An Ontology Design Pattern Recommendation System

Maleeha Arif Yasvi

MT18112

Under the Supervision of **Dr. V. Raghava Mutharaju**

Submitted
in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science & Engineering

Indraprastha Institute of Information Technology, New Delhi
July 2020

© Indraprastha Institute of Information Technology (IIITD), New Delhi 2020

Certificate

This is to certify that the thesis titled **"An Ontology Design Pattern Recommendation System"** being submitted by **Maleeha Arif Yasvi** to Indraprastha Institute of Information Technology, Delhi for the award of *Master of Technology* in *Computer Science & Engineering* is an original research work carried out by her under my guidance and supervision.

The results contained in this thesis have not been submitted anywhere else for the reward of any other degree/diploma.

July, 2020

Dr. V. Raghava Mutharaju

Department of CSE

Indraprastha Institute of Information Technology

New Delhi 110 020

Acknowledgment

I would like to express my deepest gratitude to **Dr. V. Raghava Mutharaju**, Assistant Professor, Department of CSE, IIIT Delhi. His patience, valuable opinion and encouragement were key motivations throughout my study. His steadfast integrity and selfless dedication from the initial to final level enabled me to develop better understanding of the subject. It has been an honour and great pleasure for me to work under his kind and able guidance.

I would like to extend my gratitude to my colleagues from KRACR lab for their constant support.

I would also like to thank my family and friends for encouraging, supporting and showing belief in me and my work.

Abstract

Ontologies evolve over time due to changes in the domain and the requirements of the application. Maintaining an ontology over time and keeping it up-to-date with respect to the changes in the domain and the requirements of application is hard. But a high quality ontology can significantly reduce the effort and the cost of ontology maintenance. Ontology Design Patterns (ODPs) can be used to improve the quality of an ontology and make it more modular and reusable. But with increasing number of ODPs spread across different categories, it is not easy to determine the right set of ODPs to choose for a particular use case even for experts. This becomes even more difficult in the case of refactoring existing ontologies using the right set of ODPs. We describe here a tool named ODPReco that can recommend the possible ODPs to use in a given ontology by analyzing the lexical, structural, and behavioural aspects of the given ontology. ODPReco is open-sourced and is publicly available at `https://github.com/kracr/ODPReco`.

Keywords: Ontology Design Pattern, Refactoring Ontologies, Modular Ontologies, Recommendations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ontologies are a means to model the structure of a system [12]. They provide description of entities and their relations with each other. An example of such a system can be the different types of food items [5]. Ontology Design Patterns (ODPs) are the reusable small blocks of ontology [11]. They are defined as light-weight, modularized versions of design principles that serve for good modelling practices. Examples of ODPs include event pattern which can be used in ontologies for planned occasions, for example annual function at school. Another example of ODP is information realisation pattern [19]. It describes the relation between an information object and its physical realisation. For example book is the object and the paper copy of that book is its physical realisation.

Ontologies are the building block of semantic web technology. Main purpose of this technology is to provide structure to the web. It focuses on capturing knowledge which is machine-processable and can be used for the purpose of semantic sharing across different systems. Its primary goal is to build models, that is, to describe knowledge in terms which are easy to understand, computing inferences from those models and exchanging information which help in inter-linking of the knowledge.

1.1 Background

Ontologies convey basic concepts related to a domain. They define the things that exist in a domain and the relationships that hold between them. The entities in an ontology are represented through classes and the relationships between these classes are represented through properties. For Example, employee and organisation can be a class and *works for* is the relation between the two, that is, an employee works for an organisation. Classes describe the concept of the domain. The instances of these classes are called individuals. For Example, Mary (is an employee).

Ontologies are built to share common goals in a particular domain among different people.

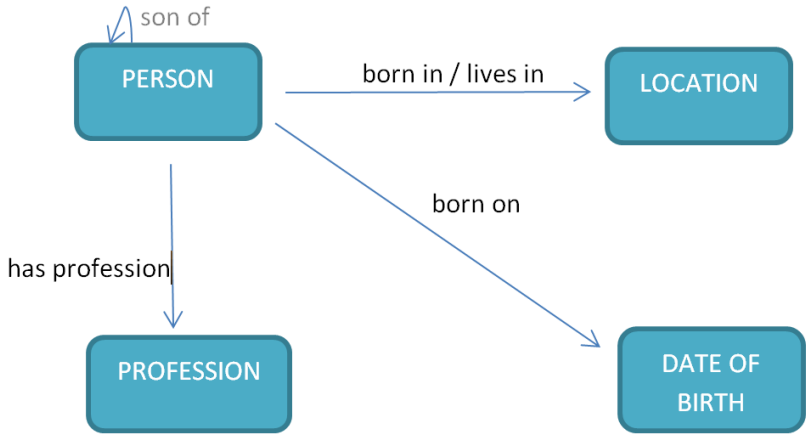

Figure 1.1: Basic ontology of an individual

For example, several websites contain medical information. If their ontology is built, then the information can be aggregated together and can be used to answer different queries or can be given as input to other applications [13]. They also help in analysis of the domain information for extending it or re-using it with some other ontology.

In practical terms, ontology development includes defining classes in the ontology, the sub-classes and describing the relationships between the classes [13].

Consider an ontology of a person. We first identify the classes that can be used in this ontology. We consider only four classes for this example and they are Person, Profession, Location and Date of birth. Then, we identify the relationships that exist between these classes. Some of the properties identified are as-

<Person> <born on> <Date of birth>.

<Person> <born in> <Location>.

<Person> <has profession> <Profession>.

<Person> <lives in> <Location>.

<Person> <son of> <Person>.

Figure 1.1 shows the graphical representation of this ontology.

1.1.1 Protégé

Ontologies are built using protégé. It is an open-source editor used to build all types of ontologies. The main components for ontology building are its classes and properties. Protégé defines different types of properties like subclasses, object, data properties.

OWL (Web Ontology Language) is the knowledge representation language used to build

ontologies. Three types of entities are associated with OWL. These include the classes, properties and individuals. All the entities are identified by using IRIs (international resource identifier). These are similar to URIs except that encoding of the identifier is universal character set. The classes follow upper-camel case pattern in which the first letter is capital while as properties and individuals follow lower-camel case pattern in which first word letter in a compound word excluding the first word is capitalized.

For example, an ontology of school can be built in which classes include the department, people, course and extracurricular activity.

Basic statements involving the entities are referred as *axioms*. Some axioms that are commonly used are as

1. Sub Class: This axiom declares a subclass relation between two classes. In our school ontology, department can have sub-classes like primary, middle, secondary department. Similarly, people can have have sub-classes as teacher, staff and student. Course can also have sub-classes of different subjects taught in a school. Extracurricular activity can include sub-classes of different clubs present in the school and also activities like camping, swimming etc. Some examples of this axiom are-

   SubClassOf(primary Department)

   SubClassOf(Secondary Department)

2. Object Property: The information about two classes or individuals is connected by a certain property and this property is referred as the object property. It has a specific domain and range. In school ontology, object properties that hold between classes include teaches, studiesIn, worksFor, participatesIn. These relations are written in the form of ObjectPropertyDomain and ObjectPropertyRange.

   <Teacher> <teaches> <Student>. is classified into domain and range as-

   ObjectPropertyDomain(teaches Teacher)

   ObjectPropertyRange(teaches Student)

   <Student> <studies in> <Department>. has axioms

   ObjectPropertyDomain(studiesIn Student)

   ObjectPropertyRange(studiesIn Department)

   <Student> <participates in> <Extracurricular Activity>. has axioms

   ObjectPropertyDomain(participatesIn Student)

   ObjectPropertyRange(participatesIn Extracurricular Activity)

   <Teacher> <works for> <Department>. has axioms

   ObjectPropertyDomain(worksFor Teacher)

   ObjectPropertyRange(worksFor Department)

3. Data Property: These properties relate the classes with data types. Each data property has a domain and range. In school ontology, data properties present include hasName, hasRollNo, etc.

   <Student> <has name> <String>. has domain and range as-

   DataPropertyDomain(hasName Student)

   DataPropertyRange(hasName String)

   <Student> <has rollno> <Integer>. has domain and range as-

   DataPropertyDomain(hasRollNo Student)

   DataPropertyRange(hasRollNo Integer)

4. Disjoint Classes: This property represents the classes which have nothing in common. In our ontology, classes course and people are completely different from each other and can be associated as disjoint classes. This axiom is writtten as-

   DisjointClasses(Course, People)

Other axioms present include sub-property , equivalent classes, role-chains.

Sub-class axiom also have property restrictions like AllValuesFrom (universal quantifier), SomeValuesFrom (existential quantifier), IntersectionOf, UnionOf and cardinality restrictions. In our ontology, an example of property restriction is

SubClassOf(ObjectUnionOf(Primary Middle Secondary) Department)

This axiom states that the department can either be primary, middle or secondary.

Figure 1.2 represents the structure of classes and sub-classes in protégé and Figure 1.3 represents the object property *works for*. This property forms the relation between teacher and student. So, teacher is the *domain* and student is the *range*. Other object and data properties can also be represented in this way.

1.1.2 Doc2Vec

Doc2Vec is also referred to as paragraph vectors. It is a type of word embedding which is used to represent numeric representation of a document. We are discussing about doc2vec because we have used it in our implementation. Doc2Vec is based on Word2Vec.

Word2Vec

Word2Vec is used to obtain the embeddings. Embeddings are obtained using CBOW (continuous bag of words) or Skip-gram model.

