JOHANNES KEPLER
UNIVERSITY LINZ

Author
Moritz Haider
k12215632

Submission
Institute for Application-oriented Knowledge Processing

Thesis Supervisor
a.Univ.-Prof. DI Dr.
Wolfram Wöß

February 2026

Annotating Ontologies With Metadata

Bachelor's Thesis

to confer the academic degree of

Bachelor of Science

in the Bachelor's Program

Computer Science

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Abstract

The Semantic Web aims to enhance data interoperability and knowledge exchange by providing machine-readable semantics for information on the web. Ontologies play a central role in this ecosystem, as they formally define domain concepts and their interrelations, thereby enabling shared understanding and automated reasoning. However, the discovery, reuse, and maintenance of ontologies remain challenging tasks. One major limitation is the lack of standardized and comprehensive metadata vocabularies for annotating ontologies. This thesis addresses the problem of non-standardized ontology metadata annotation by analyzing existing vocabularies, identifying common practices, and developing an extended vocabulary that combines and reuses established metadata standards. The proposed vocabulary aims to support interoperability and provide a structured approach for ontology documentation and reuse. Through this analysis and vocabulary design, the thesis contributes to the standardization and improvement of ontology metadata annotation practices within the Semantic Web community.

Contents

Abstract ii

List of Tables iv

List of Figures v

1 Introduction 1

2 Fundamentals 3
- 2.1 Semantic Web . . . 3
- 2.2 Metadata . . . 5
- 2.3 Data Catalogs . . . 6
- 2.4 Ontologies . . . 6
  - 2.4.1 Main Purpose . . . 7
  - 2.4.2 Interoperability . . . 7
  - 2.4.3 Core Concept . . . 8
  - 2.4.4 Representation Methods And Web Standards . . . 8
  - 2.4.5 Vocabularies . . . 15
  - 2.4.6 Documentation Tools . . . 18

3 Design Of A Metadata Vocabulary For Ontologies 20
- 3.1 Analysis Of Existing Metadata Vocabularies . . . 20
- 3.2 Use Of Those Current Existing Vocabularies . . . 22
- 3.3 Initial Design Considerations . . . 24
- 3.4 Discovering Concrete Properties . . . 26
- 3.5 Structure Of The Vocabulary . . . 26
- 3.6 The Resulting Vocabulary . . . 36

4 Trends - Conclusion And Future Work 37
- 4.1 Trends . . . 37
  - 4.1.1 Are Knowledge Graphs “Successors” of Ontologies? . . . 38
  - 4.1.2 Are Ontologies Outdated? . . . 38
  - 4.1.3 The Current Role Of Knowledge Graphs . . . 38
- 4.2 Conclusion . . . 39
- 4.3 Future Work . . . 40

Bibliography 41

Appendix 44
- Classes . . . 44
- Selection Of Properties . . . 46

List of Tables

2.1 Purpose And Typical Terms Of Commonly Used Metadata Vocabularies 16
2.2 Strengths And Limitations Of Commonly Used Metadata Vocabularies 17

List of Figures

3.1 Frequently Used Metadata Vocabularies 22
3.2 First Draft Of Vocabulary Structure 25

Chapter 1

Introduction

The Semantic Web represents an evolution of the current web, extending it from a web of documents to a web of data. It's central goal is to enable machines to understand and process information by encoding the semantics of data in standardized, machine-interpretable formats. Ontologies form the backbone of this infrastructure, providing formal and explicit specifications of conceptualizations that describe entities within a given domain and the relationships between them. Through their use, heterogeneous data sources can be linked, integrated, and semantically enriched, enabling knowledge inference and interoperability across systems [11] [5].

