DOI:

Journal of Computers and Intelligent Systems

Vol. 2, No. 1, 2024, pages 33 - 51

Journal Homepage:

https://journals.iub.edu.pk/index.php/JCIS/

Agile Ontology Development: A Comprehensive Framework from Preliminary Investigations to Evaluation

Asif Raza[1], Muhammad Imran Ali[2], Qasim Niaz[3], Majid Khawar[4], Muhammad Asif[5]

[1] Department of Computer Science, Institute of Southern Punjab, Multan Pakistan

[2]Department of Computer Science, Institute of Southern Punjab, Multan Pakistan

[3]Department of Computer Science, Institute of Southern Punjab, Multan Pakistan

[4]Department of Computer Science, Institute of Southern Punjab, Multan Pakistan

[5]Department Computer Science & IT, NCBA&E Lahore, Multan Campus, Punjab Pakistan

ARTICLE INFO

Article History:

Received:	May	01, 2024
Revised:	May	03, 2024
Accepted:	May	03, 2024
Available Online:	May	05, 2024

Keywords:

Ontology Development
OWL Ontology
Ontology evaluation
Agile methodologies
Semantic representation

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

This research proposes a comprehensive framework for ontology development within the database domain, integrating agile methodologies from initial investigations to evaluation. The framework comprises three main phases: Preliminary Investigations, Ontology Design and Development, and Ontology Evaluation, each encompassing specific sub-stages. In the Preliminary Investigations phase, entity recognition is conducted through interviews, document analysis, and keyword extraction using tools like Monkey Learn. Relationships between entities are identified to establish hierarchies and associations, facilitating semantic representation. The Ontology Design and Development phase involves domain modeling and ontology implementation, including the transformation of entity relations into OWL classes and properties. Guidelines for mapping entity relations to OWL are provided, ensuring a seamless transition from Entity Relation Diagrams (ERD) to ontology representation. Finally, the Ontology Evaluation phase employs two methods: Domain Experts Evaluation and Pellet Reasoner. Domain experts assess the ontology's credibility, consistency, completeness, and conciseness, while the Pellet Reasoner ensures logical consistency and provides essential inference services. The proposed framework offers a systematic and agile approach to ontology development, particularly beneficial in dynamic domains like database management.

Corresponding Author's Email: ImranAli.mltn@gmail.com

Citation:

1. Introduction

Ontology development is vital for knowledge representation, especially in dynamic domains like database management. Agile methodologies offer a promising approach to address challenges in ontology development, ensuring adaptability to evolving requirements. The Semantic Web relies on ontologies to encapsulate domain knowledge systematically, facilitating information exchange and reuse across platforms. In computer science, ontology formalizes information categories within a framework, playing a key role in various fields like knowledge management and e-commerce. Ontology engineering involves creating and managing ontologies, requiring research into the domain and collaboration among contributors. Agile methodologies, emphasizing adaptability and iterative development, align well with the dynamic nature of ontology engineering. This paper proposes a flexible framework for ontology engineering, integrating agile practices and

tools like Ontology and Travis CI. The framework aims to enhance efficiency and achieve better results in vocabulary development by organizing ontology development activities effectively. Objectives include reviewing ontology development techniques, designing a practical solution for ontological databases, and evaluating its impact on web performance and user satisfaction. Key questions addressed include the limitations of existing ontology engineering techniques, methodologies for user participation in ontology creation, and mechanisms for updating ontological databases. The research aims to enhance search result relevance and facilitate user-oriented information retrieval on the web. Ontological databases, structured repositories based on formal ontologies, offer a potential solution to optimize web performance and refine search outcomes. Encouraging user participation and collaboration among experts enriches database quality and coverage. Continuous learning and adaptation mechanisms ensure the relevance of ontological databases in evolving web environments [1]. The research extends to developing an ontology to recommend a Database Management System (DBMS), utilizing agile development techniques and expert evaluation. This contributes to the evolution of ontological databases and user-centric web performance optimization.

2. Related Work

Ontology serves as a formal and explicit description of how we conceptualize various aspects of reality, providing a standardized framework for sharing and reusing information within the AI community. It offers a structured representation of domain knowledge, delineating vocabularies and their relationships within a specific domain. Notable real-world examples include Yahoo! categories, Amazon's product catalog, and various specialized ontologies like SWEET and GENE. To facilitate the definition of ontologies, several ontology languages have been developed, with the Web Ontology Language (OWL) emerging as a standardized language for the semantic web. Definitions of ontology vary, reflecting different scopes and levels of formality. Gruber's definition emphasizes ontology as an explicit, formal description of shared conceptualizations, featuring formal and explicit language, community-driven consensus, and a perspective on modeling the world. Guarino's logics-focused approach restricts ontology to a logical theory offering an explicit, partial description of a conceptualization, highlighting logical accuracy and contextual character. From an engineering perspective, ontologies are viewed as frameworks for combining words and relations to form extensions of vocabulary, facilitating system integration and reuse.

Ontologies can be categorized based on formality and shareability. Formality ranges from very casual to rigorously formal, with most modern ontologies falling into partially informal or semi-formal categories. Shareability refers to the extent to which ontologies are designed for broad use and collaboration. Private ontologies reflect individual viewpoints, while application-specific and openly developed ontologies involve project-specific or public contributions, respectively. Standard ontologies, developed by key organizations, aim to standardize concepts within specific domains [2].

Ontologies are further classified into top-level, domain, task, and application ontologies based on scope and purpose. Core ontologies address fundamental concepts within specific domains, while meta-ontologies, or portrayal ontologies, define basic building blocks for formalizing knowledge representation. Overall, the diverse perspectives on ontology reflect its multifaceted nature, with ongoing efforts to establish universal definitions and classifications.

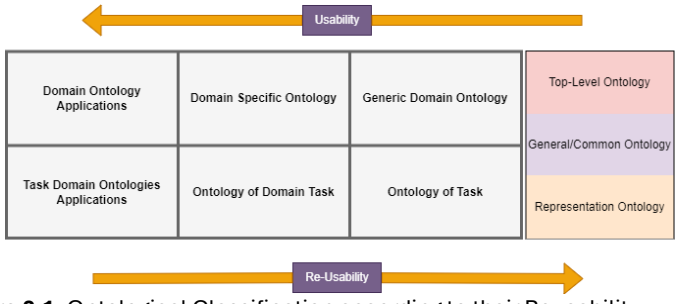


Figure 2.1: Ontological Classification according to their Reusability

Ontology languages, such as the Web Ontology Language (OWL), play a crucial role in constructing meta-descriptions or ontologies. OWL, certified by the W3C, offers enhanced capabilities compared to earlier languages, enabling not only the display but also the processing of data. It introduces a higher level of explain ability and expands the vocabulary for describing concepts, classes, properties, individuals, and restrictions within the ontology model. OWL facilitates reasoning, allowing applications to determine relationships between concepts and ensure the compatibility of ontology definitions. RDF (Resource Description Framework) serves as a widely accepted format for sharing information online. It enables easy merging of data and the development of schemas over time without necessitating updates for all data users. RDF extends the linking structure of the Web, allowing not only the naming of link endpoints but also the representation of connections between them. This

architecture supports the combination, exposure, and sharing of organized and semi-structured data across various applications, visualized as a directed, labeled graph. The use of ontologies aims to standardize information representation and sharing, enhancing our understanding of the world. [3]. Ontologies add an additional layer of abstraction to data models, providing semantic-level information that facilitates interoperability between incompatible systems. They formally characterize ideas and connections within specific domains, contrasting with database schemas that offer structural explanations. Ontologies also play a crucial role in data collection, providing versatile tools like online questionnaires applicable across various contexts.

Ontology reuse methodologies involve activities such as evaluation, matching, merging, aligning, mapping, and integration of existing ontologies. Integration processes include determining viability, specifying assumptions, identifying candidate ontologies, analyzing their relevance, and selecting suitable sources for integration. These methodologies address the challenges of ontology construction and reuse in diverse application contexts, ensuring the effective utilization of ontological resources.