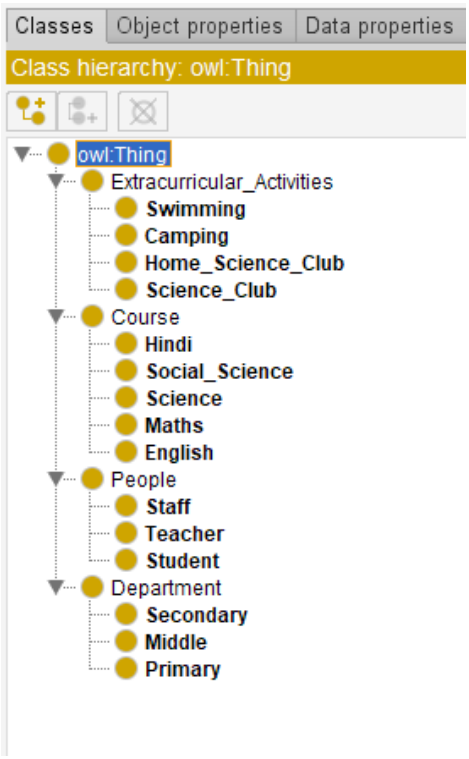

Figure 1.2: Classes hierarchy in protégé

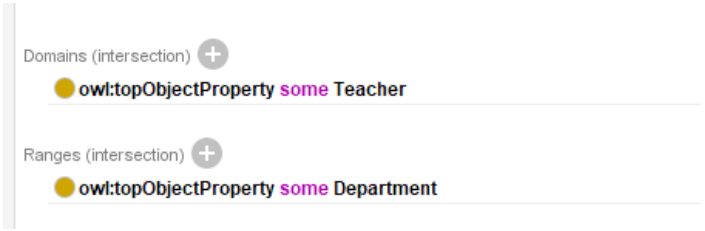

Figure 1.3: Object property representation in protégé

1. CBOW. It predicts the probability of a word given a context. Context refers to a single word or a group of words. It is a neural network and the basic structure of CBOW model has input layer, hidden layer and an output layer.

   For a single context word, we have the input layer with its one-hot encoding of size [1*V] where V=total words present.

   Consider a sentence, *This is sample example*

   If we choose context word as "this", then its input vector is [1 0 0 0].

   Input vectors dot product is done with the weight matrix to form the input hidden weight. Two set of weights are present, one is between the input and hidden layer and the other between hidden and output layer. Hidden layer performs mathematical functions to produce output that is relevant to the intended result. The hidden-input layer size is V*N where N is the number of dimension we choose to represent our word in. There is no activation function between any layers (it is linear).

   Consider N= 4 in our example. So, the matrix of input-hidden weight obtained is of size 4*4. This input-hidden weight is multiplied with the input layer to obtain hidden activation.

   Since, in our example we have taken only one word as context word, so on matrix multiplication of the two, we obtain size of (1*N) as our hidden activation. If we choose more than one word as context word, then the hidden activation is (n*N)

where n is number of context words chosen. After obtaining them, their average is found such that the hidden activation is of size (1*N). The hidden input gets multiplied with hidden output weights and output is calculated. Error between output and target is calculated and back-propagated to readjust the weights. The output layer is a softmax layer which sums the probabilities obtained in output layer to 1.

The loss function in CBOW is negative log likelihood (L(y)=-log(y)). It is used to depict how well CBOW models the data. So, in it, whenever the network assigns high confidence at the correct class, the loss is low and whenever the network assigns low confidence at the correct class, the loss is high.

After calculating the loss, it is back propagated and the weights are updated using gradient.

2. Skip-gram model. Its aim is to predict the context when a word is provided. Given a specific word in the input sentence, it looks at the words nearby. It gives out the probability of every word in our vocabulary with the input word.

   For example, if we give an input word "Delhi", then output probabilities are going to be higher for "India" and "Capital"than for unrelated words like "fruits" and "vegetables".

   The input vector of skip-gram is similar to 1 context CBOW model and the calculations up to hidden layer activation are same, that is, there is one hidden layer performing the matrix multiplication between the weight matrix and the input vector. This result is given to the output layer which computes the product between the output vector of the hidden layer and the weight matrix of the output layer.

   The loss function used with skip-gram model is same as CBOW's, that is, negative log likelihood.

By default, CBOW is used for implementation as it is faster and good for frequently occurring words.

The concept of word2vec is used in doc2vec. Doc2Vec model is based on word2vec. It adds document vector (unique identifier/label) to a document. So, when training the word vectors, the document vector is trained as well and in the end of the training it holds the numeric representation of the document.

Two methods followed in Doc2Vec are distributed memory model of paragraph vectors (PV-DM) and distributed bag of words version of paragraph vectors (PV-DBOW). PV-DM is similar to CBOW approach and the PV-DBOW is similar to skip-gram approach of word2vec.

For implementing doc2vec in java, dl4j library is imported and ParagraphVectors method has to be used. For training the model, the fields required are

1. minWordFrequency(n). minimum frequency of the word that should be considered while training. All words below the value of n will be removed before model training.
2. labelsSource(s).labels are attached to the vectors.
3. layerSize(l). defines the number of dimensions for output vectors.
4. windowSize(w). defines context window size.
5. iterate(i). to iterate over the sentences during training.
6. workers(). number of worker threads required to train the model
7. seed().initialisation state. By setting a seed, same set of results are reproducible.
8. build().sequence vectors are built.
9. fit().training the model.

After the model has been trained, vectors can be inferred for other documents and comparison can be done.

1.1.3 Good Quality Ontology

Ontologies have a life-cycle, that is, they are designed, implemented, evaluated and reused [27]. They are not static. There can be changes in the domain and accordingly the implementation of ontology will be changed. Hence, an ontology evolves with the new additions in the domain. The ontology can also be changed according to the requirements. Developers might want to change the class hierarchy or update the properties and relationships among the classes. Ontologies can be versioned, merged or evolved by following proper mechanisms as explained in [8, 30]. For an ontology to be of good quality, it has to be modular, comprehensible and more flexible to changes [17]. This will help in maintaining an ontology and thus in succession, reduce the maintenance cost significantly [2].

An ontology is regarded to be of good quality when its OWL file is consistent, logically complete and contains as much information about the domain as possible. It should also be able to answer the basic competency questions for the domain problem it represents. Competency questions (CQs) represent the domain knowledge that is involved in the ontology. They are important in the life-cycle of an ontology as these represent the requirements and the scope of an ontology. They need to be exhaustive and serve as the litmus test for an ontology. CQs should model the state of affairs in the real world as faithfully as possible. In our example of school ontology discussed in Section 1.1.1, competency questions that can be asked related to the ontology are - In which class does he study? Who teaches maths? In which department does she teach? etc.

One of the main challenges in ontology design is the re-usability. Ontologies can be reused and then adapted according to the requirements of the project but doing so is a tedious and a costly process. For re-usability, small ontologies can be treated as the basic building blocks. These basic building blocks are referred as the ontology design patterns [27]. These can be re-used in ontologies which share the same set of goals and modelling strategies.

1.1.4 Ontology Design Patterns

Ontology Design Patterns (ODPs) [16] are solutions to common modelling problems. ODPs are small, self-contained ontologies that provide solution to commonly occurring modelling problems across different domains. Users can easily adapt these to their own use-cases while still preserving relations with other versions of ontology. Large ontologies can make use of these ODPs and hence, it can help in improving the quality of an ontology. ODP repository [1] has around 220 ODPs that are divided into six categories- content, re-engineering, alignment, logical, architectural and lexico-syntactic ODPs. These are differentiated according to the particular domain problems they focus upon. A general description is provided for each ODP which contains attributes such as name, submitted by, intent of the ODP, competency questions, examples, and OWL file. MODL [27] is another well documented ODP library which contains clearly annotated ODPs. It has dedicated categories of ODPs with associated OWL files, competency questions and the axioms with their meaning. It emphasizes on FAIR, that is, findable, accessible, interoperable and reusable data practices.

AgentRole ODP is one of the ODPs present in MODL repository [27]. This ODP associates an agent with a role. It is unlikely that an agent will be associated with something for all the time. Hence, the relation is not binary and is represented as "agent x associated with a thing y and time t" as seen in Figure1.4. Axioms such as domain, range properties and inverse properties are clearly written and explained. For Example, the property *performsAgentRole* has domain Agent and range AgentRole. Similarly, *hasTemporalExtent* has domain Agent and range as TemporalExtent. Apart from axioms, few competency questions are also included with this ODP.

Other ODPs that are present in these two repositories include Event ODP that refers to an event along with the representation of agent-role activities, climatic zone ODP, news reporting ODP, plan ODP etc.

1.2 Use of ODPs in an ontology

ODPs can be used to refactor a non modular ontology. Large ontologies can be refactored to use ODPs to improve the quality of the ontology. In our example of school ontology, we can include the AgentRole ODP to represent the role of teacher and student. Using this

[1] `http://ontologydesignpatterns.org/`

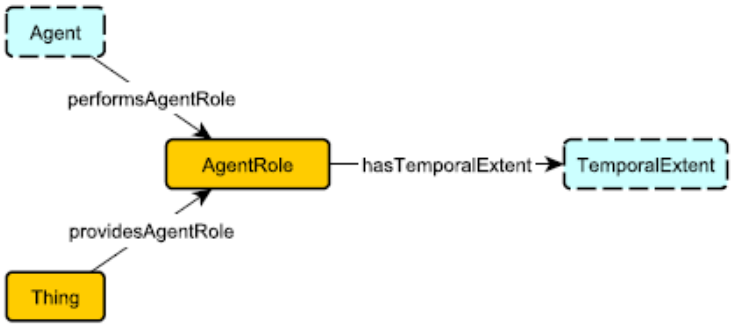

Figure 1.4: AgentRole ODP, figure taken from MODL repository

ODP, it covers the role of a person along with its time span. So, using an ODP enhances the quality of an ontology and also tries to cover all the parameters related to a particular concept.