To describe and exchange such knowledge, ontology representation languages and data models such as the Resource Description Framework (RDF), RDF Schema (RDFS), and the Web Ontology Language (OWL) have been established [1] [2] [3]. These form the technical foundation for defining classes, properties, and individuals as well as the logical axioms that capture semantic relationships. Alongside these representations, vocabularies — sets of shared, reusable properties and classes — enable the consistent annotation of both data and ontological resources. Well-known examples include Dublin Core Terms (DCTERMS), the Vocabulary of Interlinked Datasets (VoID), schema.org, and the Provenance Ontology (PROV-O) [4] [5] [6] [7].

Despite these advances, the annotation of ontologies themselves with metadata remains insufficiently standardized. Classic metadata such as creator, version, license, or domain coverage are essential for assessing the quality, provenance, and reusability of an ontology. Nevertheless, ontology designers often use different vocabularies or apply properties inconsistently, resulting in reduced interoperability and discoverability. Consequently, automated documentation and ontology discovery become difficult, as metadata cannot be reliably interpreted by tools or search services. This gap is particularly relevant for ontology documentation tools such as WIDOCO or pyLODE, which rely on embedded metadata to generate human-readable descriptions [8] [9] . Although these tools facilitate documentation generation, they offer limited guidance on how to annotate metadata correctly and comprehensively [8].

[1] https://www.w3.org/RDF/
[2] https://www.w3.org/TR/rdf-schema/
[3] https://www.w3.org/OWL/
[4] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[5] https://www.w3.org/TR/void/
[6] https://schema.org/
[7] https://www.w3.org/TR/prov-o/
[8] https://dgarijo.github.io/Widoco/
[9] https://pypi.org/project/pylode/

The absence of a unified standard for ontology metadata annotation motivates the central objective of this thesis: the development of a vocabulary that enables consistent, interoperable, and semantically precise annotation of ontologies. The aim is to capture all conceivable metadata that someone could annotate in the resulting vocabulary. Three main design principles are followed: the reuse of existing, well-established vocabularies wherever possible, the extension of these vocabularies to address ontology-specific metadata needs, and adherence to the FAIR data principles. The resulting vocabulary aims to improve ontology documentation, discovery, and reuse within the Semantic Web ecosystem.

The thesis is structured as follows: Chapter 2 provides an overview of basic concepts of the semantic web, while Chapter 3 presents a detailed analysis of existing metadata vocabularies and their applicability to ontology annotation, followed by the design and implementation of the proposed vocabulary. Finally, Chapter 4 concludes the thesis by summarizing the results and outlining directions for future work, as well as provide an outlook at current topics such as knowledge graphs.

Chapter 2

Fundamentals

2.1 Semantic Web

A student is working on a thesis about the impact of climate change on urban environments. To collect relevant data, the student enters the keywords "urban heat island," "temperature dataset" and "Vienna" into her preferred search engine. The system returns thousands of links, many of which contain irrelevant or incomplete information. The student filters through numerous pages, comparing data formats, verifying sources and identifying those that can be accessed via open APIs or properly cited in her research. After nearly half an hour, a usable dataset is assembled, though uncertainty remains regarding potentially overlooked sources.

Could this process not be simpler, faster, and more efficient?

Or, for instance, a cybersecurity manager in a multinational research consortium faces an unexpected breach affecting several partner institutions, including universities, startups, and cloud service providers. Within minutes, he must assess dependencies, identify the affected components, and coordinate an international response across time zones and technical domains. The required information exists, distributed across public registries, institutional databases, and technical documentation, but integrating it on the fly demands a level of semantic interoperability that today's web does not yet provide. This example was adapted from [18].

