A Novel Approach to Explore Edhi Foundation Knowledge for Ontology Construction

Muhammad Ahsan Raza[†], M. Rahmah[††], Fahad Qaswar[††], Roslina Abd. Hamid[††] and Sehrish Raza[†††]

[†]Department of Information Technology, Bahauddin Zakariya University, Multan, Pakistan
[††]Faculty of Computing, Universiti Malaysia Pahang, Kuantan, Malaysia
[†††]Institute of Computer Science and Information Technology, The Women University, Multan, Pakistan

Summary
The worth of the Edhi Foundation (EF) system depends on how well its structural knowledge can be extracted by EF users and staff to perform daily activities. Ontology has emerged as a semantic tool to represent the knowledge of a particular domain and thus is a good choice for the semantic organization of EF data. However, building an ontology that suits the needs of EF users is a challenging task. This study presents a novel approach that uses the UML class diagram (UCD) to construct an ontology for the EF system. We propose UCD-to-ontology transformation rules, that is, the ontology model that is used for eliciting OWL ontology. We test our approach for the successful interpretation of UCD features to OWL ontology elements and find that the system performs well with an average precision of 97.80%.

Key words:
Knowledge Management, Ontology Engineering, Ontology Validation, Semantic Web.

1. Introduction

In Pakistan, an increasing number of social welfare organizations are working with the aim of serving humanity and the betterment of their lives. The Edhi Foundation (EF) founded in 1951 has become one of the biggest and well-known nonprofit social welfare service providers in Pakistan [1]. The EF provides many services, such as shelter for deprived people, free hospitals and medical facilities, drug and rehabilitation services, national and international relief efforts, and ambulance service. More than 300 service centers of the foundation are operational across the country, including major cities, small villages, and remote rural areas. The EF knowledge system is evolving with the need to share information to the public and the workers and agents to improve their learning. The three major challenges to this knowledge sharing are as follows:

1) crafting a shared mutual understanding among the different users of the EF system (e.g., the public, government agencies, and staff);
2) organizing, utilizing, and accessing knowledge about human welfare services; and
3) facilitating the interaction between EF centers and users across different welfare services.

To meet these challenges, we consider developing a knowledge structure, namely, an ontology for describing the vocabulary (i.e., concepts, properties and relationship between concepts) of the EF domain. Furthermore, we adopt the OWL language to build an EF ontology because OWL is a W3C standard language that has a high level of semantic expressivity [2]. The new EF OWL ontology provides an unambiguous vocabulary to EF users, thus supporting interoperability among different welfare services, centers, and software agents. Furthermore, with a common ontology, developing an efficient decision support system with reasoning capabilities is possible.
Engineering an EF domain ontology requires accuracy and efficiency. If the ontology refers to an ambiguous relationship, then users do not acquire the necessary knowledge. In addition, accurate ontology development is a tedious and time-consuming task, especially for large systems, such as EF where information evolves over time.
This study focuses on engineering an OWL ontology for the EF domain while considering accuracy and efficiency. We devise a unified modeling language (UML)-based framework for ontology construction that follows four simple steps. In this framework, UML class diagrams (UCD) for the EF domain are created from EF data (i.e., documentation and user interviews) and used to build an ontology. Furthermore, the method outlines the rules to map UML diagrams into ontology vocabulary. The present approach is motivated by the lack of a model to engineer an ontology for the EF domain. Our contributions are as follows:

1) A UML-based approach for developing an ontology used in the EF knowledge system is developed.
2) Important transformation rules that map UML class level (UCL) diagram into ontology vocabulary are outlined while fully maintaining the domain semantics.

The rest of this paper begins with Section 2, which reports a review of related methodologies for ontology engineering. Section 3 presents the steps of the proposed framework to construct the EF ontology. In Section 4, the mapping model for UCD to OWL ontology generation is outlined. Section 5 discusses the results, and Section 6 presents the conclusion and future direction.

2. Literature Survey

Considerable effort has been exerted in developing ontologies (also referred to as ontology engineering approaches) to capture the domain knowledge into an ontological semantic structure [5] [7]. These works could be categorized as staged approaches and metamodel mapping approaches.