ODPs have also been used in various well-known ontologies. Several ODPs such as Agent-role, Temporal pattern, Plan pattern, Provenance pattern have been used in chess [20] and recipe [26]. Similar to these two, ODPs have also been used in enslaved ontology [28]. It is an ontology of the historic slave trade. It shows the relationship between people and their roles. The ODPs that have been used in this ontology are Event, Agent-role, Place etc. Arco is another ontology [4] that uses ODPs. Arco represents the Italian Cultural Heritage modelling the recurrent cultural events, festivals and ceremonies. The ODPs used in this ontology are Collection, Situation, Classification and Sequence.

1.3 Need of an ODP Recommender

Developers can choose ODPs according to their domain problem from the ODP repositories and use them in their ontology. But choosing appropriate ODPs for a particular ontology is a hard task even for experienced ontology developers and it becomes even harder for inexperienced ontology developers. ODPs are present across various repositories and picking the right set of ODPs for an ontology from these repositories is a difficult job. The ODPs present in these repositories are of varied domains. There are more than 200 ODPs present across various repositories and identifying the right set of ODPs to be used in an ontology across these various repositories is a tedious task.

1.3.1 Our Contribution

To overcome the issue of including relevant ODPs in an ontology in order to make it modular and more organised, we have designed a tool named *ODPReco* that recommends

relevant set of ODPs for a given ontology. It attempts in resolving the issue by analyzing the lexical, structural and behavioural aspects of an ontology.

In the next chapter, we discuss about the work related to different techniques that have been used in recommendations. In the third chapter, we discuss about the approach followed for the analysis of the ontology. In the fourth chapter, we discuss about dataset and the experiments. The experiments have been done by us and also, we conducted a user-study and based on the user feedback, we have listed the specifications. Finally in the last chapter, we summarize the contributions of the research work carried out as part of this thesis and we also talk about future extensions of this work.

Chapter 2

Related Work

Recommendation systems offer relevant suggestions for a particular entity to the user. These are defined as a means of an assisting process to make choices when there is no sufficient personal knowledge or experience of the alternatives [1]. These are widely used among e-commerce companies and social media platforms. In e-commerce applications, they make recommendations and suggestions based on user's past purchases and history. On social media platforms, friend and follow suggestions are provided. Recommendation systems have also been developed for financial, dining services, etc. Recommendation systems have also been used for recommending research papers [15]. Relevant research papers are recommended based on some important keywords like field of research, research methods etc.

In the domain of software engineering, lot of work has been carried out in recommending software design patterns by analysing the code. In [29], code smells (features of the code that indicate a problem) are analyzed to recommend software design patterns. Various other techniques are used in recommending software design patterns. One such technique is the text based approach [14]. In this technique, the description along with the scenario of the design patterns provided is preprocessed. Pre-processing involves tokenization, stemming (only keeping the root word) and stop word removal (words with less semantic relation with the document are removed). It suggests a vector space model (VSM) for the design patterns. After preprocessing, VSM is used to represent the collection of design patterns in the form of unigrams and bigrams. To ensure that only important words are present and noise is removed, cosine similarity and TF-IDF are used. Most suitable design patterns are suggested based on the cosine similarity.

Another pattern recommending approach is based on question-answering [24]. It is a collective approach in which questions are asked from the user and the user answers them with a yes/no/do not know. Based on the answers, weights are assigned and software design patterns are recommended.

In [23], anti-patterns (bad alternate solutions) are identified and these are analysed to recommend correct design patterns. Structural, behavioural and semantic analysis is

performed and based on the scores, patterns are recommended. The analysis is done using activity diagrams and WordNet. In [9], design pattern is described from structural, behavioural and semantic approaches. The structural aspects are generalization, attribute aggregation and specialization methods. The behaviour of the design patterns can be analyzed through the control flow graphs. The semantic aspect is analyzed using the programming guidelines and naming conventions.

Software design patterns are also recommended using the classification techniques [10]. Supervised machine learning techniques such as ANN, SVM, random forest are used to recommend the software design patterns. A metric based dataset is used to train the model. After the creation of dataset, preprocessing is done followed by the design pattern mining in which classes of the design patterns in the training set are checked against the trained classes [3].

Work has also been carried out in designing an ontology. In [18], sequence of steps are carried out in designing an ontology from the description provided by the user. This has been done for the ontology designing in Chinese language. It has used information retrieval techniques like the tokenization, stemming, Standford parser etc to extract keywords from the description provided by the user. The ontology is then built via the Jena interface or protege. NCBO (National Center for Biomedical Ontology) has an ontology recommender [21]. The user on entering description or keywords (related to biological domain) gets ontology recommendations. The lexical analysis that we follow in ODPReco is inspired from this work.

In [6], the ODPs are re-used based on the competency questions. Competency Questions are analysed and converted to SPARQL queries. These are then annotated with OPLa (pattern language) ontology [7] that specifies already used patterns. In [22], semantic and syntactic approach is followed for ontology matching. The syntactic approach includes the axiom matching and the semantic matching is done by representing the ontology in a graphical form. Tokens of labels are created and two tokens that appear consecutively are linked through an edge.

Ontologies that are similar to each other from the same domain are detected using Lucene [25]. The features considered for lucene approach are the classes, object properties and instances of an ontology. Using this approach, it helps in finding correspondences among entities belonging to different ontologies that relate to the same domain. Following [25], we have used Lucene in our tool.

Chapter 3

ODPReco Description

In this section, we describe the process followed by ODPReco to recommend ODPs. It analyzes an ontology by considering its lexical, structural, and behavioural aspects.

We will discuss the technique to ODP recommendation in the next two sections.

3.1 Ontology Analysis

The ontology is analyzed on the following aspects.

1. **Lexical Analysis**. The signature of an ontology constitutes the names of the classes, property names and individual names present in an ontology. The signature of the ontology is compared with the signature of ODPs present in our collection.

   Apart from the signature, description (if provided by the user) of the ontology is also used in this analysis.

   In our example of school ontology which is discussed in Section1.1.1, class names such as department, people, courses and extracurricular activity along with their sub-classes and associated properties can be considered for signature. Brief overview about this ontology can be included in the description.

2. **Structural Analysis**. The structure of the ontology is analysed by its axioms and also the ontology graph constructed from it.

   The axioms of the given ontology are compared with the ones from our collection. The axioms that we have considered for structural analysis are disjoint classes, object property domain, object property range, sub-object property and sub-class property.

   Each axiom type is of the following format.

$$< AxiomTypeID > (< PropertyInAxiom >< ClassesInAxiom >) \qquad (3.1)$$

Figure 3.1: Structural Arrangement of an ODP and ontology

AxiomTypeID is the name of the axiom, that is, it can be SubClassOf, DisjointClasses, ObjectPropertyDomain, ObjectPropertyRange, SubObjectProperty.

The example of this format in school ontology which is discussed in Section 1.1.1 is ObjectPropertyDomain(<worksFor> <Teacher>)

ObjectPropertyRange(<worksFor> <Department>)

Sub-class axioms can be denoted as

SubClassOf(<Primary> <Department>)

Structural analysis is also done through its graph. The graph of the ontology is constructed using the sub-class and object property axioms. These axioms provide relations among different classes of an ontology and hence, an edge across those classes (referred as vertices) is built.

The ontology graph constructed is mapped with the ODPs present in the collection for determining structural similarity. Those ODPs which are identical or some part of their graph (minimum 3 edges) is identical to the ontology graph are assigned weight of 0.3 while as ODPs which do not match with the ontology structure are assigned a weight of 0 for the given ontology .

Consider all the object property axioms of school ontology.

<Teacher> <works for> <Department>.

<Student> <studies in> <Department>.

<Teacher> <teaches> <Student>.

The structure created from this can be seen in figure 3.1.

Similarly, consider Hazardous Situation ODP. The object property axioms of the ODP are

<Hazard> <is type of> <Hazard Event>.

<Object> <is exposed to> <hazard Event>.

<Hazard Event> <has duration> <Time>.

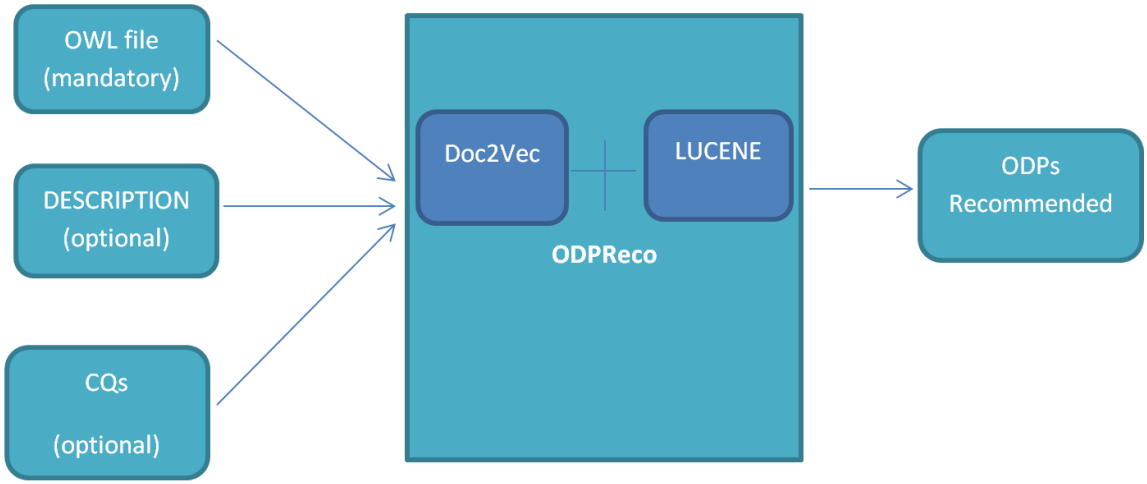

Figure 3.2: Working of ODPReco

<Hazard Event> <has participant> <Object>.