The necessity for meaning-aware, machine-interpretable information is the central motivation behind the vision of the Semantic Web. According to Tim Berners-Lee, the inventor of the World Wide Web, a steadily growing community of researchers, developers, and practitioners, the answer to whether such integration is possible, is an emphatic yes [5]. The fundamental principle of the Semantic Web is to enhance web content so that it can be interpreted not only by humans, but also by machines, thereby enabling automation at the level of meaning. Achieving this objective requires the formal representation of knowledge and relationships, allowing machines to reason about data rather than merely retrieve it. The Semantic Web differs substantially from prior approaches. Efforts to endow machines with an understanding of semantics are not new, artificial intelligence has long pursued automated reasoning and knowledge representation. However, earlier AI systems were generally confined to isolated, narrowly defined domains. Similarly, techniques for the automatic or semi-automatic extraction of semantics from unstructured data predate the Semantic Web, though they now serve as critical components in

its implementation. Such methods facilitate the semantic annotation of large volumes of information far more efficiently than manual modeling alone could achieve. What is novel about the Semantic Web is the integration of these approaches, combining distributed and heterogeneous data sources with shared ontological models and open standards to ensure interoperability. This integration enables meaning to be shared and processed across system boundaries, resulting in a decentralized web of linked data built upon a minimal set of universally adopted standards such as RDF, OWL and the SPARQL Protocol And RDF Query Language (SPARQL), through which content creators can explicitly describe the semantics of their information. Equally important is the decentralized nature of both the initiative and its architecture. The Semantic Web is not governed by a central authority but operates as a collaborative movement. Researchers, developers, and organizations contribute ontologies, datasets, and tools independently, adhering to common specifications. In this way, interoperability is achieved without central coordination, reflecting the distributed ethos of the original World Wide Web. The Semantic Web thus represents an evolutionary development: a meaning-oriented layer of the Web that aims to render the world's information not only accessible but comprehensible to both humans and machines [18].

To explain or define the term "Semantics" more precisely: semantics is the study of linguistic meaning. It examines what meaning is, how words get their meaning, and how the meaning of a complex expression depends on its parts [1]. Substantially is the distinction between sense and reference, whereby "Sense" is given by the ideas and concepts associated with an expression while "Reference" is the object to which an expression points. Very important is the comparison to syntax, which studies the rules that dictate how to create grammatically correct sentences, in general, it is commonly understood to be a system of rules for combining elementary characters to form composite characters in natural or artificial character systems [2] as well as pragmatics, which investigates how people use language in communication [3]. In computer science, syntax refers to a set of rules for generating programs or documents with specific properties, e.g., valid Extensible Markup Language (XML) documents, whereas semantics refers to the meaning of words or characters (strings) and their relationships to each other. Semantics, together with syntactics and pragmatics, is a part of semiotics. In this respect, the term "Semantic Web" is actually too narrow, since the current debate takes all three aspects into account [18]. Following John F. Sowa [23], it is obviously a matter of the "Semiotic Web": The Internet is a giant semiotic system.

[1] https://en.wikipedia.org/wiki/Semantics
[2] https://de.wikipedia.org/wiki/Syntax
[3] https://en.wikipedia.org/wiki/Pragmatics

2.2 Metadata

Web content is primarily prepared and formatted for human consumption, information expressed using Hypertext Markup Language (HTML) or as a Portable Document Format (PDF) generally suffices for human comprehension, even though the necessary meta-information, such as syntactic or contextual data at the semantic level is not explicitly provided. For machines like a web crawlers, it is initially unclear that a string such as "Mo 21.12.2006, 9:30" represents a date [18].

Metadata are often described as "data about data," yet in the context of the Semantic Web that phrase hides considerable complexity. In essence, metadata are descriptive, structural, and semantic annotations that make information about resources explicit in such a way that both humans and machines can interpret it. Through metadata, the meaning of data can be expressed, thereby relating directly to semantics in the context of the Semantic Web [1]. A more precise definition sees metadata as information that explicates the semantics, provenance, context, structure, or relations of data, enabling discovery, interoperability, validation, and reuse. Metadata may describe who created a resource, when, where, what format, what parts or components it has, how it relates to other resources, and what constraints apply. In the Semantic Web, metadata are not simply passive labels or catalog entries, but active building-blocks: they provide the means by which machines can understand meaning, align resources to ontologies, infer relationships, assure quality, and integrate data across heterogeneous systems. Because metadata can vary in granularity and formality, they range from simple descriptive metadata, e.g., an author, title or date, to richly structured schemas that define relationships, constraints and logical axioms. Another aspect of metadata is their role in enabling interoperability: without shared understanding of what metadata terms mean, data from different sources remain isolated. Semantic metadata thereby require both standardization, or agreement within communities, of vocabulary and formats, as well as tools that allow annotation, validation and transformation of metadata into forms usable by machines. Moreover, metadata must often include semantics beyond the explicit, for instance, metadata vocabularies may implicitly or explicitly incorporate class hierarchies, property domains and ranges, constraints, allowing machines to deduce implicit knowledge from explicit metadata.