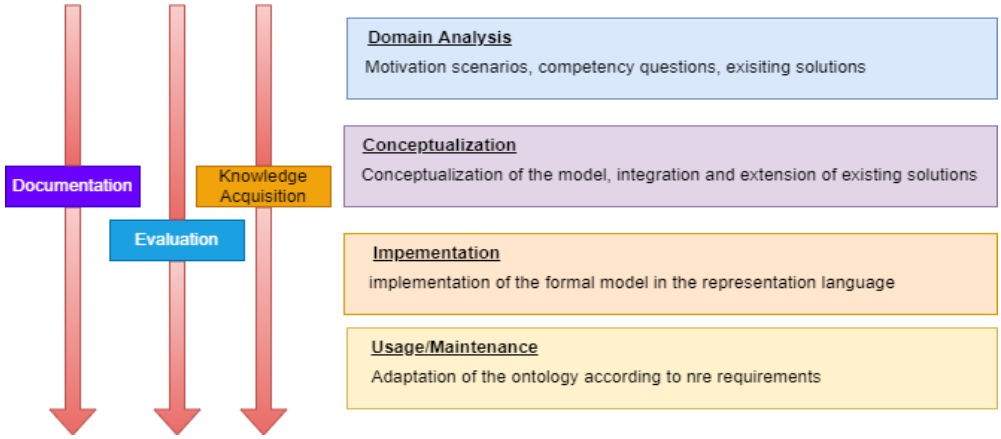


Figure 2.2: The Process of Ontology Engineering

Various methodologies for ontology development exist, each offering unique approaches and considerations. The Cyc development method focuses on manual coding, augmentation, and automation, leveraging micro theories to manage inconsistencies. Uschold and King's methodology emphasizes purpose determination, ontology capture, and integration of preexisting ontologies, employing top-down, bottom-up, and middle-out strategies. Grüninger and Fox's TOVE method employs case studies, competency question formulation, logic specification, and ontology assessment, prioritizing formal logic for computable model creation. METHODOLOGY provides a structured approach with management, development, and support tasks, emphasizing core concept definition early in the ontology construction process.

Used Approach

Our proposed model is structured around three main stages, integrating agile techniques to ensure flexibility and adaptability throughout the ontology development process. These stages are as follows:

1. **Preliminary Investigations:**

This initial stage involves conducting thorough preliminary investigations to gather relevant information and insights into the domain under consideration. Key activities include identifying stakeholders, defining project objectives, and determining the scope of the ontology development project.Sub-stages within this phase may include stakeholder interviews, literature reviews, and analysis of existing ontologies or knowledge resources.

2. **Ontology Design and Development:**

Following the preliminary investigations, the focus shifts to the design and development of the ontology. This stage involves conceptualizing the ontology structure, defining classes, properties, and relationships, and creating formal representations using ontology languages such as OWL. Sub-stages within this phase may include ontology modeling, ontology population with instances, and validation of the ontology structure against domain requirements.

3. **Ontology Evaluation:**

The final stage of the model entails evaluating the developed ontology to ensure its quality, effectiveness, and alignment with project objectives. Evaluation methods may include assessing ontology completeness, consistency, and relevance to stakeholder needs.

Sub-stages within this phase may include ontology testing, expert reviews, and usability assessments to gather feedback and identify areas for improvement. By incorporating agile techniques at each stage, our model facilitates iterative development, allowing for continuous refinement based on feedback and evolving domain requirements [5]. This iterative

approach enhances the agility and responsiveness of the ontology development process, ultimately leading to the creation of robust and adaptive ontologies that effectively represent and support the target domain.

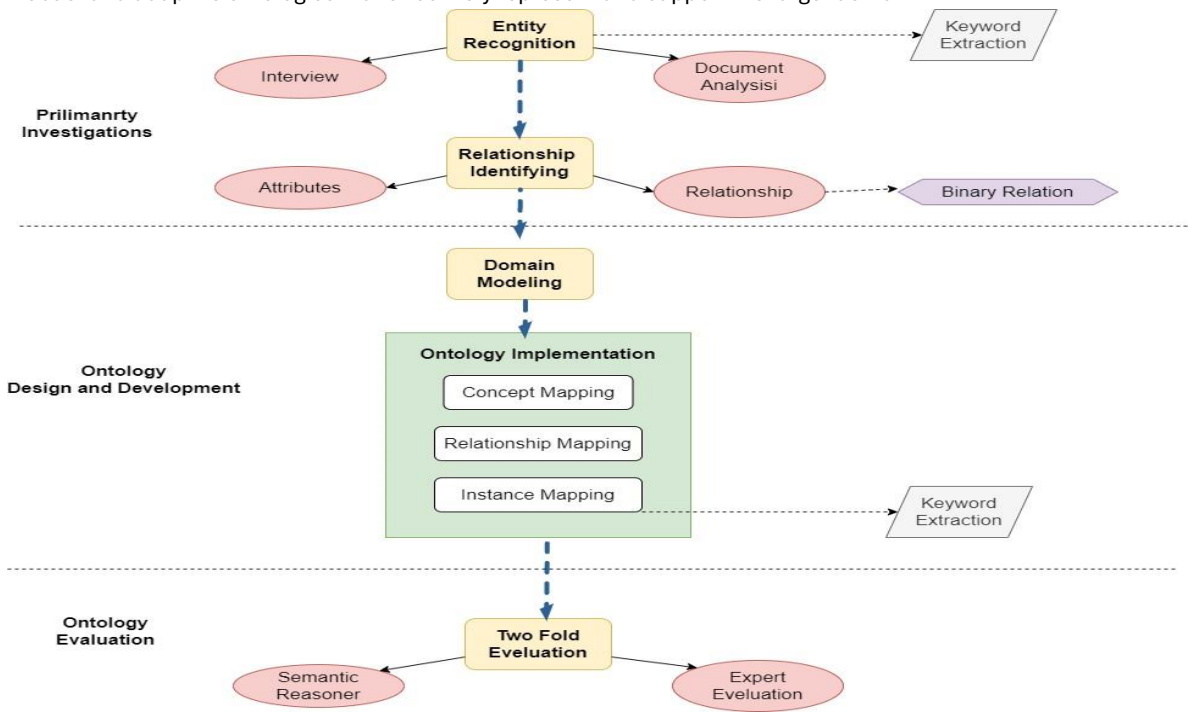


Figure 3.1: Proposed Model

Phase 1: Entity Recognition

In the initial phase of our model, entity recognition is paramount as we aim to gather relevant knowledge within the Database Domain. This involves conducting interviews to collect information and meticulously analyzing collected documents to extract key entities. Various techniques, including keyword extraction facilitated by tools like MonkeyLearn, aid in this process. Data collection begins with interviews involving Database Management experts, including Database Administrators from software companies, as well as students and teachers specializing in database studies. Insights and knowledge are gathered from diverse perspectives [6].Domain experts and end-users collaborate to assess the relevance and utility of data obtained through interviews. Since interview documents may contain extraneous information, domain experts thoroughly analyze them to identify key entities. Keyword extraction plays a crucial role in various fields, serving purposes like span detection and text analysis. The MonkeyLearn keyword extractor was used in a small document experiment, resulting in key terms illustrated in Figure 3.2.

Figure 3.2: Keyword extraction with the help of MonkeyLearn

Relationship identification

The concepts in ontologies are arranged into discrete hierarchies and are related by a variety of properties, including associations and attributes, and linkages to other ontologies, models, and data from different sources. Within the context of biorefineries, this vocabulary covers a wide range of substances and technologies applied during the building process. The classification of concepts into distinct classes (Class 1: Feedstocks, Class 2: Technologies, Class 3: Intermediates, Class 4: Products) is shown in Figure 3.3. Subcategories within the ontology comprise information pertaining to costing, applications, and life-cycle costs (LCA). The connections between the words (individuals) and the classes are defined by their properties. Features like "has output" and "is processed with" specify the relationships that help build overarching frameworks. Extra characteristics that support inference—like "may lead to," "has produces," "output," "is generated by," and "created via the phenomenon"—help users' ability to represent and retrieve information. Figure 3.4 describes the domain and range of classes' attributes.