The structure created from this can be seen in figure 3.1. On comparing the two structures, 3 edges are similar and hence a weight of 0.3 is assigned to Hazardous Situation for the school ontology.

Since, the structure does not focus on the semantics and only focuses on the arrangement of an ontology, we have assigned less weight to it. In this example, it is clearly seen that the two structures are similar but semantically, they are different. So, we assign less weight to the structural arrangement.

3. **Behavioural Analysis**. The behavioural aspect of the ontology is analyzed by making use of the competency questions associated with the given ontology. The CQs of the ontology are compared with the CQs of the ODPs in our corpus to carry out the behavioural analysis.

3.2 Approach

ODPReco analyses the ontology using *Doc2Vec* and *Lucene*. Both of them compare the given ontology parameters with those of the ODPs and generate corresponding scores for the ODPs. The scores generated (for the ODPs) from both the techniques are combined together and a threshold is set. The ODPs which have score value above the set threshold are recommended to the user.

Figure 3.2 shows the overall working of ODPReco. It is mandatory for the user to give ontology as input. Ontology is important as it has the signature along with the axioms.

So, for recommendations, the user must at least provide ontology as the input. Description and CQs are optional. It is not mandatory to provide them but it is suggested that the user provides these two also.

Once the information is entered by the user, the IRIs of the axioms are removed and the axioms are obtained in the format as mentioned in (3.1). Since, we semantically compare the given ontology with ODPs, the stop-words from the ontology are removed. A list of stop words are maintained. These are the commonly used words that are not semantically relevant to a document. These words are present in almost all the documents and do not hold any significance. They cannot be treated as the prime keywords for a document. This ensures that only relevant words are retained.

After obtaining the axioms, signature, description and competency questions of an ontology, the ontology is semantically compared with the ODPs. Doc2Vec and Lucene are used to find the relevant ODPs and rank them.

Finding and ranking the relevant ODPs

a) Doc2Vec: In our analysis, we are semantically comparing the ontology with the set of ODPs using Doc2Vec. It is used to generate vectors. The vectors generated can be of sentences/ paragraphs or documents.

We are training ODP data for the learning process. As we are analysing the ontology on three aspects, that is, lexical (signature and description), structural (axioms) and behavioural (competency questions), so we train the model separately on ontology (axioms and signature), description and competency questions. Each ODP (in each of the model) is assigned a unique identifier.

After the user has given the input, the stop words are removed. Ontology characteristics (lexical, structural and behavioural) are obtained and are compared with those of the ODPs (Figure 3.3). Since, each ODP characteristic has an associated identifier, so we compare the ontology with each of the ODPs present (through the unique identifier assigned). The comparison score between the two is obtained through cosine similarity.

Cosine similarity measures the similarity of two vectors. Its value ranges from negative to positive integers. If two vectors are exactly similar, then it returns a 1. When two vectors are dissimilar, then it returns negative integer values.

The cosine similarity values of all characteristics (description, CQs, axioms and signature) of a particular ODP (with the given ontology) are integrated together. As values are added up, the integers obtained do not have a strict upper and lower bound. The values are normalised so that the score is in the range of 0 and 1.

The lowest and the highest ranging value is obtained from the list and used in the equation of normalisation. The equation for normalising a particular *ith* value is

$$value_i = value_i - minvalue/maxvalue - minvalue \tag{3.2}$$

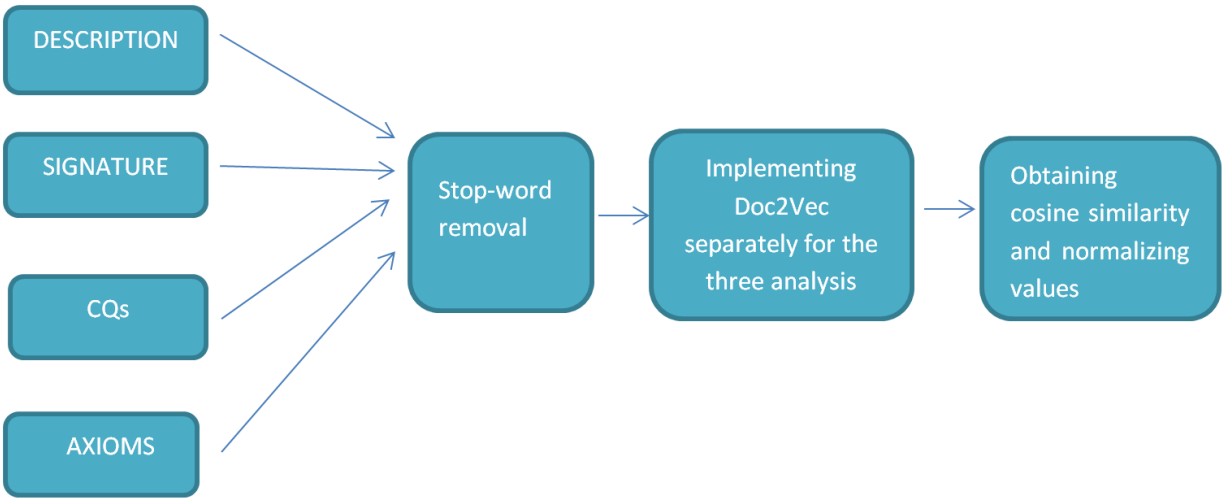

Figure 3.3: Doc2Vec approach

Here, *minvalue* refers to the lowest ranging value in the list whereas *maxvalue* refers to the highest ranging value in the list.

Apart from semantically comparing the ontology with the ODPs, the given input ontology is also compared structurally with the ODPs as discussed in Section 3.1.

b) Lucene: It is executed parallelly with the Doc2Vec approach. Lucene is a text-search engine library. It is an inverted full-text index, that is, it creates an index of the document which allows it to search. The index contains word identifier, number of documents where the word is present, and the position of the word in those documents. So when we give a single word query it just searches the index. For multi-word query, it takes the intersection of the set of files where the words are present and displays the list of matched files in the ranked order.

We have used Lucene for the ODP recommendations (Figure 3.4). The axioms that we have considered are the signature, sub-class property, object property domain, object-property range and description of an ODP. So, using the lucene approach the analysis is done on structural (axioms) and lexical (description as well as the signature) aspects. The document goes through the stop-word removal program and then to Lucene. Lucene indexes the document and lists the related ODPs in their ranked order. The scores obtained are normalized so that they range between 0-1.

ODP Recommender

After analysing the three aspects, that is, lexical, behavioural and structural, the corre-

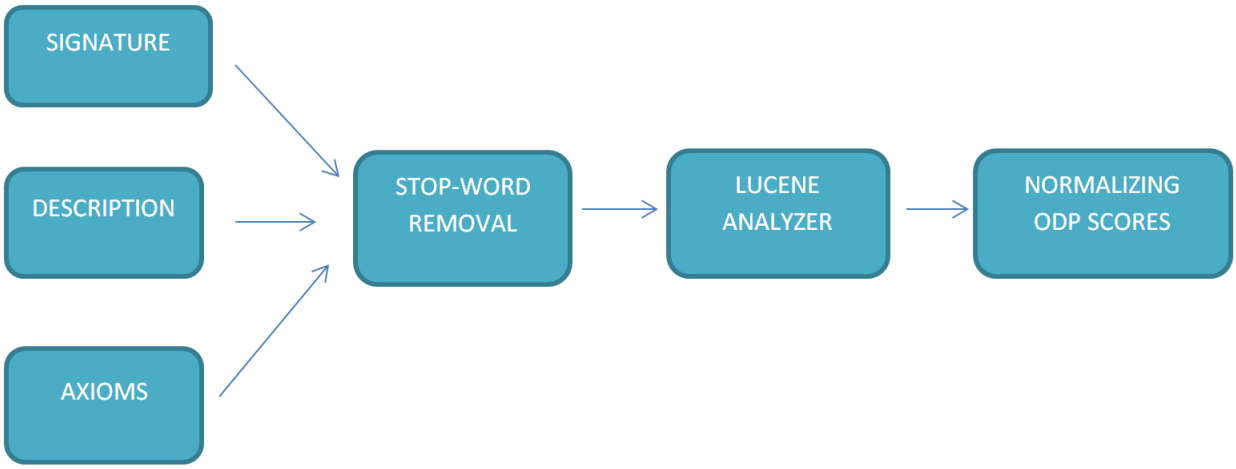

Figure 3.4: Lucene

sponding scores of each ODP are added.

$$w_l * s_{lex} + w_s * s_{str} + w_{sa} * s_{sa} + w_b * s_{beh} \tag{3.3}$$

where

w_l is the weight assigned to lexical analysis,

w_s is the weight assigned to structural analysis,

w_{sa} is the weight assigned to structural arrangement,

w_b is the weight assigned to behavioural analysis,

s_{lex} is the score obtained from lexical analysis,

s_{str} is the score obtained from structural analysis,

s_{sa} is the score obtained from structural arrangement,

s_{beh} is the score obtained from behavioural analysis.