Based on the research from Wilkinson et al. [26], metadata in the semantic web should always comply with the FAIR principle, which is an internationally recognized concept and generally describes how data should be designed so that they are findable (F - "Findable"), accessible (A - "Accessible"), interoperable (I - "Interoperable"), and reusable (R - "Reusable"). Metadata should be structured and indexed in such a way that it can be easily found by humans and machines, for instance by using a unique, persistent identifier such as DOI. The data should be accessible via standardized protocols, access can be public or restricted, but the metadata itself should always be available. Moreover it's crucial that metadata use standardized formats in order to enable integration and combination with other data sources, as well as the use of sufficient descriptions to enable reusing the data by other applications, including a clear documentation of facts like the origin, context or license terms [20].

2.3 Data Catalogs

In 2020, data catalogs were defined as ”a tool to centrally collect, create and maintain metadata” by Quimbert et al. [9], thereby constituting structured inventories of data assets within an organization, typically providing metadata such as schemas, provenance, quality indicators and access policies to support discoverability, governance and reuse. Their conceptual foundation often parallels that of ontologies, as both rely on explicit semantic descriptions to organize and contextualize information. While data catalogs primarily focus on operational metadata management, ontologies offer formalized, machine-interpretable models that enable consistent semantic interpretation across heterogeneous datasets. Consequently, ontologies can serve as the semantic backbone of data catalogs by supplying controlled vocabularies, domain models, and relationship structures that enhance interoperability, facilitate metadata integration, and support advanced querying or reasoning over cataloged resources [14].

2.4 Ontologies

While philosophical ontology investigates the fundamental nature of being, categorizing entities and their interrelations within reality, the concept of ontology in the Semantic Web context serves a distinct, operational purpose. Philosophically, ontology addresses questions of existence and the most general structures of reality, focusing on what entities exist and how they can be systematically classified [22].

In contrast, Semantic Web ontologies, as defined by Gruber in 1993, are “explicit specifications of conceptualizations” [11]. They were developed within Artificial Intelligence and are intended to formally represent domain knowledge in a manner that is interpretable by machines and reusable across software applications. These ontologies describe concepts and relationships within a domain, enabling automated reasoning and interpretation rather than merely presenting information for human consumption. The term “ontology” thus encompasses a spectrum of objectives: from providing machine-readable structures for data automation to supporting humans in complex, knowledge-intensive tasks such as knowledge management.

2.4.1 Main Purpose

Ontologies are therefore developed and used to enable data exchange between programs, enable standardization and translation between different forms of knowledge representation as well as to express the semantic of structured and semi-structured information. Structured data refers to information organized in a predefined manner, this data adheres to a strict schema, consisting of rows and columns, where each field is clearly defined, typically within relational databases for example, which then can be efficiently queried with Standard Query Language (SQL). On the other hand semi-structured data lacks a rigid schema but still contains tags or markers to separate elements and enforce hierarchies as it is the case in XML or JavaScript Object Notation (JSON). Furthermore it's crucial to understand that neither ontologies nor other kinds of knowledge representation primarily refer to the goal of presenting or visualizing knowledge networks, but rather first and foremost to a underlying model in order to formally describe this knowledge space [18].