2.1 Staged Approaches in Ontology Engineering

Researchers have proposed several ways to construct an ontology from scratch [9] [18]. These methods differed in data collection procedures, ontology application domains, ontology structuring strategies (e.g., handling taxonomic or nontaxonomic relations, properties, and constraints), and ontology languages. In [6], the authors proposed a staged approach for building an ontology in the domain of software architecture. The approach was question-based, that is, questions were acquired for various sources, such as mail data, interview, or log files. The identified questions were then analyzed to capture the structure and semantics of architecture knowledge into ontology. The authors in [10] claimed that ontology could be a valuable resource for the accurate development of a chemical engineering curriculum. The approach constructed a chemical engineering ontology by identifying the curriculum topic classification (i.e., topic and its taxonomy extraction). After taxonomic relations, other properties (attributes ad relationships) relevant to the topic were defined. The method, which demonstrated remarkable results, evaluated the generated ontology using a semantic reasoner and a case study.
In another example, an ontology for data on daily life activities was constructed using the OBO-Edit ontology editor [17]. To construct an ontology, the procedure first identified concepts and the hierarchy between the concepts from data on daily life activities. The initial ontology was then extended in the physiological context that represents facets, such as activity time, involved object in the activity, involved agent, and subactivity properties.

2.2 Metamodel for Ontology Mapping Approaches

A different kind of research approach toward ontology building was based on using a metamodel (e.g., relational database, entity relationship diagram, software engineering model). The metamodel reflected domain knowledge that could be converted into an ontological knowledge structure. Various researchers have exploited metamodeling techniques in state-of-the-art ontological engineering strategies [3] [4] [8].
The authors in [15] suggested a simple approach to convert a relational database (RDB) model to an OWL ontology. They proposed the mapping rules for RDB-to-ontology transformation, which differs from previous models in terms of subdata properties and subclass conversion. The results showed that the approach performed well in small and large databases. In recent studies, [16] proposed an idea to generate an ontology from extended entity relationship diagram (ERD) schema. With mapping information for the translation of ERD elements to OWL constructs, the model appeared to be effective in different practical scenarios.
Many researchers have focused on using UML diagrams (i.e., approaches related to this study) for ontology generation [13] [19]. In [11], authors generated RDF ontologies via UML modeling to support experts who are less familiar with the ontology structure. The system demo considered user input in the ontology construction to include unmapped terms. However, no detail about UML to RDF mapping rules or conversion processes was discussed by authors. Another similar approach was presented by [14], where an ontology was created from UCL diagrams to support e-learning. The study proposed UML-to-OWL ontology conversion but did not focus on UML mapping constraints, such as multiplicity of relationships, null value restrictions, and class disjointness.
In the present study, we presented a refined ontological engineering approach that combines the staged development method (to enable ontology construction from scratch) and the UML model (to obtain accurate domain knowledge). We outlined a comprehensive set of mapping rules to facilitate the transition of the UCD to an OWL ontology while maintaining all semantic facets.

3. Framework for EF Ontology Construction

To develop an EF OWL ontology, this study proposes a framework that utilizes software engineering modeling (as depicted in Figure 1). Our approach consists of four phases: feasibility study, conceptual model, ontology model, and evaluation. Other activities, such as knowledge acquisition of the EF domain (performed in parallel in the first two phases), use of a software engineering model (spans throughout phases 2 and 3), and EF ontology validation (conducted in the last step), facilitate the successful execution of the proposed framework.