Equation 3.3 has the scores along with the weights assigned. The scores of the behavioural, lexical and structural analysis are obtained from cosine similarity after implementing doc2vec. The structural arrangement score is the one obtained by analysing the arrangement of the ontology through its edges. If the structural arrangement is similar, then a score of one is assigned and if not, then a score of zero is assigned. The weight assigned to the structural arrangement is 0.3 and the weights assigned for lexical, structural and behavioural analysis are 1. These are combined together and normalised to range between 0-1. These normalised scores are then combined with the normalised scores of Lucene.

After combining the scores, a threshold of .85 is set. All the ODPs which have value above the set threshold are mapped to their respective IRIs and recommended to the user in descending order of their scores.

We also attempted separate recommendations from both the approaches (Lucene and doc2vec) by picking the ODPs above the threshold of 0.6 from the two approaches. It gives appropriate recommendations but in order to have a subset of these recommendations, we have integrated the two scores.

We also assigned different weights to the two approaches to see the variation in recommendations. On assigning different weights, the set of ODPs recommended altered a bit and the most appropriate recommendations were provided when no weights were assigned.

Chapter 4

Evaluation

ODPReco is implemented in Java (jdk 1.8). The user can access our tool by downloading our project directly from github [1].The user can run it on command line. Prerequisites for running the tool are that java and maven should be installed in the system.

The user can run our tool by following the command-

java -jar odpreco.jar -ontdes [ ] **-ontcq** [ ] **-ontowl** [ ]

The user has to enter the description file path, CQ file path and ontology (OWL) file path of the ontology in these square brackets. It is mandatory to enter the ontology file path to obtain recommendations while as the description file or CQ file path can be left blank. After the details are entered, ODPReco undergoes processing and relevant ODPs are recommended.

The user can also see the list of ODPs which we have used in our collection. Brief description about each of these ODPs is provided in our github repository.

4.1 Dataset

In order to recommend ODPs, ODPReco maintains a list of available ODPs. The details of the ODPs maintained are - the ontology, competency questions and their description. The ODPs maintained by us are referred as our *collection*. The complete list of ODPs is provided in the Appendix 5. We have maintained 70 ODPs from the following three datasets

1. ODPs from the ODP repository: Out of the 220 ODPs available, we have considered 41 ODPs in our collection. Not all ODPs are included in our collection because several ODPs either do not have downloadable ontology or have similar ontology. So, to avoid redundancy of ontology, only 41 ODPs are included. 40 ODPs are picked

[1] `https://github.com/kracr/ODPReco`

from the Content Pattern ODP and 1 from Architectural ODP category. Apart from the ontology, the description along with the competency questions of these ODPs are maintained in our collection.

2. MODL: Modular Ontology Design Library [27] is a well-documented, downloadable collection of ODPs.The ODPs present in it have well-organised ontology along with the description and competency questions.

3. Manchester ODPs: There is a public catalog of ODPs available named as Manchester ODPs [2] which is completely dedicated for the biological domain. The ODPs present are divided into three categories - Extension ODPs (bypassing the limitation of OWL), Good Practice ODPs (for obtaining robust and a cleaner ontology) and Domain Modelling ODPs (modelling solutions in the domain of biology). 15 ODPs are present in total and all 15 present are included in our collection. We have included manchester ODPs in our collection because lot of ontologies are being made in biological domain.

On evaluating ODPReco, we would like to answer questions about the quality of the recommendations with just the ontology as input (without providing the description and CQs), the affect of providing description and CQs along with the ontology to ODPReco (and whether it necessarily improves the quality) and the performance of ODPReco on ontologies across different domains.

For testing the efficiency of ODPReco, we conducted a user-study. Along with the user study, we have also self evaluated ODPReco on a few ontologies to check the relevance of its recommendations. We first present the results of our self evaluation followed by the response from the user study.

4.2 Self Evaluation of ODPReco

The ontologies that we used for evaluating ODPReco are as follows.

1) NCBO (National Center for Biomedical Ontology) Bioportal: These include the ontologies from bio-medical field [3]. The ontologies that we have used for ODPReco are Kidney disease ontology, Illness and injury ontology, Gender ontology, Orthology ontology and Ontology for biomedical investigations.

2) CSO (Computer Science Ontology) [4]: It combines semantic technologies, machine learning, and knowledge from external sources to automatically generate a fully populated ontology of research areas.

[2] `http://www.gong.manchester.ac.uk/odp/html/`
[3] `https://bioportal.bioontology.org/`
[4] `https://cso.kmi.open.ac.uk/about`

4) Enslaved ontology: It is an ontology about the historic slave trade [28]. It is built for integrating data about the historic slave trade from diverse sources.

5) Be-aware ontology [5]: It is a crisis management ontology for climate related natural disasters. It consists of climatic disasters, analysis of data from sensors and rescue team assignments.

Some of the tests are as-

1. Enslaved ontology

It is an ontology about the historic slave trade [28]. It captures the historical events, including the people associated in those events. It is a modular ontology that uses ODPs like Event, Place, Temporal ones etc.

No CQs are present for this ontology. So, we tested our tool in two ways- first only with the ontology and then in the second test, we have given ontology as well as description as our input.

Table 4.1 shows the results obtained with ontology. ODPs like activity pattern and reaction which represent workflow procedure and roles of different agents are appropriate recommendations for this ontology. Trajectory ODP is used to track the timings and can be used with workflow timings. List manchester ODP is not an appropriate recommendation as it represents the entities related to the biological domain.

ODPs recommended	**Relevance**	**Comments**
Activity Pattern ODP	Yes	discusses agent-role and workflow planning
Reaction ODP	Yes	since concept of agent-role and events is used
List Manchester ODP	No	used for lists restricted to biological domain
Trajectory ODP	Yes	used for workflow and timings

Table 4.1: ODPs recommended for enslaved ontology when ontology is provided to our tool

Table 4.2 has the results obtained when ontology and description are given as input to ODPReco. Bag ODP is relevant for the ontology as it can be used for the list of roles and agents. Event ODP is also appropriate as it lists the activities and the roles of participants during an event. Using description along with the ontology, two more relevant recommendations are provided.

When both, ontology and description are given as input to ODPReco, it gives more choices in recommendations.

2. Be-Aware ontology :

It has ontology, description as well as CQs. We ran 4 tests on this ontology- one only with ontology, other with ontology and description , third one with ontology and CQs and the last with all the three.

[5] `https://github.com/beAWARE-project/ontology/blob/master/beAWARE_ontology.owl`

ODPs recommended	Relevance	Comments
Bag ODP	Yes	for the components
Trajectory ODP	Yes	-
Reaction ODP	Yes	-
Event ODP	Yes	-
List Manchester ODP	No	used for lists restricted to biological domain

Table 4.2: ODPs recommended for enslaved ontology when ontology and description is provided to our tool

Table 4.3 shows the results of be-aware when only ontology is given as input. All the ODPs recommended are appropriate. Agent-role, LCA and activity pattern ODPs represent the activities and roles of different agents. So, these can be used for assigning team roles and assignments. Toco ODP is also an appropriate recommendation. It represents tele-communication devices and it can be used with sensor devices.

ODPs recommended	Relevance	Comments
Agent-Role	Yes	for team roles ans their assignments
Toco ODP	Yes	as it lists devices which can be used with sensors
LCA ODP	Yes	can be used with rescue team
Activity Pattern ODP	Yes	team assignments

Table 4.3: ODPs recommended for Be-Aware ontology when ontology is given as input

Table 4.4 show the recommendations when ontology and description are given as input. All the recommendations except the Adapted SEP are appropriate.

ODPs recommended	Relevance	Comments
LCA Pattern ODP	Yes	can be used with rescue team
Toco ODP	Yes	-
Task Role ODP	Yes	can be used with rescue team work
Adapted SEP ODP	No	used for biological domain
Activity Pattern ODP	Yes	-
Temporal extent ODP	Yes	referring to time

Table 4.4: ODPs recommended for Be-Aware ontology when ontology and description is provided to our tool

Table 4.5 show the results obtained when ontology and CQs are given as input to ODPReco. Agent-role odp and activity pattern odp are recommended and both are appropriate recommendations and can be used for assigning rescue team works and for planning the workflow procedure.

Table 4.6 lists the results when ontology, CQs and description are given as input to ODPReco. Among the odps recommended, adapted sep and entity quality odp are not

ODPs recommended	Relevance	Comments
Agent Role ODP	Yes	-
Activity Pattern ODP	Yes	-

Table 4.5: ODPs recommended for Be-Aware ontology when ontology and CQs are provided to our tool

relevant for be-aware as these are related to biological domain.

ODPs recommended	Relevance	Comments
LCA Pattern ODP	Yes	used with agents and events
Task Role ODP	Yes	-
Entity Quality ODP	No	related to biological cells
Toco ODP	Yes	-
Task Execution ODP	Yes	execution analysis of a task
Adapted SEP ODP	No	restricted to biological domain

Table 4.6: ODPs recommended for Be-Aware ontology when ontology and CQs are provided to our tool

So, on analysis across different inputs given to ODPReco, the ODP recommendations change slightly depending on the input. Giving ontology as the input gives appropriate recommendations. On giving description and CQs as input, the number of recommendations change and in some cases, we obtain better set of recommendations.

The self evaluation helped us in answering few of the questions we wanted to know from these experiments. When only ontology is given as input, we obtain appropriate set of recommendations. Also, on running ODPReco on ontologies across different domains, the set of ODPs recommended are appropriate and when all the inputs are provided, the set of relevant ODP recommendations increase.