2.4.2 Interoperability

As Blumauer et al. argues [18], interoperability refers to a condition in which business processes and IT architectures are standardized and streamlined both within and across organizational boundaries. Its primary objective is to facilitate interaction among heterogeneous data sources and applications at the technical, organizational, and semantic levels, without undermining the autonomy of individual subsystems. This is particularly relevant in contexts where participating actors employ different data formats, terminologies, or definitions, and where the nature and extent of data exchange vary depending on the specific situation. Typical examples include cross-national communication between public institutions, such as ministries, health insurance providers or large corporations that, due to historical developments, rely on proprietary data formats or organization-specific practices unfamiliar outside their own domain. Since the late 1990s, numerous national and supranational initiatives have been launched to promote interoperability across diverse sectors such as agriculture, tourism, healthcare, and transportation. The complexity of the associated challenges arises primarily from the substantial coordination required among stakeholders and interest groups. As a result, interoperability has become a significant political issue in recent years. Its political relevance is driven especially by the need for cross-sector and cross-national alignment of IT architectures, as well as efforts to establish widely accepted norms and standards that enable large-scale interoperability of existing information and communication systems and terminologies.

2.4.3 Core Concept

In ontology engineering, several essential modelling elements are distinguished, forming the structural and logical backbone of an ontology. The central components are classes, properties, axioms, individuals and annotations, each fulfilling a specific role in representing domain knowledge. Classes (often referred to as concepts or types) describe the abstract categories of entities within a domain and are typically organized in taxonomic hierarchies through subclass relations. Such class hierarchies allow inheritance of characteristics and support logical inference, as widely described by Gómez-Pérez et al. [10]. Properties (also called relations or attributes) describe how entities are connected or which characteristics they possess. Ontology languages such as OWL distinguish between object properties — which link individuals to other individuals, datatype properties — which assign literal values, and annotation properties — which provide human-readable metadata. Properties may also be constrained by domain and range declarations, and can exhibit logical characteristics such as transitivity, symmetry, functionality or inverses, all of which affect reasoning behaviour. Individuals or instances represent the concrete entities of the domain and instantiate the classes and properties defined in the ontology. They form the assertional layer of the ontology, commonly referred to as the ABox, in contrast to the terminological layer (the TBox), which contains the definitions of classes and properties, a distinction emphasized in many introductions to description logics, such as Baader et al.'s *The Description Logic Handbook* [2]. Axioms constitute the formal constraints that must hold in every model of the ontology and thereby enable automated reasoning. They specify subclass relations, equivalence conditions, disjointness, cardinality restrictions, property characteristics, and complex class expressions. Through axioms, an ontology becomes a logical theory rather than a mere vocabulary, enabling the derivation of implicit knowledge and the detection of inconsistencies, which is a core principle in formal ontology engineering [12].

2.4.4 Representation Methods And Web Standards

URI A Uniform Resource Identifier (URI) is a standardized string of characters used to uniquely identify a resource on the internet, independently of its location or retrieval mechanism. It provides a global naming scheme that allows resources to be referenced unambiguously, facilitating interoperability and integration across distributed systems. URIs serve as the foundation for addressing and linking information in web-based architectures, enabling consistent identification of documents, data objects, services and other entities [4]. URIs are a generalization of Uniform Resource Locators (URL).

Resource Description Framework The Resource Description Framework (RDF) is a formal language designed for the description of structured information. RDF enables applications to exchange while preserving its original meaning. Unlike HTML and XML, which primarily focus on the correct presentation of documents, RDF supports the integration and further processing of the contained information. For this reason, RDF is often regarded as a foundational representation format for the development of the Semantic Web. RDF originated in the 1990s, influenced by several precursor languages. The first official specification was published by the W3C in 1999, with a primary emphasis on representing metadata about web resources. In 1999, RDF was primarily conceived for expressing statements about web pages, such as authorship or licensing information. Over time, the vision of the Semantic Web expanded to encompass the general representation of semantic information, extending beyond simple RDF data to include web documents as primary subjects of description. Today, a wide array of practical tools for working with RDF is available. Almost every programming language provides libraries for reading and writing RDF documents. Numerous systems for processing large volumes of RDF data, so-called RDF stores or triple stores, are freely accessible and commercial database vendors increasingly offer corresponding extensions to their systems. In specific application domains, RDF is actively used for exchanging (meta-)data, with RSS 1.0 as a prominent example for news subscription [18].