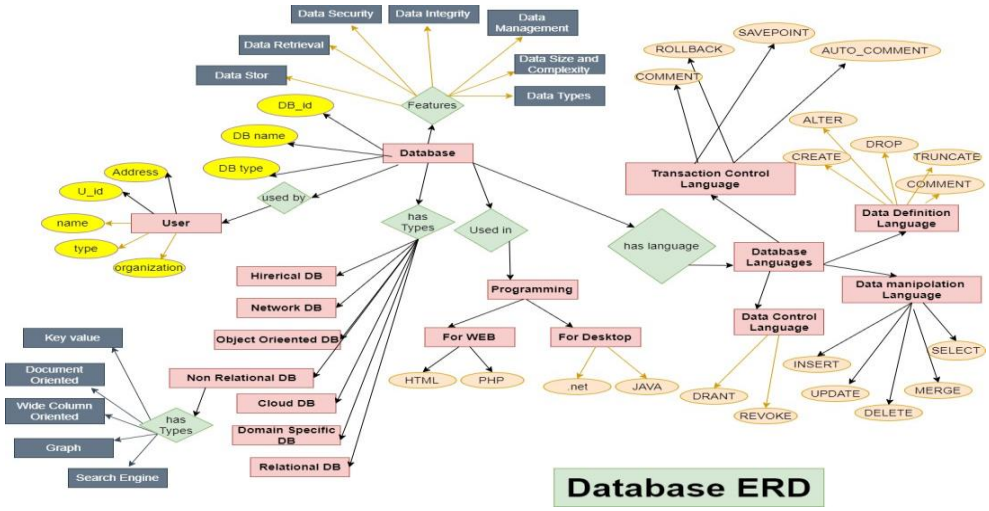


Figure 3.3: Entity Relation Diagram of Database Instances

Phase 2:

Ontology Design and Development

Our effort is split into two subprocesses at this phase: Ontology Implementation and Domain Modeling. We further divide the Ontology Implementation process into Concept, Relationship, and Instance mapping, which map our ontology components.

Transformation Framework

An entity-relationship diagram (ERD) is a tool used in SASD for data modeling. Entities and their interactions are represented in both ERD and OWL (Web Ontology Language). As such, the process of moving from the ERD to the OWL ontology is simple: the ERD's entities and attributes are translated into OWL classes and the Datatype Properties that go along with them. Two object qualities are used in the process of creating a two-way connection. Furthermore, cardinalities and modalities obtained from the ERD can be used to further describe the OWL class.

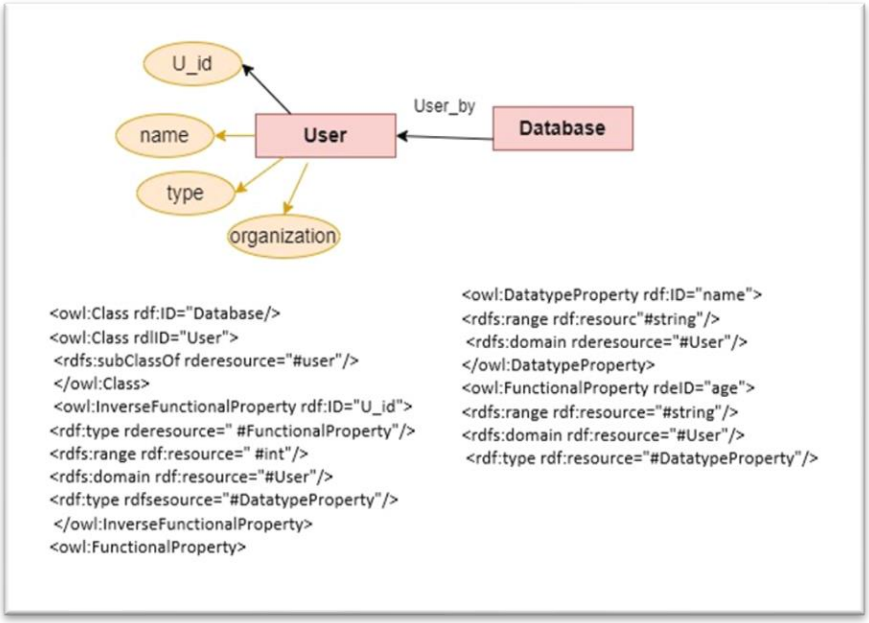


Figure 3.4: Mapping of Entity Relation to Class of OWL

Rules of Mapping Entity Relation to OWL

The process of converting an Entity Relation into an OWL ontology is described in this section. In the OWL Ontology, every Entity in the Entity Relation is translated into a class. The Entities that have been transformed into OWL classes are shown in Fig. 3.3.An entity's attributes are bits of information that need particular mapping techniques into OWL datatype characteristics. It's critical to make sure that each element has a locally unique name when transferring attributes into properties [9]. Our system automatically appends "start of attribute similar" to one of the names (Entity: Attribute) when two entities have the same name. Furthermore, range mapping of values is in charge of mapping ER domain datatypes to XSD datatypes.

In the case of Simple Attributes, this means that the attribute of an entity must be transformed into a simple datatype property in the corresponding OWL class. The "range" of an attribute (e.g., string, int) is determined by the datatype of the attribute; the entity is the "domain" of the property. Mapping ER datatypes to XSD datatypes is necessary and is accomplished by range mapping values. The "functional" tag needs to be appended to an attribute's datatype property in OWL DL in order to guarantee that it can only accept one value out of multiple possible values. This guarantees that the property can only accept a single value, as the example "Employee Name" in Fig. 3.4 illustrates.There are two ways to transfer Composite Attributes, like "Address," to OWL datatype properties.

```
<owl:FunctionalProperty rdf:ID="Organization">
<-xdfs:range rdf:resource="#string"/>
<rdf:type rdf: resource="#DatatypeProperty"/>
 <rdfs:domain rdf:resource="#User"/>
<owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="Street">
<rdfs:domain rdf:resource="#User"/>
 <rdf:type rdfsesource=" #DatatypeProperty"/>
<rdfs:range rdf:resource=" #string/>
</owl:FunctionalProperty>
```

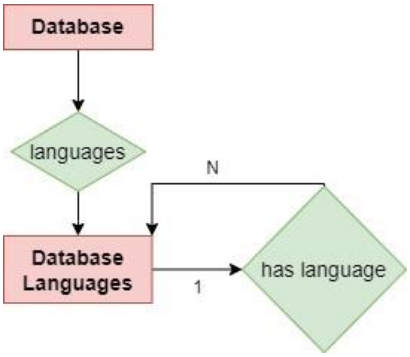
```
<owl:DatatypeProperty rdf:ID="Address">
<rdfs:domain rdf:resource="#User"/>
<rdfs:range rdtresource="#string"/>
<owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="street">
<rdf:type rdf:resource="#FunctionalProperty"/>
<rdfs:subPropertyOf rdtresource="#Address"/>
<fowl:DatatypeProperty> cowl:DatatypeProperty rdf:ID="Organization">
<rdf:type rdf:resource="#FunctionalProperty"/>
 <rdfs:subPropertyOf rdf:resource="#Address"/>
</owl:Dataty-peProperty>
```

Figure 3.5: Mapping of Attributes to Datatype Property

A second method is to map simple attributes that make up composite attributes to the subproperties of the corresponding datatype properties and the composite attributes to the datatype properties themselves. Relational Schema only contains simple, component attribute instances; composite attribute instances are ignored, hence the former is preferred when transforming relational databases. This rule caused us to lose our conceptual modeling of composite characteristics while moving from ontology to ER.

[10]. Because it breaks down the datatype property into the composite attribute and the sub property-off into the component attribute, the second method is helpful if one wants to keep this information private. The characteristics of a data type can only ever yield one value, hence they should all be categorized as "functional." The procedures needed to transform a Composite Attribute into an OWL ontology are shown in Figure 1.6.

Attribute with multiple values. multiple values This sort of attribute maps to a datatype property without the extra "functional" name, just like basic attributes do. OWL DL attributes frequently allow for a wide range of values to be entered, such as "Skill," which can be interpreted in a variety of ways depending on the person. Principal Key. The primary key attribute is transformed into a datatype object, and the labels "functional" and "inverse-functional" are applied. While the "inverse-functional" tag restricts the number of objects that a specific topic can have, one type of "functional" tag limits the number of subjects that an object can have. Relations of Subtypes (IS-A). SubType relations should be replaced in the OWL ontology from the ERD with subClassOf. The OWL ontology's generalization hierarchy is described via the OWL: subClassOf property.




```
<owl:ObjectProperty rdf:ID="Language">
<rdfs-domain rdf:resource="#Database
Languages"/>
<rdfs:range rdtresource="#Database"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_language">
<owlinverseOf rdtresource="#language"/>
</owl:ObjectProperty>
```

```
<owl:Class rdf:ID="Database Language">
<owl:Restriction><owtsomeValuesFrom>
<owl:Class rdf:about="#Database Language "/>
</owl:someValuesFrom> cowtonProperty>
<owl:ObjectProperty rdf:resourse="#used_in"/>
</owl:onProperty> </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID="used_in">
<rdfs:domain rdf:resource="#Database"/>
<rdfs:range rdf:resource="#Database"/>
```