4.3 User Study

We conducted a user-study in which 13 users participated. These users are familiar with ontology development. They provided an ontology of their choice and in some cases, they also provided the description of the ontology and the CQs. These were given as input to ODPReco. Along with the ODP recommendations, the users were requested to fill an anonymous feedback form. The questions from the feedback form are as

1. How experienced user are you with ontology modelling?
2. Do you frequently use ODPs while modelling?
3. How experienced are you with using ODPs?

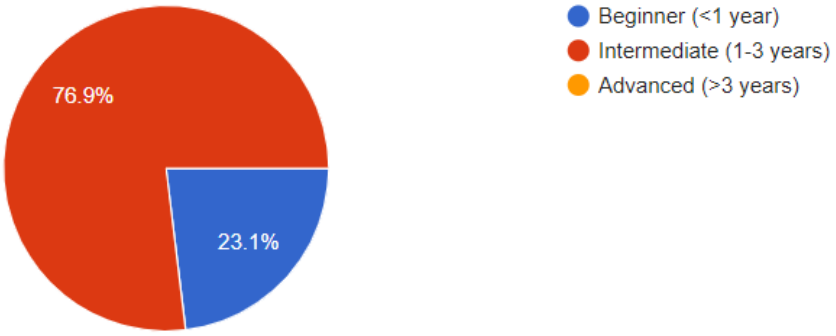

Figure 4.1: Ontology modelling experience statistics of the users

4. If you have been frequently using ODPs during ontology development, list your frequently used ODPs?
5. Provide the name of the ontology and briefly describe it.
6. Apart from the OWL file, what other parameters were given as input?
7. What are the ODPs recommended by ODPReco?
8. Are the ODP recommendations relevant for the ontology?
9. Are you happy with the ranking of recommendations?
10. Would you rank the ODP recommendations differently? If yes, what would be the ranking?
11. If you think some of ODP recommendations are missing, please mention them?
12. On a scale of 1-5, how would you rate ODPReco?
13. Do you have any suggestions to improve ODPReco?

Questions 1-4 help us to determine how skilled the user is in ontology building and questions 5-13 are related to the performance of our tool.

Among the 13 users, 10 users have an intermediate experience of ontology modelling, that is, 1-3 years of experience and 3 users are beginners in ontology modelling, that is, they are into ontology modelling since few months as seen in Figure 4.1. So, the users we have considered have some amount of prior knowledge in ontology modelling.

Among these 13 users, 7 users have an intermediate experience (1-3 years) in using ODPs in their ontologies and 6 users have an experience of few months in using ODPs (Figure 4.2. So, the users are aware about the ODPs. Also, in our survey, we asked them to

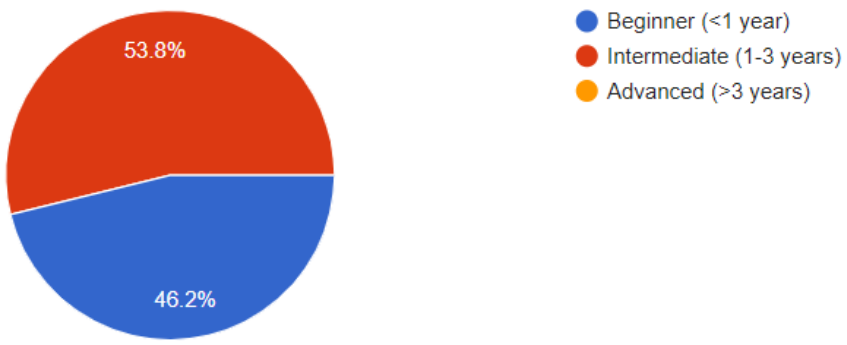

Figure 4.2: ODP usage experience statistics of the users

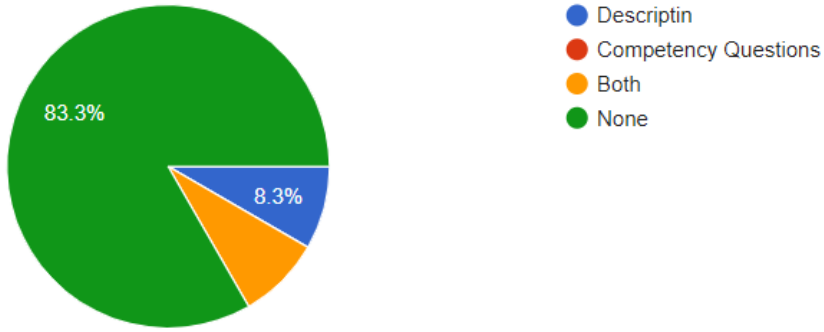

Figure 4.3: Input parameters given by user

provide the frequently used ODPs. Most commonly used ODPs by them are Agent-Role ODP, Event ODP, Stub ODP, Provenance ODP, Information Realisation ODP, Activity Pattern ODP, Object-Role ODP.

The ontologies that users have used in testing ODPReco are from different domains. These include Cooking ontology, Census ontology, University ontology, Hospital management ontology, Netflix ontology, Snomed clinincal ontology, Pizza ontology, Global terrorist ontology, Nutrition ontology, Profession ontology, Egyptian University ontology, News reporting ontology, UNIV-BENCH-OWL2DL ontology.

Among these 13 users, 11 users have provided just the ontology, 1 user has provided the description of the ontology along with the ontology and 1 user has given all the three parameters, that is, ontology, description and CQs as input. It can be seen in Figure 4.3. This reflects the real-world scenario wherein most ontologies do not have a description and CQs associated with them.

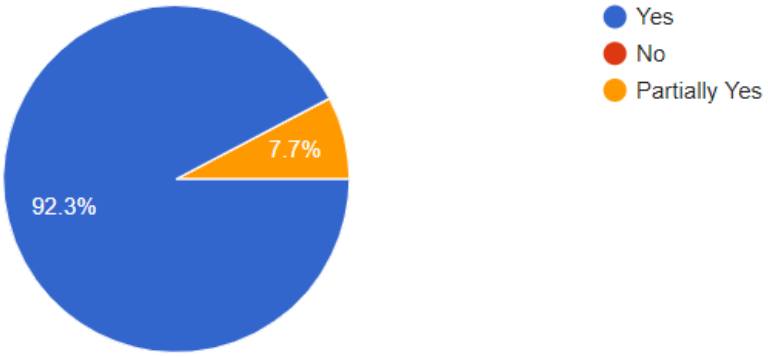

Figure 4.4: Relevance of recommendations

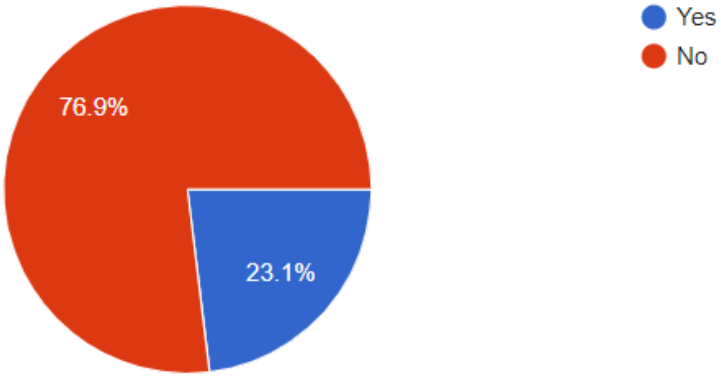

Figure 4.5: Statistics showing if user wanted to rank ODPs differently

12 users found our ODP recommendations completely relevant while as 1 user found it partially relevant (Figure 4.4).

We also asked the users if they would have ranked the ODP recommendations differently and three users said that they preferred a different order (Figure 4.5). Two users wanted 3rd and 4th ranked ODP as the first ranked ODP among the recommendations and one user wanted to shift an ODP to bottom rank.

We also asked the users if any relevant ODPs were missing from the list recommended by ODPReco. Three users said that an ODP was missing and suggested the name of the ODP. Two users suggested Event ODP for News reporting ontology and School ontology and one user suggested on keeping design patterns related to medicine field.

We also asked them to rate our tool on the scale of 1-5, where 1 is poor and 5 is excellent.

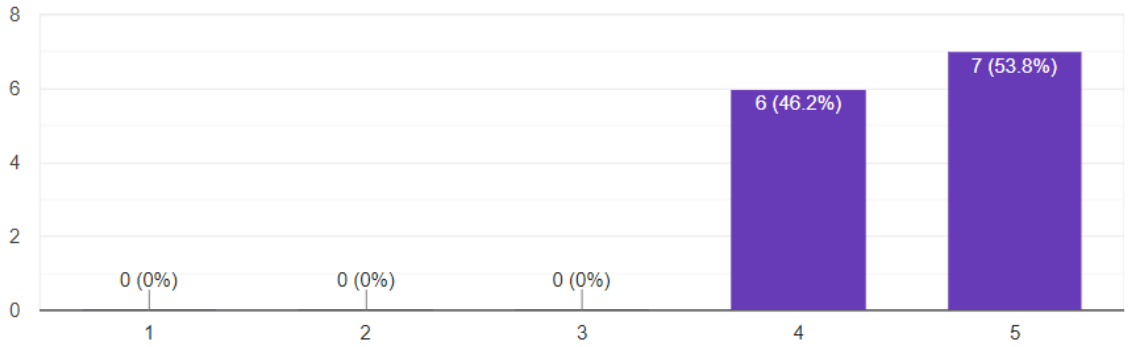

Figure 4.6: User ranking for our tool

Out of thirteen users, six users gave rating of 4 and seven users gave rating of 5. (Figure 4.6).

From these ratings, we can suggest that ODPReco is recommending appropriate ODPs.