An RDF document represents a directed graph composed of a set of nodes connected by directed edges. Both nodes and edges are assigned unique identifiers, ensuring unambiguous reference within the graph. In contrast, information encoded in XML is structured as a tree. Tree structures are well-suited for organizing information in electronic documents, as such data frequently exhibits strictly hierarchical characteristics. Moreover, information arranged in trees can typically be searched and processed efficiently. RDF, however, is built on graphs rather than trees. A central motivation for this design choice is that RDF was not intended to depict the hierarchical structure of individual documents, but instead to describe general relationships between resources. For example, it may express that a dog named “Schörli” belongs to an owner, “Max Mustermann”. The relationship between dog and owner constitutes information that is not inherently subordinate to either resource. Consequently, RDF treats such relationships as fundamental informational units. The aggregation of many such relationships naturally gives rise to a graph rather than a hierarchical tree. Another rationale lies in RDF’s intended use as a language for describing data on the web and other distributed environments. In such settings, information is often stored and managed in a decentralized manner. RDF enables seamless combination of data from multiple sources: for instance, an RDF graph describing “Schörli” can easily be merged with an RDF graph describing the owner, producing a larger graph that may reveal previously implicit relationships. If websites were to expose their data solely in isolated XML structures, such integration would be considerably more complex. The union of tree structures does not generally yield another tree and information pertaining to the same entities may be dispersed across distinct documents. RDF graphs are therefore particularly well-suited for the integration and composition of decentralized information [18].

A significant challenge arises from the fact that resources, as in XML, can be assigned different identifiers across RDF documents. On the one hand, the same resource may be labeled differently, for example, because there is no universally agreed-upon identifier for a person such as "Max Mustermann." On the other hand, identical identifiers might inadvertently be used for distinct resources — for instance, "Max Mustermann" could refer both to a person and to a company with the same name. Such ambiguities must be carefully avoided to ensure accurate and unambiguous representation of resources. To solve the latter problem, RDF generally uses URIs to designate all resources. When describing online documents like HTML pages in RDF, the corresponding URLs are often used. However, in most practical applications, the objective is not to exchange information about web pages, but rather to describe general resources. In principle, any object with a clearly defined identity within the context of a given application can serve as a resource: books, locations, people, publishers, relationships among these entities or even abstract concepts. These resources are not necessarily accessible online, and their URIs are employed solely for unambiguous identification, functioning as Uniform Resource Names (URNs) rather than URLs.

URIs provide a means to identify abstract resources that cannot be directly represented or processed by machines. They serve as references to the intended entities, such as people, books or publishers, hence their interpretation depends on the application. Specific programs may assign additional semantics to particular URIs, for example, by linking a book URI to purchasing options or current prices. In contrast, concrete data values such as numbers, dates or boolean values, typically require a consistent interpretation across contexts. In RDF, such values are represented as literals, which are reserved identifiers for resources of a specific data type. The value of a literal is generally expressed as a character string and its interpretation is determined by its datatype. For instance, the strings "42" and "042" denote the same natural number but are distinct character sequences. Literals without an explicitly assigned datatype are interpreted as strings. In RDF graphs, literals are depicted with rectangular nodes to distinguish them from URIs, which are shown in ovals. Importantly, literals cannot serve as the subject of RDF statements, and edges in the RDF graph cannot be labeled with literals, although this restriction rarely poses practical limitations.