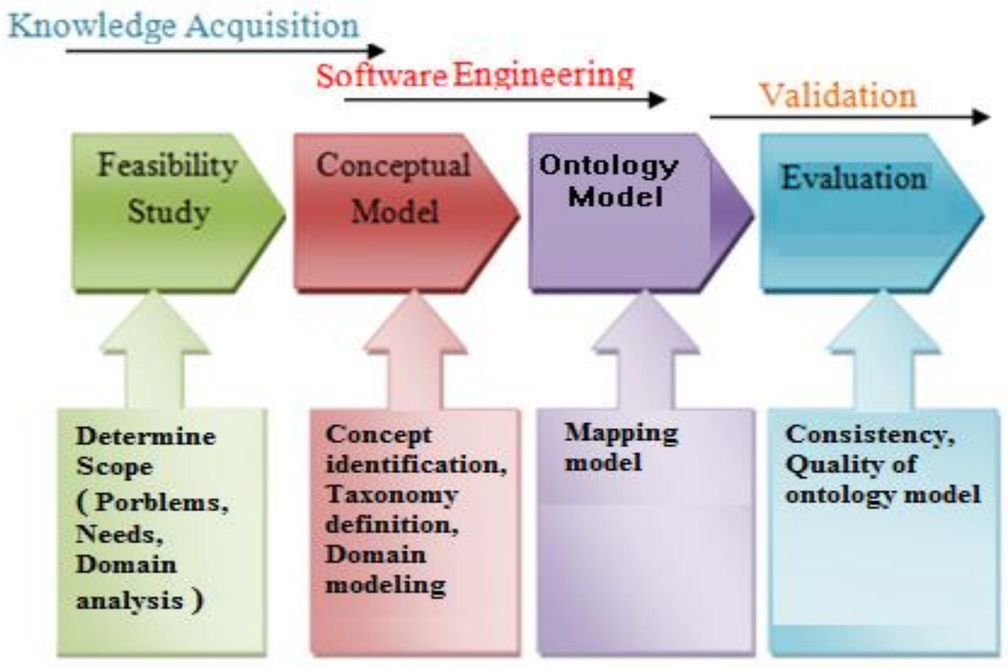

Knowledge Acquisition
Software Engineering
Validation
Feasibility Study
Conceptual Model
Ontology Model
Evaluation
Determine Scope (Porblems, Needs, Domain analysis)
Concept identification, Taxonomy definition, Domain modeling
Mapping model
Consistency, Quality of ontology model

Fig. 1 Steps of EF ontology construction framework.

3.1 Feasibility Study

The EF knowledge system involves numerous centers that handle various social welfare services; thus, the first step of our approach is to identify existing EF problems and needs. This information is acquired by investigating EF documentation and interviewing users. A representative sample of users for each EF service centers is interviewed about their daily activities. For example, Edhi home service staff members are asked about the supply of food and clothing to poor persons, room management, and health arrangements.

Based on the data about problems and needs, the feasibility study determines the scope of the EF ontology, that is, why ontology is being developed (i.e., purpose), what should be added or excluded from the ontology (i.e., the conceptualization of domain), and the type of information the ontology should provide (useful for evaluation). This phase ends with a feasibility document that contains requirement specifications and other features needed to build the ontology.

3.2 Conceptual Model

In this phase, the EF concepts (words or phrases) are identified from the data collected in the previous step. Furthermore, the concepts are classified to represent the ontology’s vocabulary, such as class and properties (attribute and relationship). The conceptual model phase is implemented in three steps as follows:

1) Concept Identification

The list of concepts that describe the subjects (i.e., nouns) is identified and further clarified with the EF users. For instance, Edhi_homes, Edhi_services, Ambulance, Charitable_shop, and Children_services are recognized as key concepts of the EF.

2) Properties and Hierarchy Definition

This step defines the attributes and properties (relationships with other concepts) of the identified concepts. The individual concept with defined properties is now called a class. Moreover, the list of classes is organized into a taxonomy (e.g., the ambulance class is categorized as a subclass of Edhi_service class). The authors in [6] have identified three approaches to define the classification: (1) a top-down model that starts with the definition of the most general class and then specifies the subclasses; (2) a bottom-up model that is an inverse of the first model, where the most specific class is described first; (3) a hybrid model that combines the properties of the top-down and bottom-up models. To build the EF taxonomy, we focus on the hybrid classification model.

3) Domain Modeling

Modeling is necessary to identify the interaction and behavior of identified classes. From existing models [5], we select a UML model for two reasons: (1) UML is a well-established model in the software engineering field. In addition, UML diagrams are readily available and require minimal guidance from software project experts. Therefore, these diagrams can be created accurately and efficiently for any domain of interest. (2) UML is similar to an ontology structure in terms of scope (i.e., UML components are relevant to OWL ontology vocabulary). In particular, we focus on the class level diagram in UML to model the EF domain taxonomy, concept behavior, and interaction between concepts. For instance, Figure 2 depicts the hierarchical relationship of the Edhi_Foundation class with its subclasses (e.g., Edhi_service and Edhi_centres) in UML.