We also asked users for their suggestions in improving the tool. Two users suggested building an interface for the tool and one user suggested providing an explanation of recommendations by ODPReco. Along with that, one of the users also suggested to provide the names of the classes and properties that can be replaced with the recommended ODPs. We will explore these ideas as part of our future work.

From the user study conducted, we can state that the recommendations provided by our tool are appropriate.

Chapter 5

Conclusion and Future Work

ODPs help in making an ontology modular. But finding the right set of ODPs across different repositories is a difficult task and it becomes even more difficult for a naive user. Our tool ODPReco helps in solving this problem by recommending the appropriate and relevant set of ODPs for a given ontology.

The inputs required for ODPReco are CQs (optional), description (optional) and axioms (mandatory). The ODPs that we have considered in our dataset are taken from ontology design pattern resository, manchester odps repository and MODL repository. The experiments conducted by us and the user-study indicate that the recommendations provided by the tool are appropriate.

In future, ODPReco can be made as a plugin in Protégé so that the user can get recommendations directly in it. Also, other software pattern approaches can be integrated with this tool in order to increase the efficiency. Along with that, another useful feature that could be made part of ODPReco is the explanation feature for the recommendations suggested during the user-study. We will also explore the option of building it as a website.

Bibliography

[1] Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, November 2015. Publisher: Elsevier.

[2] Anas Bassam AL-Badareen and et. al. The Impact of Software Quality on Maintenance Process. *International Journal of Computers*, 5:183–190, 2011.

[3] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018, January 2018.

[4] Valentina Anita Carriero, Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, and Amedeo Modigliani. An Ontology Design Pattern for representing Recurrent Events. page 12.

[5] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems and their Applications*, 14(1):20–26, 1999.

[6] Paolo Ciancarini, Andrea Giovanni Nuzzolese, Valentina Presutti, and Daniel Russo. SQuAP-Ont: an Ontology of Software Quality Relational Factors from Financial Systems. *Semantic Web*, pages 1–15, April 2020. arXiv: 1909.01602.

[7] Ricardo de Almeida Falbo, Monalessa Perini Barcellos, Julio Cesar Nardi, and Giancarlo Guizzardi. Organizing Ontology Design Patterns as Ontology Pattern Languages. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data*, Lecture Notes in Computer Science, pages 61–75, Berlin, Heidelberg, 2013. Springer.

[8] Rim Djedidi and Marie-Aude Aufaure. Ontology Evolution: State of the Art and Future Directions. *Ontology Theory, Management and Design: Advanced Tools and Models*, pages 179–207, 2010.

[9] Jing Dong, Yajing Zhao, and Tu Peng. A Review of Design Pattern Mining Techniques. *International Journal of Software Engineering and Knowledge Engineering*, 19:823–855, 2009.

[10] Ashish Kumar Dwivedi, Anand Tirkey, and Santanu Kumar Rath. Software design pattern mining using classification-based techniques. *Frontiers of Computer Science*, 12(5):908–922, October 2018.

[11] Aldo Gangemi. Ontology Design Patterns for Semantic Web Content. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *The Semantic Web – ISWC 2005*, Lecture Notes in Computer Science, pages 262–276, Berlin, Heidelberg, 2005. Springer.

[12] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer, Berlin, Heidelberg, 2009.

[13] Nagyp'al G'abor. Ontology Development. In Rudi Studer, Stephan Grimm, and Andreas Abecker, editors, *Semantic Web Services: Concepts, Technologies, and Applications*, pages 107–134. Springer, Berlin, Heidelberg, 2007.

[14] Abeer Hamdy and Mohamed Elsayed. Automatic Recommendation of Software Design Patterns: Text Retrieval Approach. *Journal of Software*, 13:260–268, 2018.

[15] Khalid Haruna, Maizatul Akmar Ismail, Damiasih Damiasih, Joko Sutopo, and Tutut Herawan. A collaborative approach for research paper recommender system. *PLOS ONE*, 12(10):e0184516, October 2017. Publisher: Public Library of Science.

[16] P. Hitzler and et. al. *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*. IOS Press, Amsterdam, The Netherlands, 2016.

[17] Pascal Hitzler and Cogan Shimizu. Modular Ontologies as a Bridge Between Human Conceptualization and Data. In *Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018.

[18] D. Huaqiang, L. Hongxing, X. Songyu, and F. Yuqing. The Research of Domain Ontology Recommendation Method with Its Applications in Requirement Traceability. In *2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, pages 158–161, October 2017.

[19] Krzysztof Janowicz, Aldo Gangemi, and Adila Krisnadhi. Introduction: Ontology Design Patterns in a Nutshell. *International Semantic Web Conference*, page 6.

[20] Adila Krisnadhi and Pascal Hitzler. Modeling With Ontology Design Patterns:Chess Games As a Worked Example. In *Ontology Engineering with Ontology Design Patterns*, 2016.

[21] Marcos Martínez-Romero, Clement Jonquet, Martin J. O'Connor, John Graybeal, Alejandro Pazos, and Mark A. Musen. NCBO Ontology Recommender 2.0: an

enhanced approach for biomedical ontology recommendation. *Journal of Biomedical Semantics*, 8(1):21, December 2017.

[22] Eleni Mikroyannidi, Manuel Quesada-Martínez, Dmitry Tsarkov, Jesualdo Tomás Fernández Breis, Robert Stevens, and Ignazio Palmisano. A Quality Assurance Workflow for Ontologies Based on Semantic Regularities. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, Lecture Notes in Computer Science, pages 288–303, Cham, 2014. Springer International Publishing.

[23] Nadia Nahar and Kazi Sakib. Automatic Recommendation of Software Design Patterns Using Anti-patterns in the Design Phase: A Case Study on Abstract Factory. *Journal of Software*, page 8, 2015.

[24] F. Palma, H. Farzin, Y. Guéhéneuc, and N. Moha. Recommendation system for design patterns in software development: An DPR overview. In *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*, pages 1–5, June 2012.

[25] G. Pirro and D. Talia. An approach to ontology mapping based on the lucene search engine library. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*, pages 407–411, 2007.

[26] Monica Sam, Adila Alfa Krisnadhi, Cong Wang, John Gallagher, and Pascal Hitzler. An Ontology Design Pattern for Cooking Recipes: Classroom Created. In *Proceedings of the 5th International Conference on Ontology and Semantic Web Patterns-Volume 1302*, pages 49–60. CEUR-WS.org, 2014.

[27] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. MODL: A Modular Ontology Design Library. *International Semantic Web Conference*, 2019.

[28] Cogan Shimizu, Pascal Hitzler, Quinn Hirt, Dean Rehberger, Seila Gonzalez Estrecha, Catherine Foley, Alicia M. Sheill, Walter Hawthorne, Jeff Mixter, Ethan Watrall, Ryan Carty, and Duncan Tarr. The enslaved ontology: Peoples of the historic slave trade. *Journal of Web Semantics*, page 100567, April 2020.

[29] Stephen Smith and D. R. Plante. Dynamically recommending design patterns. In *International Conference on Software Engineering and Knowledge Engineering*, 2012.

[30] Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. Ontology evolution: a process-centric survey. *The Knowledge Engineering Review*, 30(1):45–75, January 2015.

List of Publications

1. Maleeha Arif Yasvi, Raghava Mutharaju: ODPReco - A Tool to Recommend Ontology Design Patterns In *Workshop on Ontology Design and Patterns WOP @ International Semantic Web Conference, 2019* : 71-75

2. Invited chapter to a book titled "Advances in Pattern-based Ontology Engineering". IOS Press. To be published.

Appendix

ODPs used for the evaluation of the tool are as follows

1. **Acting For ODP**: This ODP represents that some agent is acting in order to forward the action of an agent.

2. **Chess ODP**: The ODP represents a flexible schema for linked data querying of chess games. Players are modeled as AgentRole. In addition, notions such as chess moves, chess tournaments, as well as chess game annotation are included.

3. **Climatic Zone ODP**: The intent of this ODP is to be able to represent climatic zones for aquatic resources.This pattern allows to query what climatic zones are typical of an aquatic resource.

4. **Description ODP**: To formally represent a conceptualization or a descriptive context.This ODP allows the designer to represent both a descriptive context and the components that characterize in that context.

5. **Hazardous Situation ODP**: This ODP provides a building block for modelling situations where one or more object is exposed to one or more hazards to some extent.They also result in some consequences.

6. **Region ODP**: It is a basic ODP which talks about attributes/parameters/dimensions, while still referring to their values.

7. **DUL ODP**: The Affordance ODP relies on the descriptions and situations and is combined with a frame-based representation scheme. This allows to extend the notion of affordance not only to physical objects, but also complex situations afford actions.

8. **Activity Pattern ODP**: It incorporates the general two perspectives of activities: a workflow perspective, which are often observed in planning-related applications, and a spatiotemporal perspective, which are often found in geographic activity analysis.

9. **Born Digital Archives ODP** The ODP models the domain of born digital archives. It associates activities and events around an agent.

10. **Classification ODP**: To represent the relations between concepts (roles, task, parameters) and entities (person, events, values), which concepts can be assigned to.

11. **Componency ODP**: It represent that objects either are proper parts of other objects, or have proper parts.It allows designers to represent part-whole relations and distinguish between parts and proper parts.

12. **Computer System ODP**: It models computer systems based on a hardware/software approach.It is expected to facilitate the creation of computer system domain ontologies that can be exploited in numerous fields.

13. **Constituency ODP**: Represents the constituents of a layered structure. It models the physical objects and its parts.

14. **Digital Video ODP**: It models digital video files, their components and other

associated entities, such as codecs and containers.This ODP is expected to facilitate the creation of digital video domain ontologies.