Figure 3.6: Mapping of Relationships of Bi-Directional to the Object Properties

Bi-directional Connection. An object property at the class level represents any pairwise association between things. Due to the two-way relationship between the entities, object attributes in OWL are unidirectional while in ERD they are bidirectional, necessitating the creation of two object properties for each entity. The relationship's ERD representation is matched by the first object attribute, and its inverse is matched by the second. When "Employee" and "Department" establish a relationship named "Work," OWL will create two properties for the related objects: "Work," which has the domain "Employee" and the range "Department," and "Has ,Work," which has the domain "Work," and the range "Has Work.[13]". The mapping between the Manages property in the Employee OWL class, which has the same domain and range as the Manages relationship in the Employee entity, is shown in Figure 1.6. It can be seen in Figure 3.6. Unary Relationship M: N. Every time an existing unary link between M and N items is considered, two new OWL classes are produced. Many things are made up of multiple smaller parts. In this case, after converting the Item entity to the appropriate OWL class, we create the second OWL class, Contains, which uses some of the data from the original Item class. The OWL class is also translated into the connected Object or Connection containing attribute, as shown in Fig. 4.8. Relationship and Associative Entity attributes are mapped to the data type properties of OWL classes. Relationship 1 to many (required). Conversion of modalities and cardinalities into OWL constraints in the corresponding OWL classes. Numerous connections can become limitations, as Fig. 4.8 illustrates. Relationship of One to Many (Optional). If the corresponding class is optional, we don't place any restrictions on it. Numerous to Numerous Connections.

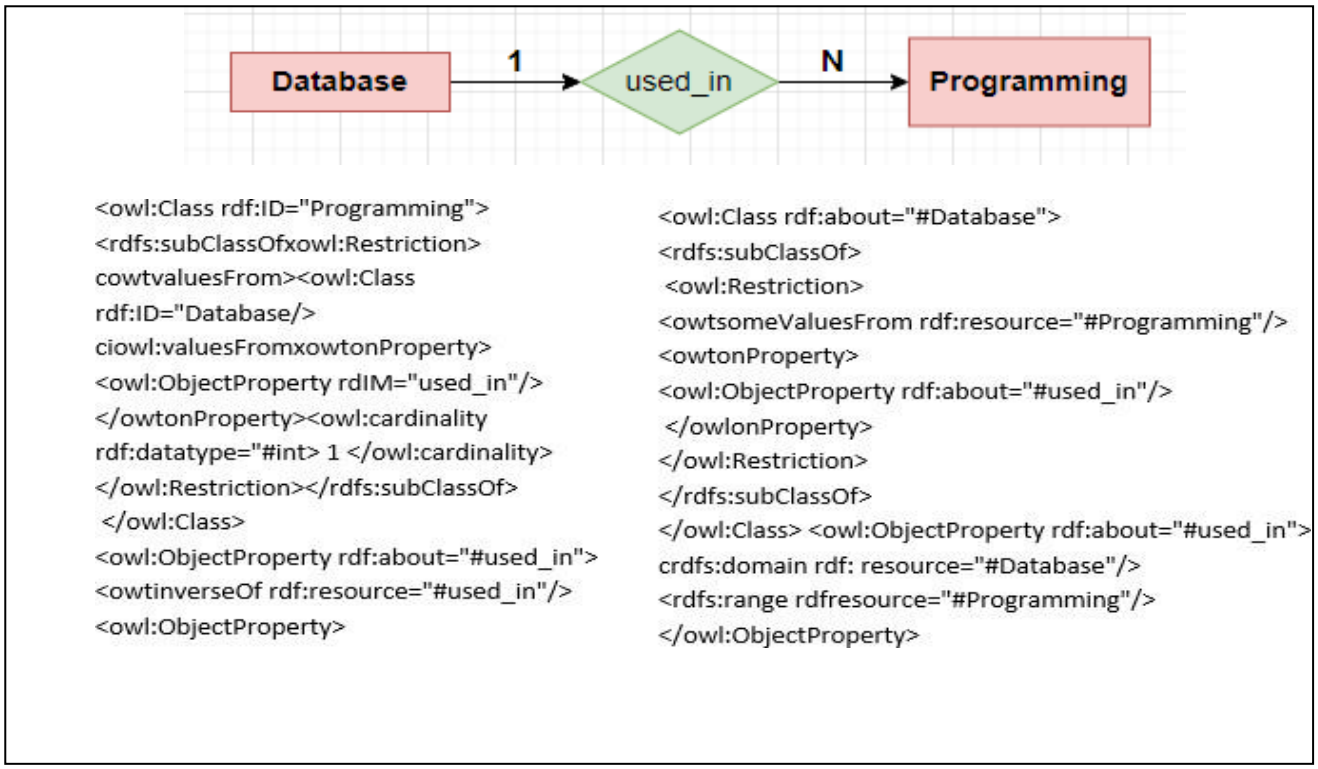


Figure 3.7: to Many Relationships Mappings

Phase 3:

In the last stage of our model, Ontology Evaluation, we assess our ontology using two distinct techniques. Initially, we assess our ontology with the assistance of domain experts, and then Pellet Reasoner will assess it.

Assessment of Ontologies

As mentioned above, we employ a two-pronged evaluation strategy in this step.
1. Subject Matter Experts
2. Reasoner Pellet

Domain Experts Evaluation

In the professional review (or accreditation or connoisseurship) paradigm, individuals with in-depth knowledge of the subject topic hold the only authority. The reviewing process may be arranged using dimensions; this is thought to have little bearing on the accuracy of the results, but it could help to standardize the evaluation and condense the review results. Experts recommend that the reviewing panel be carefully chosen, with a range of 5 to 10 people being the ideal number. This methodology generates professional evaluations of the tested artifact's suitability for application in the designated issue area (or its related evaluation criteria)
14]. This method is frequently applied before other assessment approaches have a chance to take effect in order to expeditiously improve the quality of the artifact. The findings are directed to multiple authorities in the pertinent field.

- Consistency: This criterion deals with the presence or absence of overt or covert discrepancies in the ontological material represented.
- Comprehensiveness: For an ontology to be deemed comprehensive, it must address the objective, either directly or indirectly.
- Conciseness: In addition to the above feature, conciseness calls for the removal of unnecessary definitions and the redundancy-free illustration of an ontology's application domain.
- Extendibility/sensitivity: The standards include the capacity to add new concepts to the ontology without altering the existing ones.
- Pellet Reasoner: Although consistency in ontologies is crucial, this method by itself does not provide extremely innovative applications of an ontology. For a long time, the communities involved in taxonomy and description logic have concurred that a specific combination of inference services is essential for numerous application

types and knowledge engineering endeavors to succeed. Any practical OWL reasoner must support at least the "standard" set of Description Logic inference services (which include, but are not limited to, the resolution of logical contradictions and the inference of truth values) because OWL-DL is a syntactic difference of the highly expressive Description Logic SHOIN (D):

- Checking for consistency, This ensures that there are no incompatible elements in a given ontology. Pellet utilizes the OWL Abstract Syntax & Semantics article's formal definition of ontology consistency. The "consistency check" in DL refers to the process that establishes if an ABox and a TBox are consistent with one another.
- The concept satisfiability method, which checks if a class can actually contain instances. The entire ontology collapses when an instance of an unsatisfiable class is defined.
- Classification: it establishes a complete class hierarchy by figuring out the subclass relationships among every class in a namespace. You can use the class hierarchy to get the answers to queries such as "Who are all the direct siblings of a class?"

