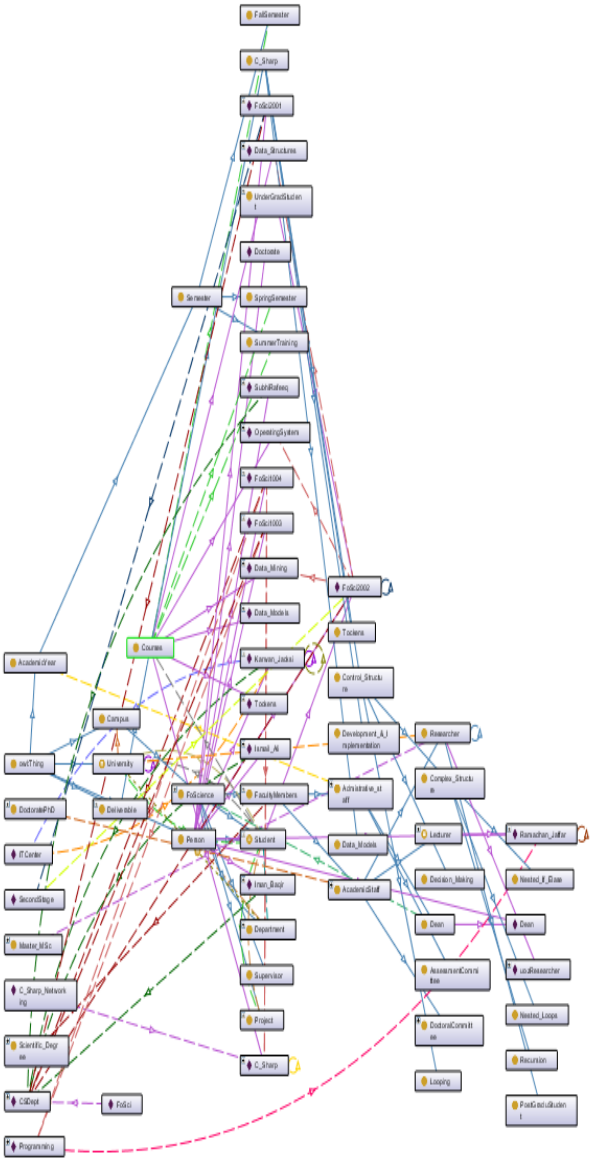

Figure 2. Ontology Visualization using OntoGraf

Nowadays, many instructors introduce the C# programming based on different aspects and parameters, such as the selection and arrangements of the topics. Although the presented materials depend on the teacher arrangement, and the hierarchical structure is not ignored. Below are the stages of the proposed methodology according to the domain that has been chosen:

4.1 Feasibility Study

In this stage, the C# programming part of ontology focuses on C# course, which is the C# 7.3. that was released in 2018 alongside Visual Studio 2017 version 15.7.2 (Visual Studio 2017 15.7 Release Notes | Microsoft Docs 2017). This part of ontology is derived for the C# learning domain, which takes into account the C# approach to programming variances.