15. **Event ODP**: It provides a minimalistic model of event where it is not always possible to separate its spatial and the temporal aspects, thus can model events that move or possess discontinuous temporal extent. Events according to this model has at least one participant, attached via a participant-role, and may have additional descriptive information through its information object.

16. **GoTop ODP**: It allows to categorize gene-related types.It represents parts of gene-related entities (transitively).

17. **Gear Species ODP**: It outlines the relations between aquatic species and types of fishing gear that are suitable for catching. While we do not make this distinction from the species' viewpoint, we do distinguish what species are gear types targeted to and what can be also incidentally caught.

18. **Information Realisation ODP**: Represents information objects and their physical realization.It distinguishes information objects from their concrete realizations.

19. **Intension Extension ODP**: Represents the meaning of an information object: the concepts it expresses, the things it is about.It employs a simple set of properties to link information objects to their meanings, and to entities they can be about.

20. **Invoice ODP**: Heterogeneous models for invoices can be aligned to this ODP, which then acts as a semantic facade to different invoice management applications. The Context class can be used to gather temporal, spatial and organizational data.

21.**LCA Pattern ODP**: Life Cycle Assesment ODP studies the environmental impact of products taking into account their entire life-span and production chain.It specifies key aspects of LCA/LCI data models, namely the notions of flows, activities, agents, and products, as well as their properties.

22. **List ODP**: Represents ordered lists, through a specialization of the bag pattern, where each resource in the bag is referred through an item, so that the same item can occur in several places. The usual properties of lists are also there, i.e. the sequence of elements, and references to the first and last item.

23. **n-ary Participation ODP**: Represents events with their participants, time, space, etc.All sorts of relations denoting events with multiple participants, space-time indexing, etc. can be represented with this pattern.

24. **News Reporting Event ODP**: It can be used for modelling situations in which we are not certain that a particular actual event has the properties which were described in a news message. It defines the properties of an actual event which were reported (time, place, actors, subevents, cause, effect etc.), but not to treat them as universal, verified knowledge.

25. **Object Role ODP**: Represents objects and the roles they play.

26. **Place ODP**: Models places of things and represents, transitively, where something

is located. It remains unspecified what kind of location relation to represent: reference location, partial location, physical location, social or metaphoric location, etc.

27. **Policy ODP**: It models policies, their characteristics and their associated entities, such as processes and agents.

28. **Reaction ODP**: Models dynamic situations, tracking agents and actions they produce, events that are results of some action(s), and consequences as new actions.

29. **Task Role ODP**: Represents the assignment of tasks to roles. It allows to put roles in the domain of discourse.

30. **Set ODP**: It is a collection that cannot contain duplicate elements. A Set is expressed by linking to it directly all the members (elements), multiple identical values of members (elements) will be eliminated because by default they are treated as a set.

31. **Simple Aggregated ODP**: It represents objects that can be simple or aggregated (that is, several objects gathered in another object acting as a whole).The aggregated members should belong to the same concept. For example, a turbine is part of an engine.

32. **Species Bathemetry ODP**: Represents species together with their typical environment in terms of bathymetric range and water area.

33. **Tagging ODP**: Rrepresents a tagging situation, in which someone uses a term, from a list, to tag something (or the content of something). It also represents the time and the polarity of the tagging.

34. **Task Execution ODP**: Represents actions through which tasks are executed. It allows designers to make assertions on roles played by agents without involving the agents that play that roles, and vice versa.

35. **Time Indexed Participation ODP**: To represent participants in events at some time.

36. **Topic ODP**: To represent topics and their relations.Topics are modelled as conceptual complexes with part of (containment), overlap, and vicinity relations, and can be related to any kind of entity.

37. **Transition ODP**: Represents basic knowledge about transitions (events, states, processes, objects).

38. **Vertical Distribution ODP**: The intent of the ODP is to be able to represent vertical distribution for aquatic resources.This pattern allows to query what vertical distribution is typical of an aquatic resource. Whereas such values can be subject to observation, another pattern based on the generic observation pattern should be used.

39.**Vessel Species ODP**: Provides a direct relation between aquatic species and vessels that are able to catch them, regardless of the fishing gear used.

40. **Trajectory ODP**: It provides a model of trajectory, which is understood as a sequence of spatiotemporal points.This pattern is suitable for a variety of trajectory datasets and locations and is easily extendible by by aligning to or matching with existing trajectory

ontologies, foundational ontologies, or other domain specific vocabularies.

41. **Toco ODP**: It is an ODP for telecommunication networks. It has properties like-hasDevice, hasLink,hasService etc.

42. **Interaction ODP**: To model different interactions where the interactors can have different roles.

43. **Manchester Sequence ODP**: Models the sequence of events, one after the other. This ODP has been taken from manchester odp site.

44. **Manchester List ODP**: The List is used to model ordered elements, representing the semantics of the order. This ODP models the list of genes.

45. **AdaptedSEP ODP**: Models selective transitive propagation in the biomedical domain.

46. **Exception ODP**: Models exceptions in the biomedical field without breaking the strict class-subclass hierarchy: for example the class MammalianRedBloodCell (with restriction of hasNuclues as 0)would be a subclass of EukaryoticCell (with the restriction HasNucleus exactly 1).

47. **Data Type Relationship ODP**: Represents a datatype value with more than one aspect. Numerical values can have different aspects. For example, a boiling point has a temperature value, a pressure, etc.

48. **Entity Feature Value ODP**: Models features with the simplest structure possible.This ODP is used to represent modifiers with multiple aspects, thus features (e.g. colour with a certain brightness and saturation).

49. **Selector ODP**: To recreate selectors in the biomedical field, that is refining entities that can be used to choose between to alternatives: for example, right or left hand. A selector is a modifier that can be used to select between identical entities, e.g. right and left hand.

50. **Normalisation ODP**: To untangle a polyhierarchy, coding the subsumption relationships using restrictions rather than class-subclass relationships. The application example for this ODP is adapted from the Cell Type Ontology. In the example, the subsumption relationships that already are in the Cell Type Ontology are inferred by the reasoner instead of hard-coded.

51. **Upper Level ODP**: Creates an ontology that can integrate different ontologies in itself.

52. **Closure ODP**: Simulates the closed world assumption in a concrete class.One of the examples of such problem is the fact that plenty of users think that asserting an existential restriction is enough to close a relationship, when in fact a universal restriction is also needed: it is not enough to say that carnivore eats some meat, as that is equivalent to saying that it can eat another things apart of meat.

53.**Entity Quality ODP**: To model qualities without relying in a proliferation of object

properties. This ODP is modeled for biomedical field.

54. **Value Partition ODP**: Models values of attributes. It models biological regulation, being negative or positive.

55. **Entity Property Quality ODP**: To model qualities of independent entities (e.g. position, colour).

56. **Manchester Description ODP**: To simulate an If-Then of the type: if something fulfills certain conditions, it should have a further given attribute.

57. **Agent Role ODP**: This ODP has been obtained from MODL. It represents the roles of an agent at some point of time. It models its tasks, events, location etc.

58. **Bag ODP**: The pattern for an Aggregation, Bag, or Collection is relatively simple. The Bag is a type of unordered collection. This pattern demonstrates a more approachable interface for the partonymy pattern, with respect to membership.

59. **Explicit Typing ODP**: This ODP is used when there is a finite, but mutable number of types of a thing. We find this easier to maintain than a series of subclass relationships.

60. **Identifier ODP**: It is used for associating some sort of identifier and metadata with a thing. It can associate additional information aside from its type with a thing, e.g. an identifier may be a URL or a primary key value in a database.

61. **Name Stub ODP**: The NameStub ODP is a specialization of the Stub Pattern.

62. **Participant Role ODP**: It is a specialization of the AgentRole Pattern, many axioms are inherited due to this.This pattern has additional synergies with the Event.

63. **Provenance ODP**: It suffices to align a sub-pattern to the core. The EntityWith-Provenance class is any item of interest to which a developer would like to attachprovenance information.It is interested in capturing, who or what created that item, what was used to derive it, and what method was used to do so.

64. **Quantities ODP**: It allows a developer to express a quantity of some stuff. The nature of quantities is rather complex, due to the fact that there are a multitude of dimensions, unit types, and ways to measure quantities. The Quantity class is used to express the nature of the quantity via its QuantityKind. A QuantityValue expresses the magnitude of the Quantity via an xsd:double and a Unit.

65. **Reification ODP**: It is essentially a metapattern. It represents a set of axioms that will allow a developer to quickly reify a concept by specializing the framework.

66. **Spatial Extent ODP**: It is characterized by a set of Interiors, which are in turn characterizedby a PointInSpace-Sequence. A PointInSpace-Sequence consists of PointInSpace-SequenceElements, which are constituted by PointInSpace. A PointInSpace is described by a value and a reference system.

67. **Spatio-Temporal Extent ODP**: It wraps the Trajectory Pattern. Essentially, it uses the Trajectory patterns ability to capture discrete snapshots of something moving along some dimension, but casts it into the familiar physical dimensions, plus time. This is

done by adding the atPlace and atTime properties.It is used when it is difficult to separate space and time when talking about a concept.

68. **Stub ODP**: This ODP defines the meta-data of the attributes. It relates the concept with the data types.

69. **Temporal Extent ODP**: It is composed of a number of ComplexTimeIntervals, which may be intervals of non-zero length (i.e.TimeIntervals) or intervals of length 0 (i.e. PointsInSpace).

70. **Tree ODP**: It allows a developer to organize data into a tree data structure which uses properties such as hasChild, hasAncestor, hasDescendant, hasSibling etc.