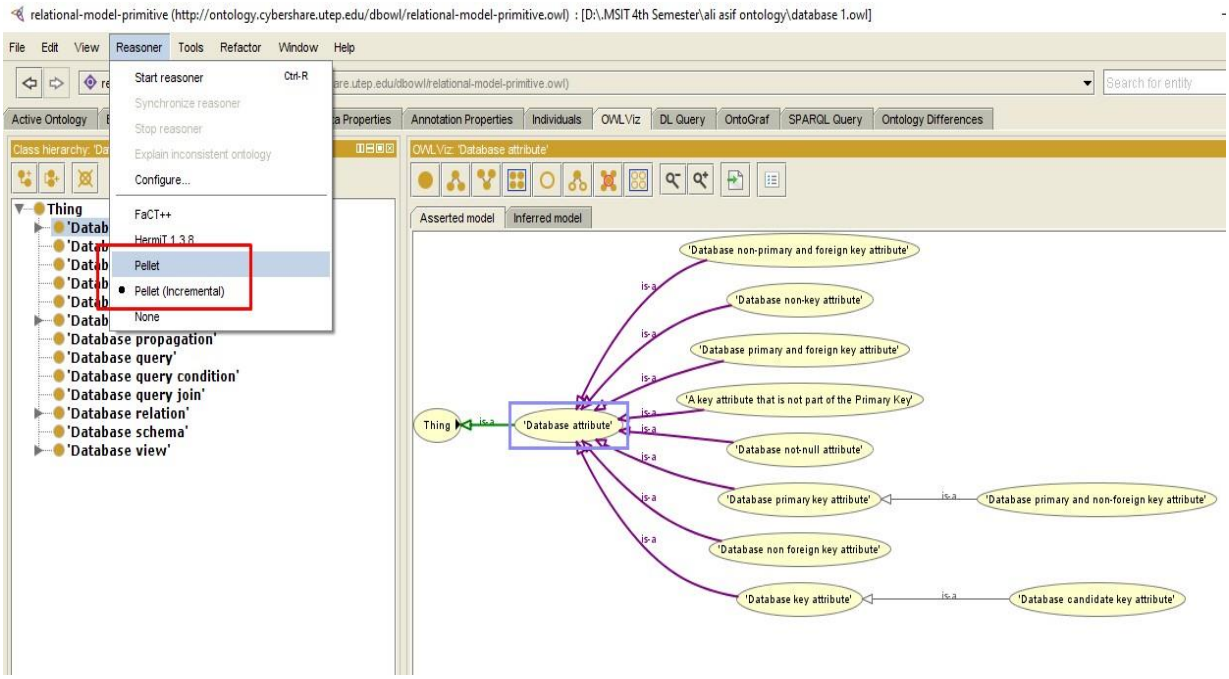


Figure 3.8: Pellet reasoner in our protégé

3. Results & Evaluation

- A free and open-source software program called Protégé OWL was created to make ontologies for the Semantic Web easier to create. In particular, it is an add-on part for the ontology editor Protégé. Protégé With the use of descriptive logic classifiers, users of the Web Ontology Language (OWL) can make sure that their ontologies remain consistent while they are changed. Protégé OWL can help with classification tasks, which are a lot of the problem-solving tasks that intelligent application developers in the biomedical field desire to automate.
- Protégé OWL gives customers the ability to work on their own applications while using state-of-the-art classifiers and other knowledge representation formats, like OWL. Because of the integration between the OWL Plug-tight in and Protégé, users can benefit from the latter's main features, such as its graphical user interfaces, customizable storage options, and extensive collection of data collection and visualization tools. Ultimately, Protégé OWL provides an API for integrating it with pre-existing software. Protégé OWL is a great option for developing OWL ontologies and the intelligent applications that utilize them because of its extensive feature set.
- APIs and graphical user interfaces (GUI). Based on the Protégé frame-based knowledge architecture (individuals), Protégé OWL's graphical user interface (GUI) manages class, slot (property), and instance editing. Its application programming interface (API) allows programmers to include Protégé OWL into their own applications.

- The Logical OWL Expressions graphic editor. Protégé OWL comes with an easy-to-use expression editor that lets users quickly create expressions with the keyboard or mouse. Furthermore, it utilizes a graphical object-oriented representation of the custom and default classes. This editor lets you cut, copy, and paste.

Entities Representation

The ideas that make up our ontology are shown in the Figure 4.1 that can be seen below.

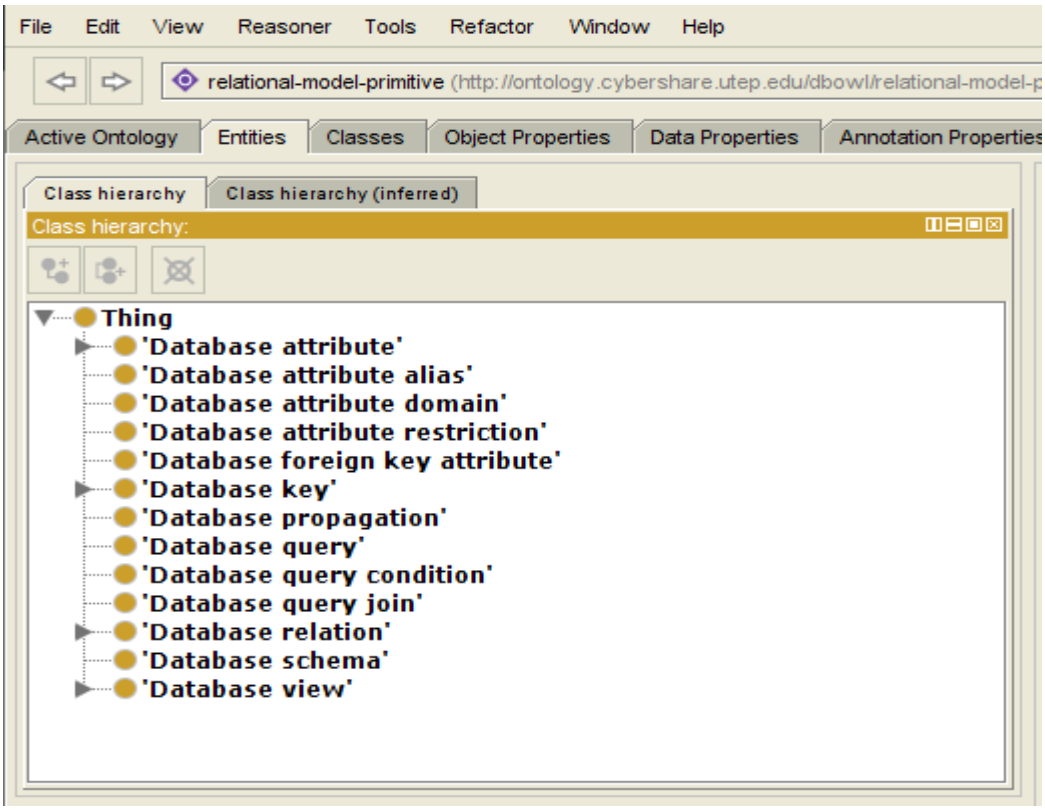


Figure 4.1: Entity Representation

Object Properties

A new window displaying the selected property's stated features opens when you click on one of the object properties. A series of radio buttons represent the features. A feature that has a checkbox next to it means that at least one ontology in the current ontology's imports closure has asserted it. A characteristic is not claimed in any taxonomy in the imports closure of the active ontology if its box is left unchecked. The concepts of our ontology are displayed in Fig. 4.2 below.

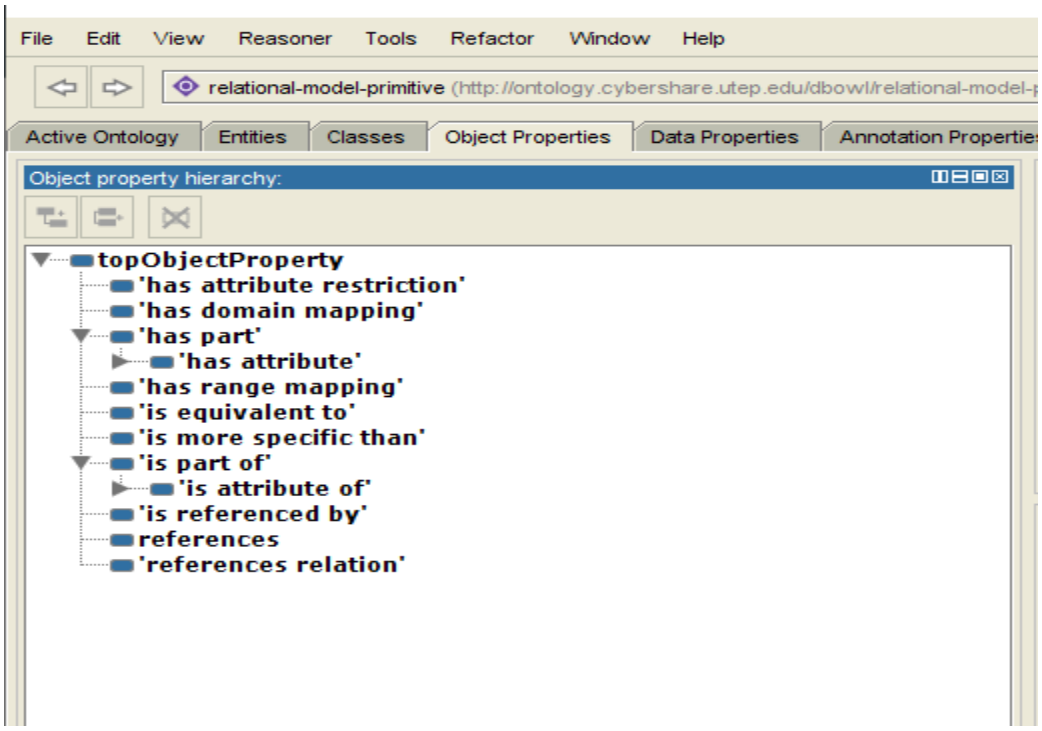


Figure 4.2: Object Properties

Data Properties

Within Protégé, the claimed data value hierarchal structure view functions as a fundamental component of the navigational system. It is laid down in the form of a tree, with the nodes of thetree representing different data attributes. Data properties that are sub properties of a parent node are represented by child nodes. Our ontology's data attributes are shown in Figure 4.3 below.