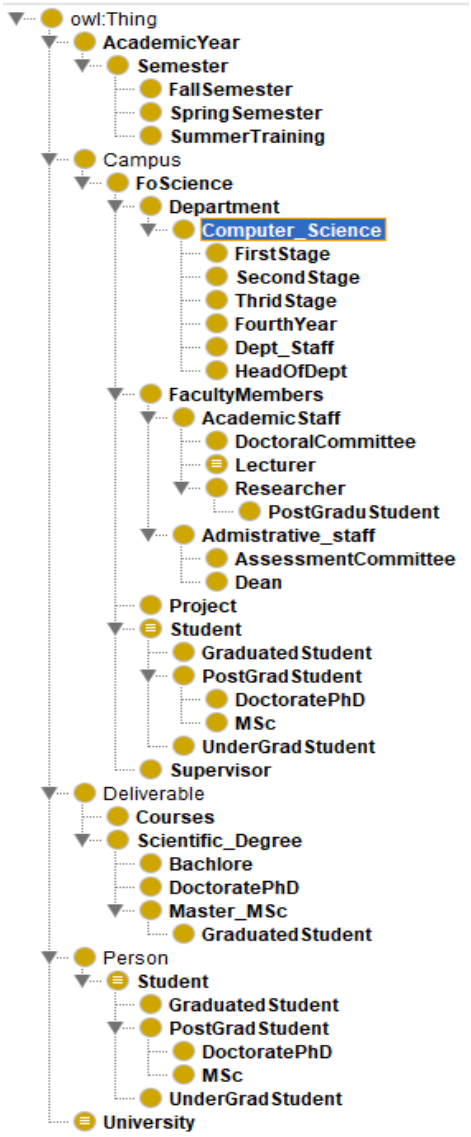

Figure 3. General Structure of the Implemented Ontology

The feasibility of the ontology as whole, as shown in Figure 2 is very high, the C# programing fraction can later be integrated with Moodle, the used e-learning system at the UoZ, after evaluation. The developed ontology is classified as shown in Figure 5. The taken part of the ontology, C# programming language, with it is relations such as faculty, instructor, and the department that offers, are shown in Figure 6. In the developed ontology, the word C_Sharp is taken instead of C# because the ontology editor tool assumes the character (#) as an illegal character.

```
<owl:NamedIndividual rdf:about="http://www.owl-ontologies.com/ontouoz.owl#C_Sharp">
    <rdf:type rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#Courses"/>
    <ComposedOf rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#Tockens"/>
    <HasCode rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#C_Sharp"/>
    <HasMember rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#Data_Models"/>
    <IsTaughtBy rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#Karwan_Jacksi"/
    <ProgrammeOf rdf:resource="http://www.owl-ontologies.com/ontouoz.owl#CSDept"/>
    <CourseCode rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CS-CS201</Course
    <CourseDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Computer
    <CourseObjective rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Computer Sc
    <ECTS_Credits rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">6</ECTS_Credi
</owl:NamedIndividual>
```

Figure 4. C# Programming Course as OWL

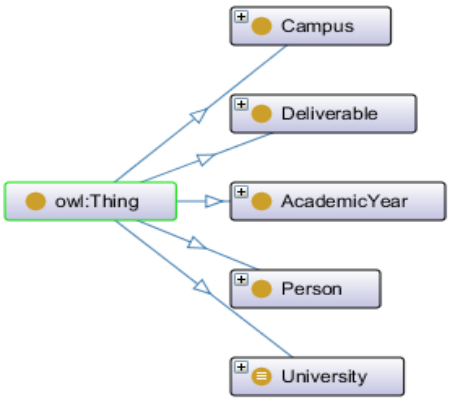

Figure 5. Classification of the Developed Ontology

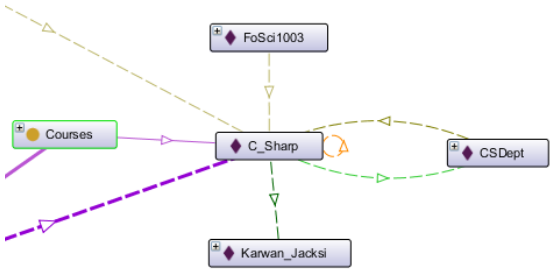

Figure 6. C# Programming Course Relations

4.2 Gaining Domain Glossary

In the development stage of the ontology, the classes of the C# programming class will be taken from the vocabulary/ glossary of the domain. The materials of the ontology and the domain glossary are taken from different resources such as: introduction to programming, Object Oriented Programming (OOP), distributed programming with C#, and data structures and algorithms, from Computer Science Department at the UoZ, IT Department at Zakho Technical Institute, Text books and online resources. The main taxonomies and concepts used for the first edition of the C# programming are: Development and Implementation, Tokens, Data Models, and Control Structure as shown in Figure 7.