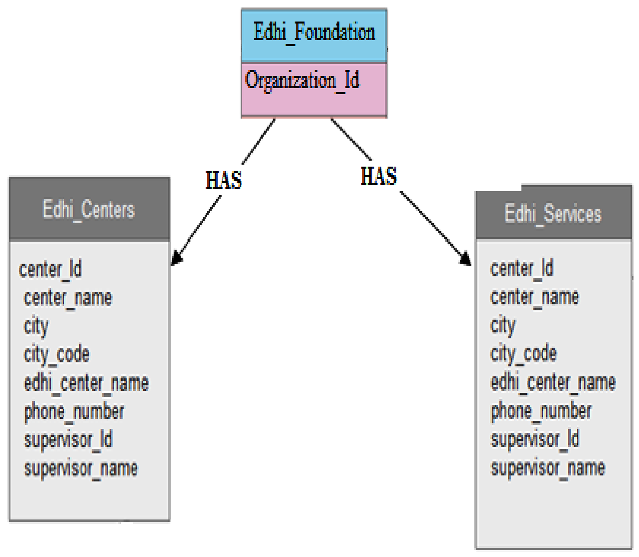


Fig. 2 UML class hierarchy diagram

3.3 Ontology Model

One important part of the EF ontology engineering approach is to map the already crafted UCD (i.e., the output of domain modeling step) into the ontology vocabulary. In the ontology model phase, we define the rules for transforming each aspect of the UCD (i.e., class; relationships, such as dependency and association; and participation) to the corresponding component of the ontology. These mapping rules are called collectively as an ontology model, which is discussed in Section 4.

3.4 Evaluation

The EF ontology (generated after implementing the rules of the ontology model) can be evaluated in two different parameters: the accuracy of the ontology schema (consistency parameter) and the effectiveness of the rules in the ontology model (quality parameter).

1) Consistency Metric

Various semantic reasoners (such as Pallet, HerMiT) for inferring the logical consequences within the ontology vocabulary are available [12]. An automated reasoner derives mismatches within the ontology taxonomy, thereby indicating the clarity and consistency of the newly generated ontology.

2) Quality Metric

To address the quality of the proposed ontology model (i.e., mapping rules) in the transformation of UML to ontology, we adopt a precision ratio, which is a widely accepted measure (See Section 5).

4. Ontology Modeling

We develop an intuitive ontology model that describes the rules to transform the UCD element to an OWL ontology vocabulary because of the component similarity between the UCD and the ontology (i.e., both use classes and relationships). In this section, we first describe the UCD model of the EF domain. Second, we discuss the proposed ontology model (i.e., UCD to OWL transformation) to create the EF OWL ontology.

4.1 EF Conceptualization

The proposed framework uses UCD to represent the EF domain data. For UCD formalization, we only focus on the core features of UCD that are necessary to express the ontological knowledge. Table 1 outlines the UCD features and associated EF conceptual data (i.e., total number classes, corresponding attributes, relationships, and constraints). Edhi services and centers are recognized as two major classes of the EF. The Edhi services class is divided into 20 subclasses, such as ambulance, Edhi education, and hospital classes. The Edhi centers class consists of five subclasses, e.g., Sindh province class and

KPK province class. These identified subclasses are further extended into 63 child classes. Furthermore, the classes contain attributes and are linked with others using relationships.

4.2 Ontology Model: Introducing Transformation Rules

The ontology model proposes rules for transforming the EF UCD syntax to an EF OWL ontology. Table 2 lists these transformation rules, whereby each UML element (also represented in symbolic form) is mapped to the OWL vocabulary (given as OWL syntax) by applying a mapping rule. For instance, the UCD class is mapped to the OWL class, the class attribute is translated to an OWL datatype property, and the UCD relationship is interpreted as an OWL object property. In addition, the UCD cardinality restriction (for attribute and relationship) is also mapped to the OWL restriction by setting min-cardinality or max-cardinality construct.

Table 1: EF Conceptual Data in Terms of UCD Elements

	UCD features	*Examples*	*Total number*
1.	Classes	Edhi foundation, Edhi services, Edhi centers, Provinces, Ambulance, Children services, Edhi homes, Orphanages, Edhi maurge, Edhi rikshaw ROZGAR, Educational services	90

2.	Attributes	Edhi foundation (regno, regName);
		Edhi services (serviceID, Title, dateOffered, purpose);
		Edhi center (centerID, Cname, Phone)
			215

3.	Types of relationships	ISA, Existence dependency, composition, aggregation, association	5
4.	Types of constraints	Unique, Disjoint, cardinality	3

5. Implementation and Result Discussion

Our approach used the EF system UCD and the proposed ontology model to generate an OWL ontology for the EF knowledge system. We developed a prototype using the Protégé ontology editor as a proof of concept. Protégé facilitates UCD-to-OWL ontology translation via effortless and quick (using graphic user interface) implementation of ontology model rules and reduces the cost of ontology development because it is a free open-source platform. Other plug-ins, such as the OntoViz tool and the Pallet semantic reasoning engine, were respectively used for the visualization and evaluation of the newly created EF ontology. Our system prototype took the UCD file of the EF system as input, parsed it according to the rules of the ontology model, and constructed the OWL ontology. Figure 3 from the OntoViz plug-in illustrates the structure of the resulting EF ontology.