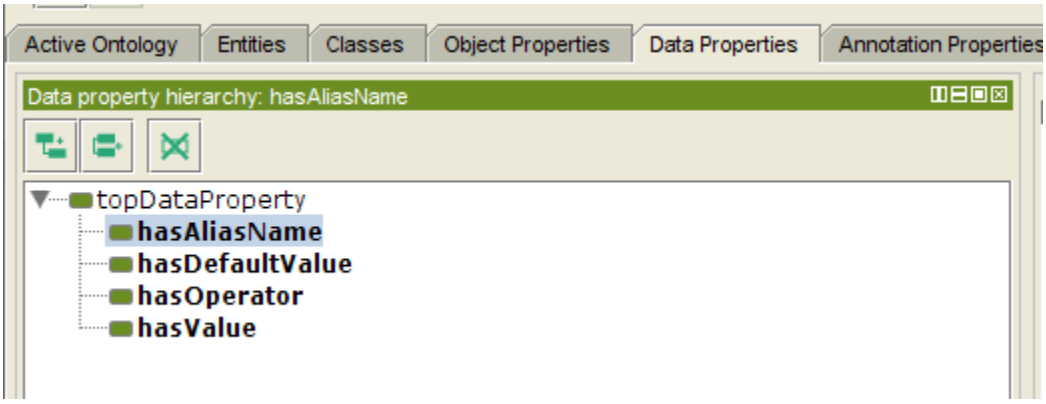


Figure 4.3: Data Properties

OwlViz **Illustration**

OWLViz performs best when used with the Protege-OWL editor. It enables the user to compare the declared class hierarchy with the inferred class hierarchy by letting them view and explore the class hierarchical structures in an OWL Ontology step

by step. The same color scheme is used by the Protege-OWL editor plugin OWLViz, which makes it simpler to identify discrepancies in your model, notice deliberate changes to the class hierarchy, and distinguish between primitive and specified classes. OwlViz beautifully illustrates our ontology, as seen in Figs. 4.4, 4.5, and 4.6 below.

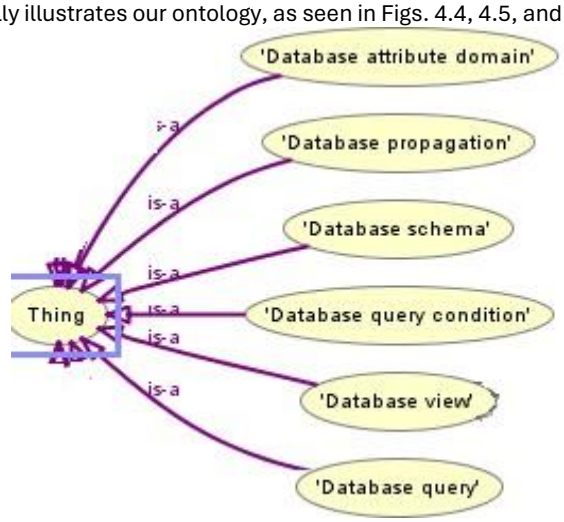


Figure 4.4: Owl Viz Representation of Database instances

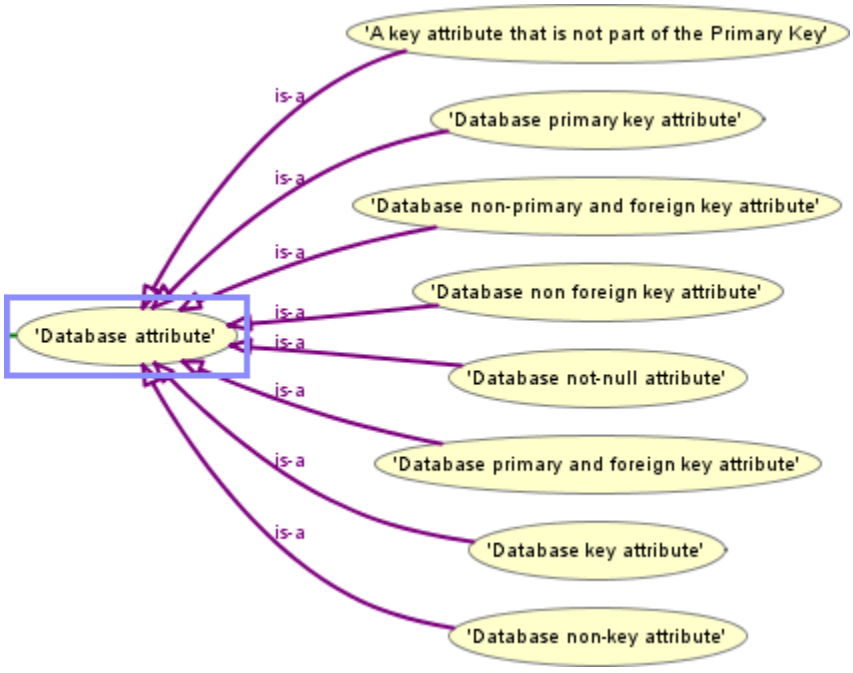


Figure 4.5: Owl Viz Representation of Database Attributes

Figure 5.6: OwlWiz Representation of Database Relations

Moving On to Graph Illustration

To investigate the relationships among your OWL ontologies, Onto Graph offers interactive navigational capabilities. You can use one of several templates to automatically organize the structure of your ontology. Supported connection types include subclass, unique, domain/range, object-property equivalency, and equivalency. By removing the connections and node kinds you are not interested in, you can create whatever kind of view you desire. Figure 4.7 below shows our ontology represented as an ontograph.

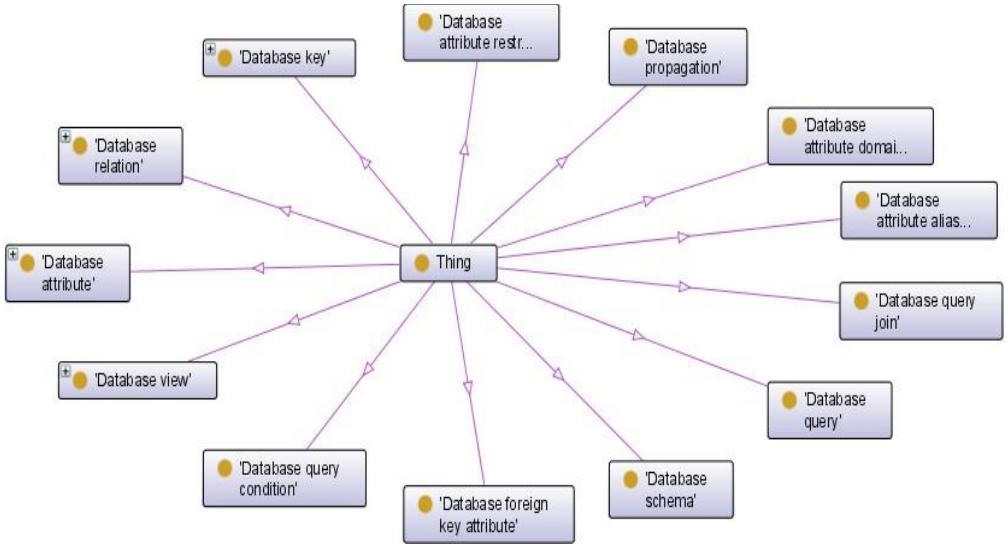


Figure 4.7: Onto Graph Representation

Assessment of Ontologies

Ontologies are among the most important of the numerous technologies that comprise the Semantic Web. Ontologies define the definitions and connections between terms used in data description. According to Gruber (1993), their "explicit statement of a notion" is beneficial. Ontologies have become the foundation for a wide range of useful services in recent years, such as DM, QA, and IR. One type of technology used to store complex data so a computer system can access it is called a knowledge base. Finally, the biggest problem they encounter is the validation of large ontologies. These two viewpoints take into account a number of parameters including the following:

- An ontology's degree of accuracy is determined by how well it defines and characterizes its classes, traits, and individuals.
- Completion: The degree of completion denotes how well the subject matter is covered in this ontology.
- Conciseness: We may assess if an ontology is excessively long by examining if it includes any information unrelated to the subject at hand.
- Adaptability: An ontology's ability to forecast its future uses is a gauge of its adaptability. Your ontology should serve as the theoretical foundation for a wide range of upcoming projects.
- Clarity: We assess the clarity of ideas to determine how effectively they are defined by the ontology. Definitions that are impartial and independent of the context are what we require.
- Computational efficiency: This is a measure of how quickly reasoners may finish tasks outlined in an ontology when assessing the ontology.
- Consistency: The absence of inconsistencies in the ontology indicates that it is consistent.