Representation Kinds RDF expresses knowledge through the abstraction of triples, consisting of a subject, a predicate and an object. While this conceptual model remains independent of any specific syntax, various serialization formats have been developed to encode RDF data for storage, exchange and processing. Each of these representation forms follows the same semantic foundation but differs in readability, compactness and suitability for particular use cases.

One of the earliest and most widespread syntaxes is RDF/XML. It represents RDF graphs using XML as an underlying structure. RDF/XML was standardized by the World Wide Web Consortium (W3C) and serves as a bridge between traditional XML-based data interchange and semantic web technologies. Despite its strict syntactic rules and broad compatibility with existing XML tools, RDF/XML is often criticized for its verbosity and complexity. The nested nature of XML can obscure the logical relationships between RDF resources, making manual inspection and debugging difficult. An example of a simple RDF/XML representation is shown below, describing a person named Alice:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:Person rdf:about="http://example.org/alice">
    <foaf:name>Alice</foaf:name>
  </foaf:Person>
</rdf:RDF>
```

To improve human readability and authoring efficiency, the Terse RDF Triple Language (Turtle) was introduced as a more compact alternative. Turtle provides a concise textual notation that closely reflects the underlying RDF triple structure. It allows the use of prefixes to abbreviate URIs and introduces syntactic shortcuts to group related predicates and objects. These features significantly reduce redundancy and improve the clarity of RDF statements. The following Turtle example represents the same information as the RDF/XML snippet above:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex:   <http://example.org/> .

ex:alice a foaf:Person ;
foaf:name "Alice" .
```

Another serialization, known as N-Triples, offers an extremely simple and machine-friendly representation. In N-Triples, each RDF statement is written as a single line containing a subject, predicate and object, separated by whitespace and terminated by a period. The syntax avoids prefixes and complex structures, which simplifies machine parsing and facilitates large-scale processing. The same RDF data in N-Triples format can be written as:

```
<http://example.org/alice> <http://xmlns.com/foaf/0.1/name> "Alice" .
```

The last of the most widely used representation formats is provided by JSON-LD, which serializes RDF data using JSON. JSON-LD (Linked Data) is designed to facilitate the integration of semantic data into web applications and Application Programming Interfaces (APIs). By leveraging the popularity and simplicity of JSON, it enables RDF to be consumed and produced by systems that primarily operate with JSON-based data structures. The example below illustrates the representation of the same RDF graph in JSON-LD:

```
{
 "@context": {
  "foaf": "http://xmlns.com/foaf/0.1/",
  "name": "foaf:name",
  "Person": "foaf:Person"
 },
 "@id": "http://example.org/alice",
 "@type": "Person",
 "name": "Alice"
}
```

RDF Schema RDF provides the capability to make simple statements about resources. To construct ontologies based on such statements, however, it is necessary to categorize terms and to enable statements to be made about these categories or all members thereof. RDF Schema (RDFS) allows for the definition of classes of objects in RDF, the specification of subclass relationships between classes and the assignment of value and domain constraints to properties. This is achieved through the definition of classes such as rdfs:Resource and rdfs:Class, as well as predicates including rdfs:subClassOf, rdfs:domain, rdfs:range and rdf:type. Within RDF Schema, properties are also considered resources and belong to the class rdf:Property [1].

OWL The Web Ontology Language (OWL) extends RDF and RDF Schema by enabling the definition of classes and relationships between them, which RDF Schema alone does not support, such as the explicit specification or enumeration of subclasses or instances. OWL introduces additional classes and predicates to facilitate ontology construction, allowing for the creation of terminologies, precise property restrictions, and the definition of logical characteristics and equivalences of concepts. New classes can be constructed in OWL through: enumeration of instances, intersection and union of classes, cardinality constraints and complement operations. Subclasses can also be defined by restricting property values of instances, e.g., the subclass of all cars with the color red. Property characteristics such as transitivity, functionality, symmetry, inverses and inverse functionality can be specified. OWL distinguishes between object properties, whose values are class instances and datatype properties, whose values are RDF literals or XML Schema datatypes. OWL is stratified into three levels. OWL Full is fully expressive and includes classes of classes but is undecidable. OWL DL enforces a strict separation between classes (owl:Class) and instances (owl:Thing), as well as between object and datatype properties, while preserving decidable reasoning. OWL Lite further restricts class construction to intersections and property constraints, providing a simpler, implementable subset suitable for practical applications.