To validate the structure of the final EF ontology, we relied on the semantic reasoner test and the quality checking of UCD-to-ontology conversion on the basis of the ontology model's rules (as mentioned in Section III-D). The semantic reasoner was a good choice to assess the accuracy (including class duplication, class and subclass taxonomy, ontology consistency) of the new ontology according to the defined conceptualization. To this end, we used the Pallet logic reasoner (a Protégé plug-in) and found a consistency of 100% among the components of the resultant EF OWL ontology.

To estimate the successful implementation of the ontology model rules via system prototype, we relied on the opinions of experts to assess whether UML elements are precisely mapped to corresponding OWL ontology components. Two groups of experts (each comprising 12 computer science research students and faculty members) were given three types of files: (1) UCD file, (2) ontology model, and (3) newly generated OWL ontology file. Each group evaluated the OWL file in terms of class mapping, taxonomy mapping, property (data type and object) mapping, and constraint (cardinality and disjoint restriction) mapping while considering the EF UCD document. In addition, we shuffled the perceived evaluation of each group with other group to achieve accurate results.

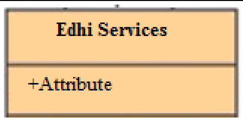
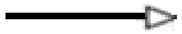
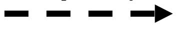
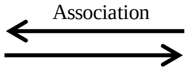
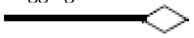
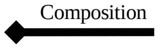
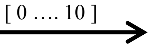
On the basis of the assessment data of the experts, we manually calculated the precision measure using Eq. 1. The precision metric was adopted to indicate the effectiveness of our method for the accurate conversion of UCD elements to OWL ontology vocabulary.

$$\text{Precision} = \frac{\text{Valid number of V transformation}}{\text{Total number of V}}, \qquad (1)$$

where V represents UCD features, such as class, attribute, relationship, or cardinality.

The precision measure was tested for five OWL facets in EF ontology (i.e., class, subclass of relationship, data type property, object property, and restrictions on properties). Figure 4 presents the precision statistics of these OWL facets in the form of a column chart. Overall, the proposed method attained 97.20% value for average precision during the EF ontology building process.

Table 2: Ontology Model Representing UML to Ontology Mapping

UML element	UML symbol	Mapping rules	OWL ontology example
Class: Represent set structure	 (e.g., UML Edhi services class)	- Map UML class to class entity in ontology	<owl:Class rdf:about="#Edhi_Services" > <rdfs:comment> this class contain all instances of Edhi services </rdfs:comment> </owl:Class>
Attribute: Represent property of class	+AttributeName : Typename [*constraint*] (e.g., city is attribute of UML Edhi Center class)	- Map UML attribute to Data-type Property in ontology, where domain of property is set to class and range of property is set to XML data type.	<owl:DatatypeProperty rdf:ID ="city"> <owl:domain rdf:resource ="#Edhi_Center"/> <rdfs:range rdf:resource = "http://www.w3.org/2000/01/rdf-schema#string"/> </owl:DatatypeProperty>
Attribute Constraints: Represent restrictions on property of class	Null [*]	-Min-cardinality is set to zero for Data type property. (e.g., center_comment attribute of class may contain null value)	<owl:minCardinality rdf:datatype = "&xsd; Integer"> 0 </owl:minCardinality>
	Not Null [1..5]	- Min-cardinality is set to one for Data type property. - Max-cardinality is set to two for Data type property.	<owl:minCardinality rdf:datatype = "&xsd;Integer"> 1 </owl:minCardinality> <owl:maxCardinality rdf:datatype = "&xsd; Integer"> 5 </owl:minCardinality>
	Unique [1]	- Map to Functional and inverse Functional Data-type Properties. (e.g., center_id attribute of Edhi_Cetners class must have unique value)	<owl:FunctionalProperty rdf:about ="#center_id" /> <owl:InverseFunctionalProperty rdf:about ="# center_id"/>
Generalization: Represent hierarchical relationship between classes	Generalization  (e.g., UML Edhi services class is child class of UML Ehdi foundation class)	- Map Edhi services class as SubclassOf construct to Edhi foundation class in ontology.	<owl:Class rdf:about ="#Edhi_Services "> <rdfs: comment> Edhi Services is the sub-class of Edhi Foundation. Which shows inheritance</rdfs:comment> <rdfs:subClassOf rdf:resource ="# Edhi_Foundation"> </owl:Class>
Dependency relationship: A class existence depends on other class.	Dependency  (e.g., UML Ambulance class existence is dependent on Edhi services class)	- Map to object property (e.g., Include) by setting domain and range to a class in ontology. - Include FunctionalProperty because Ambulance class must have a unique individual against Edhi service class.	<owl:ObjectProperty rdf:ID =" Include "> <rdf:type rdf:resource ="&owl;FunctionalProperty" /> <rdfs:domain rdf:resource ="# Edhi_Services " /> <rdfs:range rdf:resource ="# Ambulance" /> </owl:ObjectProperty>
Association Relationship A class is seen in two roles with associated other class.	Association  (e.g., UML Edhi centers class has two roles with UML Edhi services class)	- Map to two Object Properties in ontology, where one object property is inverse of other object property.	<owl:ObjectProperty rdf:ID ="Offers"> <rdfs:domain rdf:resource ="# Edhi_Centers "/> <rdfs:range rdf:resource ="# Edhi_Services "/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID ="OfferedBy"> <owl:inverseOf rdf:resource ="#Offers"/> </owl:ObjectProperty>
Aggregation relationship It is a sort of association relationship, representing Is_Part relationship	Aggregation  (e.g., UML Sakhar center class is a part of UML Sindh center class)	- Map to object property (e.g., Center_Part) by setting domain and range to a class in ontology.	<owl:ObjectProperty rdf:ID ="Center_Part"> <rdfs:domain rdf:resource ="#Sakhar_Center"/> <rdfs:range rdf:resource ="#Sindh_Center "/> </owl:ObjectProperty>
Composition relationship: It is inverse of aggregation, representing Is_Whole relationship	Composition  (e.g., UML Sindh center class represent a whole of UML Sakhar center class)	- Map object property as inverse of aggregation property.	<owl:ObjectProperty rdf:ID ="Whole_Center"> <owl:inverseOf rdf:resource ="#Center_Part"/> </owl:ObjectProperty>
Multiplicity of Relationships [low….high] :	[0 …. 10]  (e.g., 0 to 10 multiplicity exist for Offer association relationship	- Map to minimum and maximum cardinalities of object property in ontology.	<owl:minCardinality rdf:datatype = "&xsd; Integer"> 0 </owl:minCardinality> <owl:maxCardinality rdf:datatype = "&xsd; Integer">