To evaluate the quality of our ontology, we employ a dual criterion assessment methodology. In this section, we evaluate our ontology's quality using resources like as Expert Reviews and Pellet Reasoner.

Pellet Reasoner

Architecture Pellet

Pellet's main components are all displayed in Figure 4.8. Fundamentally, Pellet functions as a Descriptive Logic reasoner based on tableaux. All other reasoning services are reduced to consistency checking for a knowledge base because of the tableaux reasoner. The structure of the reasoner makes it possible to load different tableaux algorithms into it. Although the default method handles SROIQ(D), there are other tableaux methods that can be used, such as those for handling non-monotonic extensions and integrating with rules.

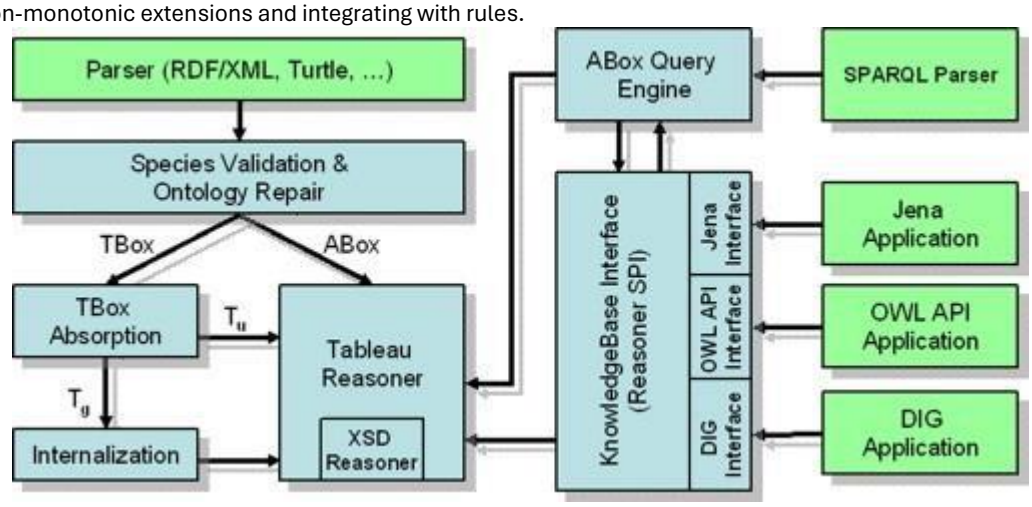


Figure 4.8: Main components of the Pellet Reasoner

An OVL management tool with an integrated module has been built to make the most of the services offered by Pellet. Both the OWL API toolkit and the Pellet open source APIs (version 2.1.1, published on May 3, 2010) play significant roles in this section.

The results of pellet reasoner are shown in the following figure 4.9.

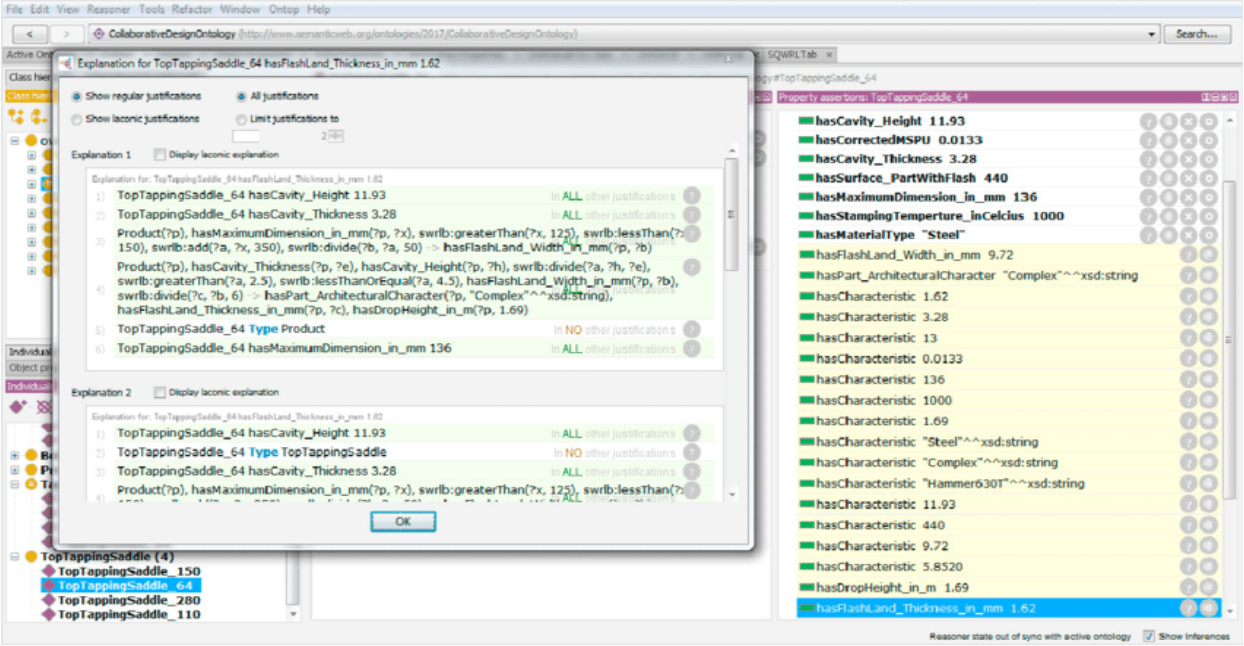


Figure: 4.9: Pellet Reasoner Results

Furthermore, experts review the identified language (i.e., concepts, relationships) in the new ontology to assess the effectiveness of the prototype. So far, twenty persons have really taken part in the assessment: ten research students and academics from two separate universities, respectively. An ER schema and the accompanying produced ontology as a result of the system prototype were handed to each group of experts so they could evaluate the four fundamental components of ontology: (1) concepts, (2) data property, (3) object property, and (4) constraints. To further avoid bias, group assessments were combined with one another. Our computations of accuracy figures will enable us to assess the prototype model's performance. Since precision measures how accurately domain knowledge is modeled, it is an important metric.

$$Precision = \frac{valid\ \ number\ of\ T\ extracted}{Total\ number\ of\ T\ extracted}$$

Eq. 1

Our suggested ontology's accuracy values are laid forth in simple fashion in Table 4.1.