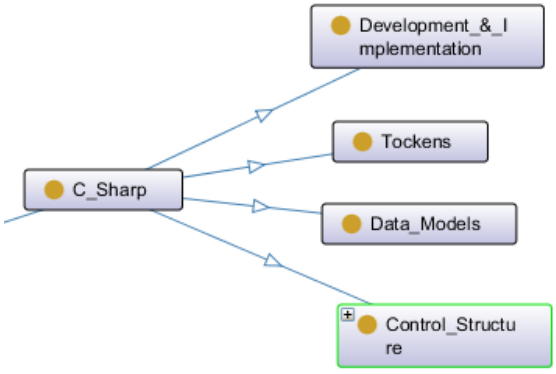

Figure 7. Taxonomy of C# Language Concepts

4.3 Concepts and Properties Catalog

A number of concepts and properties are produced in this stage of the ontology. Also, the semantic translation of the concepts is achieved in this stage. The shared vocabulary attained in this stage by assigning the property and properties value. To model this part of the ontology, Protégé 5.5 has been used, a snapshot of the tool is shown in Figure 10. The general Ontology structure based on the concepts derived is shown in Figure 3. The subclass structure of the class C# programming and the suitable properties is made and classified based on the four main concepts or classes shown in Figure 7. The rest of the subclasses for C# class are then generated as shown in Figure 8.

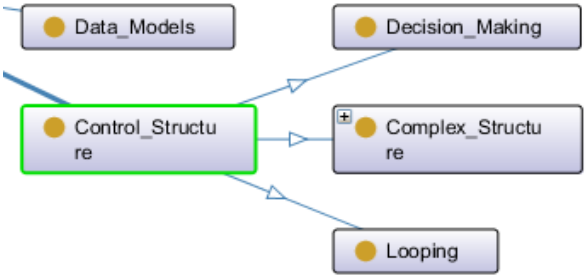

Figure 8. Binary Relationship on Control Structure Concept

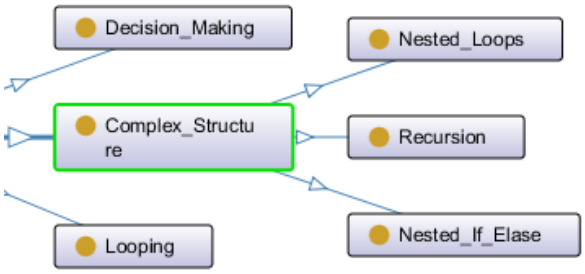

Figure 9. Binary Relationship on Complex Structure Concept

4.4 Glossary Classification and Mutual Relationship

The main idea behind this stage is the structure concept. The classification of the main concepts is delivered in this stage. The followed approach is the Top-Down where the start of the procedure from the most general concepts and subsequent of the concepts. As shown in Figure 7 the version of the classification of concept C# language. The classes of the ontology will be organized in a hierarchical form, if any individual in class A, which is a superclass of B, this means that the individual in B is also an individual in A. The concepts that are belonging to the C# language are shown in Figure 7, Figure 8 and Figure 9, which are: Data Model, Control Structure, Development and Implementation, and Tokens. The developed ontology (UoZ_Ontology) up to the date of submission of this paper can be found online (http://karwanjacksi.net/uoz_ontology).

4.5 Concepts Internal Descriptions

In this stage of the ontology, the concept attributes and relationships are described for the concepts. The class will attach these properties. The relationships between the members of the individual of each class with other items will be included in this phase. The property value of the superclass C# assigned to the object property assertion "Karwan_Jacksi" is shown in Figure 11. Where Data property and Object property are indicated. The other task of this phase is the number of features of value are described such as the type, number of value and others. For instance, the name should be set it as string and the type of the value string as well. Each individual can have more than one values of the class as shown in Figure 11.