Represent cardinality restriction of relationships	between UML Edhi services class and UML Ambulance class		10 </owl:minCardinality>
Disjoint Constraint: Represent that two classes member are distinct.	{ disjoint } (e.g., Edhi services class and Ehdi centers class are disjoint)	- Map to disjointWith constructor in ontology	<owl:Class rdf:about ="#Edhi_Services "> <owl:disjointWith rdf:resource = "#Edhi_Centers"/> </owl:Class>

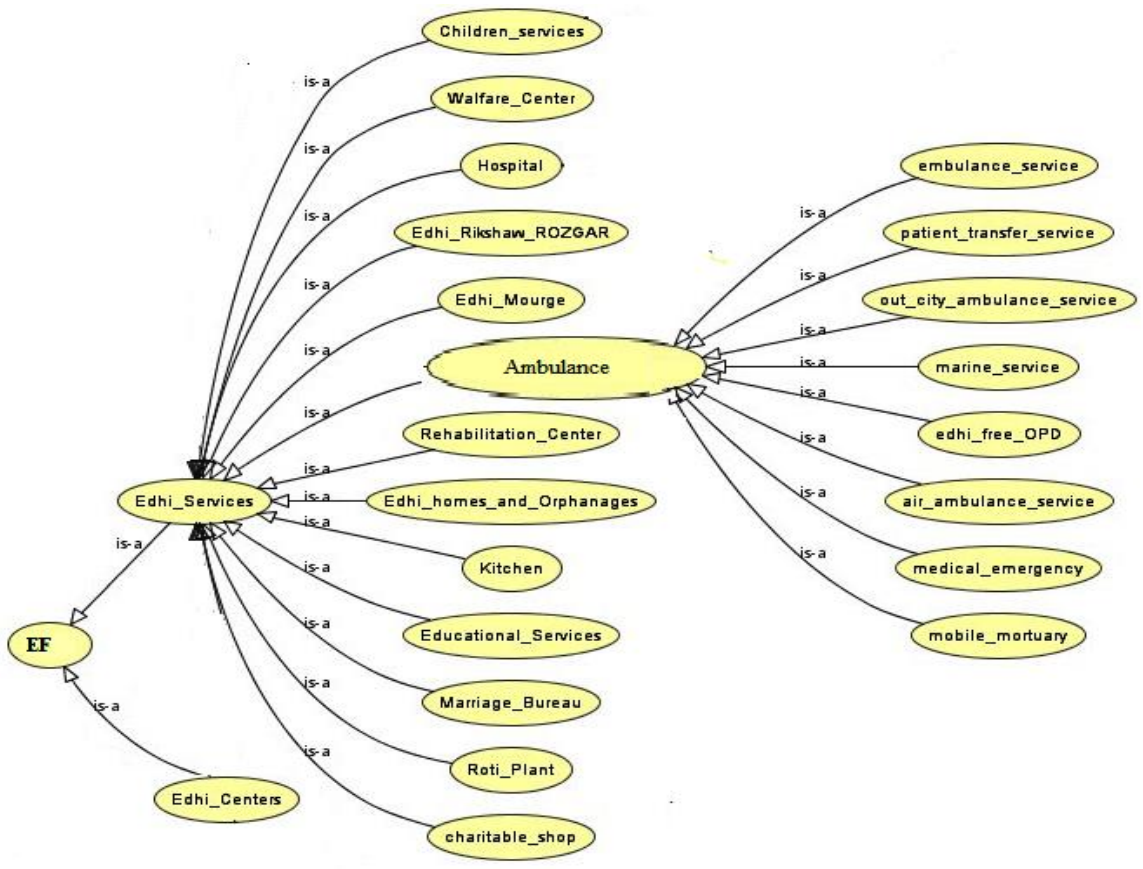


Fig. 3 Snapshot of EF ontology from OntoViz

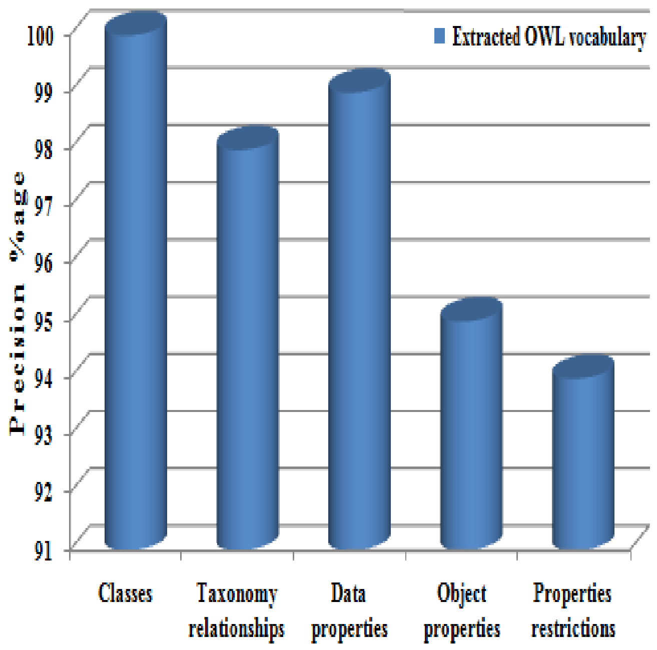


Fig. 4 Precision values for five OWL facets

6. Conclusion and Future Work

The use of ontology for the EF system can improve knowledge representation and acquisition. For an ontology-oriented EF system, EF ontology must be constructed in consideration of the domain semantics and different needs of individual EF users. However, manual ontology creation for the EF knowledge system is challenging because EF is a large organization that has a diverse number of users. This study aims to describe a manual ontology engineering approach in the context of large and complex systems, such as EF organization. Our approach uses the UCD document that is created from EF documentation and user input. EF semantic vocabulary is then identified from the UCD document and modeled in OWL ontology.
We devise an ontology model that outlines the rules for transforming the UCD elements into OWL constructs. For instance, the UCS class is translated to an OWL class, the UCD class hierarchy becomes OWL sub-class relationships, and the UCD class attributes are converted to OWL properties of the appropriate class. UML cardinality mapping is also defined to construct a precise ontology for the EF system. We find that the use of UCD in ontology construction is beneficial: (1) UML is a well-established modeling language, (2) building a UCD document is quick because experts are readily available, and (3) the class level diagram of UML shares a semantic similarity with OWL ontology language in terms of components.
Our approach caters for the consistency test of the resultant EF ontology. We use a Pallet semantic reasoning engine that shows no traces of violation (such as class duplication, problem in class hierarchy assertion, or misinterpretation of properties) in new OWL ontology vocabulary. Moreover, we evaluate the proposed ontology model rules that influence the successful implementation of our system prototype. Experts assess the EF ontology generated by the prototype and confirm that it correctly represents all the semantics of the EF UCD. These findings show that our approach is suitable for EF knowledge gathering and its accurate explicit representation in the form of EF ontology. In the future, we plan to improve our approach for handling the UCD ternary association in ontology construction using the reification phenomena.