TABLE. 4.1. PRECISION AND RECALL OF EXTRACTED VOCABULARY

Vocabulary	**Evaluation Measures**
	Precision
Concepts	0.98
Data properties	0.96
Object properties	0.96
Constraints	0.93
Average Results	0.9575

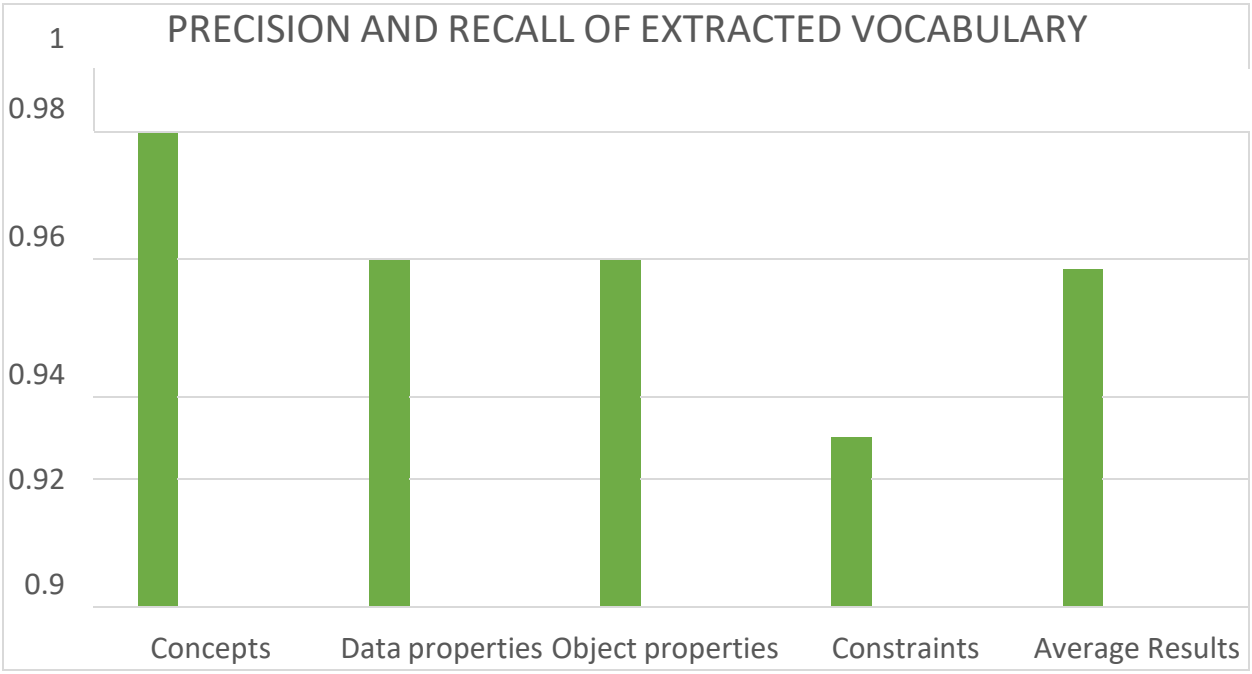


Figure 4.10: Precision And Recall of Extracted Vocabulary

Comparison of different ontologies and their ontology engineering methodologies

This section compares the major domain ontologies and their methodologies. It includes only those ontologies which are documented. We attempted to include ontologies of different periods and different domains.

No	Name of the ontology	Author	Year	Methodology used
1	TOronto Virtual Enterprise ontology (TOVE)	(Gruninger, M., and Fox, 1995)	1995	Gruninger & Fox Methodology
2	The Reference Ontology	(Arpírez, Gómez-Pérez, Lozano-Tello, & Pinto, 2000)	(1998)	Methontology
3	Knowledge Acquisition Ontology	Blázquez, J. Fernandez,	(1998)	Methontology
		M. García-Pinar, J. & Gómez-Pérez, A.		
4	Chemical ontology	Fernadez-Lopez, M., et. al	(1999)	Methontology
5	Environmental pollutants ontology Rojas-Amaya, 1999)	(Asunción Gómez-Pérez and Dolores	1999	Methontology

6	Legal Ontology	(CORCHO, O., 2002)	2002	Methontology
7	Pizza ontology	(Drummond, N., Horridge, M., Stevens, R., Wroe, C. and Sampaio, 2005)	2005	No clue on methodology
8	Wine Ontology	(Graca, J., Mourao, M., Anunciacao, O., Monteiro, P., Pinto, H.S. and Loureiro, 2005)	2005	Enterprise Ontology+ Methontology
9	Information Science Ontology	Sawsaa, A. & Lu, J	(2010)	Methontology
10	Beer Ontology	(Heflin, 2012)	2012	No clue on methodology
11	Quran ontology for Juz' Amma	(Iqbal, Mustapha, & Yusoff, 2013)	2013	ontology merging approach
12	Textile chemical ontology	(Ferrero & Lloret, 2014)	2014	Methontology
13	Food Ontology	(Dutta, Chatterjee, & Madalli, 2015)	2015	YAMO

Conclusion & Future Work

The paper begins by exploring Ontology Learning, a burgeoning field with significant growth potential. Current efforts primarily focus on extracting key ideas and basic linkages, but future advancements aim to capture more nuanced aspects of ontology and complex interactions between concepts like attributes, axioms, and constraints. While existing systems typically accept text input, there's potential to incorporate structural information from other sources like databases. Currently, ontology creation processes often require manual intervention, particularly in naming relations, but future solutions may automate these tasks. Ontology reuse, although less defined than Ontology Learning, presents significant opportunities. Methods for translating and comparing ontologies exist, but they often overlook aspects beyond conceptualizations. Future work will likely involve comparing characteristics, axioms, and entire ontology structures. Effective reuse also hinges on locating suitable ontology components, which can be facilitated by taxonomy libraries and top-level schemas. However, ontology developers still face challenges in determining which elements to reuse due to uncertainty about interdependencies and the lack of readily available, vendor-provided ontology components. Case studies highlight the organic relationship between ontologies and Software Patterns, revealing parallels between ontology graph structures and diagrammatic representations of analysis patterns. Exploring these connections and developing ontology patterns could be a fruitful direction for future research. Additionally, investigating similarities between Ontology Development and Software Development, as proposed by Devedzic, could foster collaboration and communication between the two fields, benefiting areas like development approaches, patterns, and component-based engineering.

Conclusion

The exploration thus far has highlighted numerous subfields within each major category that remain unexplored. Future efforts will focus on understanding how these areas intersect to enhance the utility of ontology patterns for teaching and ontology reuse. Vladan Devedzic notes in [17] that Software Development and Ontology Development share more

similarities than commonly perceived, suggesting potential benefits from collaboration between the two domains. Our research primarily delves into business ontologies and smaller-scale applications, emphasizing the need for intuitive or automated ontology construction accessible to domain experts who lack specialized ontology knowledge. The overarching goal is to enhance opportunities for ontology reuse and streamline the Ontology Development lifecycle. These objectives were identified as promising areas for further investigation, with ontology patterns serving as valuable guides for construction and database organization.

Furthermore, knowledge exchange with other disciplines, particularly Software Engineering, is essential for enriching our understanding and approaches in ontology development. Collaboration with diverse fields can offer fresh perspectives and insights, contributing to the advancement of ontology-related practices.

References

1. Asunción Gómez-Pérez and Richard Benjamins. “Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods”. In: IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings. 1999.
2. S. M. Cahn, editor. *Classics of Western Philosophy*. Hackett Publishing Company, 6 edition, 2002.
3. M. Heidegger. *Ontologie. Hermeneutik der Faktizitat (Frhe Freiburger Vorlesung Som- ¨mersemester 1923)*. Klostermann, 1988.
4. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.
5. R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):35–56, 1991.
6. N. Guarino. Formal Ontology and Information Systems. In *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems FOIS1998*, pages 3–15. IOS-Press, 1998.
7. T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing.

   International Journal of Human-Computer Studies, 43(5/6):907–928, 1995.
8. R. Studer, V. R. Benjamins, and D. Fensel. Knowledge Engineering Principles and Methods. *Data and Knowledge Engineering*, 25(1/2):161–197, 1998.
9. Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “Methontology: from ontological art towards ontological engineering”.In: (1997)
10. Simone Braun et al. “Ontology Maturing: a Collaborative Web 2.0 Approach to OntologyEngineering.” In: *Ckc* 273 (2007).
11. Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “Methontology: from ontological art towards ontological engineering”.In: (1997)
12. Nabil Mohammed Ali Munassar and A Govardhan. “A comparison between five models of software engineering”. In: *IJCSI* 5 (2010), pp. 95–101.
13. Kent Beck. *Extreme programming explained: embrace change*. addisonwesley professional, 2000.
14. Sören Auer and Heinrich Herre. “RapidOWL—An agile knowledge engineering methodology”. In: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer. 2006, pp. 424–430.
15. Holger Knublauch. “An agile development methodology for knowledgebased systems including a Java framework for knowledge modeling and appropriate tool support”. PhD thesis. Universität Ulm, 2002.
16. F. Fonseca. The double role of ontologies in information science research. *Journal ofthe American Society for Information Science and Technology*, 58(6):786{793, 2007.
17. Gaševic, D., Djuric, D., & Devedžic, V. (2009). Model driven engineering and ontology development. Springer Science & Business Media.