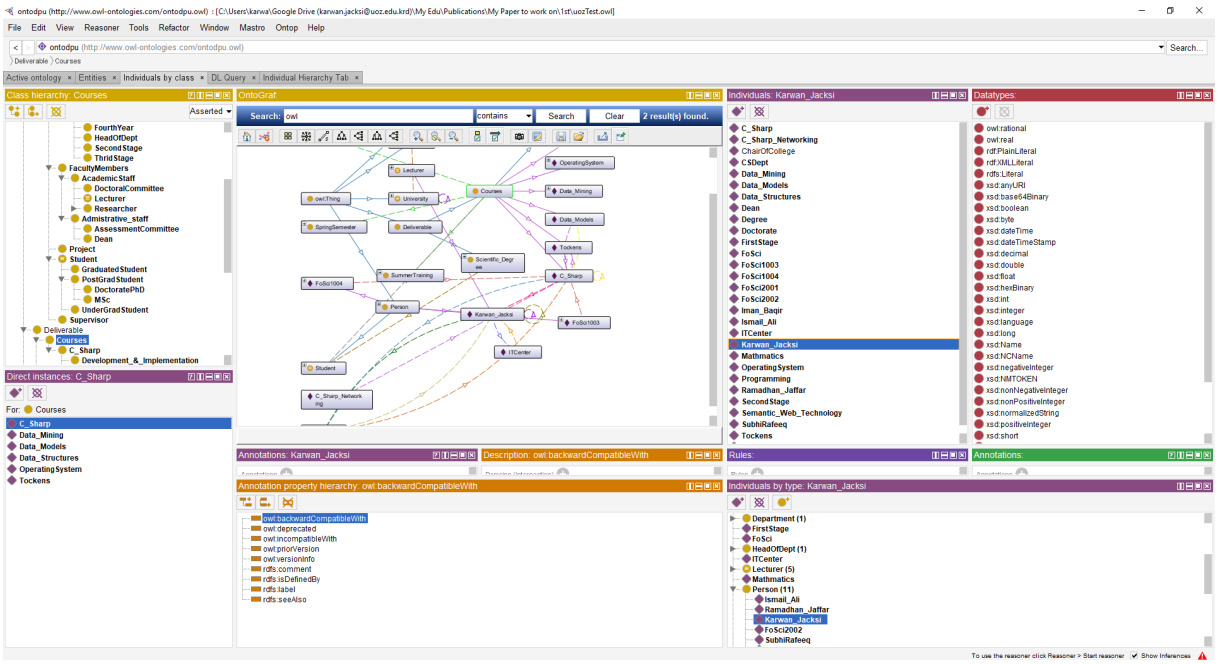

Figure 10: Protégé with property values and complex restrictions and rules

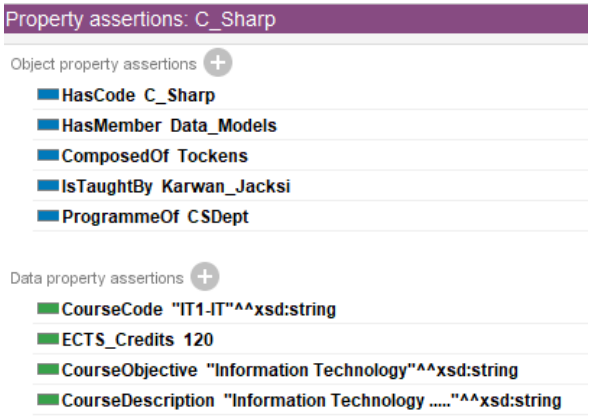

Figure 11: Superclass to Object Property Assertion

4.6 Glossary-Data Binding

Creating the individuals and instances of the classes is the last step. In order to create an individual for each classes in the ontology, first the developer should choose the specific class to add the individual to that class; second, give a value/data to each instances such as (string, time, data…etc.) as mentioned at the previous stage. At this point, the word or the glossary is certainly associated with its actual data.

5. CONCLUSIONS AND FUTURE WORK

In this paper, a hybrid methodology for developing ontologies is proposed based on the well proven software engineering concepts. An extensive e-campus ontology for the University of Zakho (UoZ) is created that include the e-learning part as well. The proposed methodology has been applied on the developed ontology and focused mainly on the part which is concerning the learning of the course C# programming. The developed ontology provides a dense hierarchical structure of e-campus and includes the topics for C# programming which can be used for teaching and learning. The implemented ontology can be integrated with any e-learning systems, and it can be reused for educational organizations where the e-campus can be applied. The visualization of the classes and concepts are made using the OntoGraf plugin within the Protégé.