References

[1] M. Saba, A. Noor, and S. Malik, “An Analysis into the Spatial Distribution of Trauma Incidents and Ambulance Functionalities in Karachi,” Journal of Space Technology, vol. 7, no. 1, 2017.
[2] M. Fahad, "ER2OWL: Generating OWL Ontology from ER Diagram," Intelligent Information Processing IV. pp. 28-37.
[3] Z. Xu, Y. Ni, W. He, L. Lin, and Q. Yan, “Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach,” World Wide Web, vol. 15, no. 5, pp. 517-545, September 01, 2012.
[4] M. A. Raza, M. Rahmah, S. Raza, A. Noraziah, and R. A. Hamid, “A Methodology for Engineering Domain Ontology using Entity Relationship Model,” International Journal of Advanced Computer Science and Applications(IJACSA), vol. 10, no. 8, 2019.
[5] R. Iqbal, M. A. A. Murad, A. Mustapha, and N. M. Sharef, “An analysis of ontology engineering methodologies: a literature review,” Research Journal of Applied Sciences, Engineering and Technology, vol. 6 (16) pp. 2993-3000, 2013.
[6] K. A. de Graaf, P. Liang, A. Tang, W. R. van Hage, and H. van Vliet, “An exploratory study on ontology engineering for software architecture documentation,” Computers in Industry, vol. 65, no. 7, pp. 1053-1064, 2014/09/01/, 2014.
[7] M. S. Rahman, and G. Rabby, “Design and Development of a University Human Resource Ontology Model for Semantic Web,” International Journal of Computer Science and Network Security, vol. 17, no. 1, pp. 187-192, 2017.
[8] M. A. Raza, and B. Raza, “Comparative Analysis of Ontology Extraction Techniques from Relational Database,” SCIENCE INTERNATIONAL, vol. 4, pp. 3589-3595, 2016.
[9] Y. Zeng, J. Zhuang, and Z. Su, "Construction of Domain Ontology for Engineering Equipment Maintenance Support," Knowledge Graph and Semantic Computing:

Semantic, Knowledge, and Linked Big Data. 2016, pp. 33-38.

[10] M. Bussemaker, N. Trokanas, and F. Cecelja, “An ontological approach to chemical engineering curriculum development,” Computers & Chemical Engineering, vol. 106, pp. 927-941, 2017/11/02/, 2017.

[11] D. De Paepe, G. Thijs, R. Buyle, R. Verborgh, and E. Mannens, "Automated UML-Based Ontology Generation in OSLO2," The Semantic Web: ESWC 2017 Satellite Events. pp. 93-97.

[12] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, and A. Steigmiller, “The OWL Reasoner Evaluation (ORE) 2015 Competition Report,” Journal of automated reasoning, vol. 59, no. 4, pp. 455-482, 2017.

[13] N. Hong-Seok, O. H. Choi, and L. Jung-Eun, "A Method for Building Domain Ontologies based on the Transformation of UML Models." pp. 332-338.

[14] B. Bouihi, and M. Bahaj, “Moodle's Ontology Development from UML for Social Learning Network Analysis,” in Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, Rabat, Morocco, 2018, pp. 1-6.

[15] T. Naz, M. Shuja, S. K. Shahzad, and M. Atif, “Fully automatic OWL generator from RDB schema,” International Journal of Advanced and Applied Sciences vol. 5(4), pp. 79-86, 2018.

[16] Z. Telnarova, "Transformation of Extended Entity Relationship Model into Ontology," Intelligent Information and Database Systems. pp. 256-264.

[17] P. R. Woznowski, E. L. Tonkin, and P. A. Flach, “Activities of Daily Living Ontology for Ubiquitous Systems: Development and Evaluation,” Sensors, vol. 18(7), 2361, 2018.

[18] Y. E. ALLIOU, and O. E. BEQQALI, “O’Neurolog – Building an Ontology for Neurology in Mobile Environment,” International Journal of Computer Science and Network Security, vol. 12, no. 7, pp. 91-100, 2012.

[19] J. Bakkas, and M. Bahaj, “Automatic Conversion Method of Class Diagrams to Ontologies Maintaining Their Semantic Features,” International Journal of Soft Computing & Engineering vol. 2, no. 6, 2013