In future work, the ontology will be integrated with the electronic systems available at the UoZ, and specially with the available e-learning management system (Moodle). Other future work is to add enhancements to the developed ontology by adding Semantic Web Rule Language (SWRL).

REFERENCES

Abbas, Muhammad Aun. 2016. “A Unified Approach for Dealing with Ontology Mappings and Their Defects.”

Akerman, Art, and Jeff Tyree. 2006. “Using Ontology to Support Development of Software Architectures.” *IBM Systems Journal* 45(4): 813–25.

AL-Zebari, Adel, Subhi R. M. Zeebaree, Karwan Jacksi, and Ali Selamat. 2019. “ELMS–DPU Ontology Visualization with Protégé VOWL and Web VOWL.” *Journal of Advanced Research in Dynamic and Control Systems* Volume 11(01-Special Issue): 478–85.

Bhatia, MPS, Akshi Kumar, and Rohit Beniwal. 2016. “Ontologies for Software Engineering: Past, Present and Future.” *Indian Journal of Science and Technology* 9(9): 1–16.

Corcho, Oscar, Mariano Fernández-López, and Asunción Gómez-Pérez. 2006. “Ontological Engineering: Principles, Methods, Tools and Languages.” In *Ontologies for Software Engineering and Software Technology*, Springer, 1–48.

Fensel, Dieter. 2002. “Ontology-Based Knowledge Management.” *Computer* 35(11): 56–59.

Ganapathi, Gopinath, Ravi Lourdusamy, and Veeraraghavan Rajaram. 2011. “Towards Ontology Development for Teaching Programming Language.”

Gaševic, Dragan, Dragan Djuric, and Vladan Devedžic. 2006. *Model Driven Architecture and Ontology Development*. Springer Science & Business Media.

Happel, Hans-Jörg, and Stefan Seedorf. 2006. “Applications of Ontologies in Software Engineering.” In Citeseer, 5–9.

Jacksi, Karwan, Nazife Dimililer, and Subhi R. M. Zeebaree. 2015. “A Survey of Exploratory Search Systems Based on LOD Resources.” In *PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON COMPUTING & INFORMATICS*, Proceedings of the International Conference

on Computing & Informatics, ed. Jamaludin, Z and ChePa, N and Ishak, WHW and Zaibon, SB. COLL ARTS & SCI, INFOR TECHNOL BLDG, SINTOK, KEDAH 06010, MALAYSIA: UNIV UTARI MALAYSIA-UUM, 501–9.

Jacksi, Karwan, Nazife Dimililer, and Subhi RM Zeebaree. 2016. "State of the Art Exploration Systems for Linked Data: A Review." *International Journal of Advanced Computer Science and Applications (IJACSA)* 7(11): 155–64.

Jacksi, Karwan, Subhi R. M. Zeebaree, and Nazife Dimililer. 2018. "LOD Explorer: Presenting the Web of Data." *International Journal of Advanced Computer Science and Applications (IJACSA)* 9(1). http://thesai.org/Publications/ViewPaper?Volume=9&Issue=1&Code=IJACSA&SerialNo=7 (February 5, 2018).

John, Santhosh. 2010. "Leveraging Traditional Software Engineering Tools to Ontology Engineering under a New Methodology." In IEEE, 1–5.

Kim, Jeong Ah, and Seung Young Choi. 2007. "Evaluation of Ontology Development Methodology with CMM-i." In IEEE, 823–27.

Lee, Ming-Che, Ding Yen Ye, and Tzone I Wang. 2005. "Java Learning Object Ontology." In IEEE, 538–42.

MacIsaac, Bruce. 2003. "An Overview of the RUP as a Process Engineering Platform." In , 26–30.

Pan, Yue et al. 2006. "Model-Driven Ontology Engineering." In *Journal on Data Semantics VII*, Springer, 57–78.

Sosnovsky, Sergey, and Tatiana Gavrilova. 2006. "Development of Educational Ontology for C-Programming."

"Visual Studio 2017 15.7 Release Notes | Microsoft Docs." 2017. https://docs.microsoft.com/en-us/visualstudio/releasenotes/vs2017-relnotes-v15.7 (July 20, 2019).