Examples of statements that cannot be modeled precisely in RDF(S) include:
Every project has at least one employee.
Projects are either internal or external.
The secretaries of Hermann Agostino are exactly Ulrich Gigi and Hakob Jaas.
The superior of my superior is also my superior.
[12]

OWL documents define OWL ontologies. To facilitate their use, two different syntaxes have been developed. The first is based on RDF and is primarily intended for data exchange. It is also referred to as the OWL RDF syntax, since OWL documents in this form are valid RDF documents. The second is the abstract OWL syntax, which is generally more readable for humans. An OWL ontology primarily consists of classes and properties, as in RDF(S). However, OWL allows these classes and properties to be related to each other in more complex ways, enabling the representation of constraints and relationships that cannot be captured adequately by RDF(S) alone. The header of an OWL document contains information about the namespaces used, versioning details and annotations. These elements do not directly affect the semantic content of the ontology itself [12]. As an RDF document, every OWL document has a root element in which the relevant namespaces are defined, as illustrated in the following example:

```
<rdf:RDF
  xmlns="http://www.example.org/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">

  <owl:Ontology rdf:about="">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    SWRC Ontology, version from December 2005
    </rdfs:comment>
    <owl:versionInfo>v0.5</owl:versionInfo>
    <owl:imports rdf:resource="http://www.example.org/foo"/>
    <owl:priorVersion rdf:resource="http://ontoware.org/projects/swrc"/>
  </owl:Ontology>
</rdf:RDF>
```

Example from the book ”Semantic Web”, written by Pascal Hitzler et al. [12].

SPARQL SPARQL (SPARQL Protocol and RDF Query Language) is the standard query language recommended by the W3C for retrieving and manipulating data stored in RDF format. Analogous to the role of SQL in relational databases, SPARQL provides a formal mechanism to query semantic data by matching graph patterns rather than table structures. Its design reflects the graph-based nature of RDF, allowing users to express complex queries over distributed and heterogeneous data sources. A SPARQL query typically consists of a set of triple patterns that are matched against the RDF graph. These triple patterns can include variables, which are bound to corresponding elements during query evaluation. The result is a set of variable bindings that satisfy all specified conditions. SPARQL supports a variety of query forms, including SELECT, ASK, CONSTRUCT and DESCRIBE, each serving distinct purposes:

- SELECT queries return tabular results with variable bindings.
- ASK queries return a Boolean value indicating whether a given pattern exists in the dataset.
- CONSTRUCT queries generate new RDF graphs based on the matched patterns.
- DESCRIBE queries return RDF data that provides information about specific resources.

An example of a simple SPARQL query is shown below:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person ?name
WHERE {
 ?person rdf:type foaf:Person .
 ?person foaf:name ?name .
}
```

This query retrieves all resources of type foaf:Person along with their corresponding names, using namespace prefixes to improve readability. SPARQL also provides mechanisms for filtering and combining query results. Filters can restrict variable bindings based on logical or arithmetic conditions and query results can be joined across multiple datasets using GRAPH clauses or SERVICE endpoints. This enables federated queries across distributed semantic data sources, an essential feature for the Semantic Web. In addition to querying, the SPARQL standard includes an update language, SPARQL Update, which allows the insertion, modification and deletion of RDF triples in datasets. The SPARQL Protocol defines how queries and updates are transmitted over HTTP, making SPARQL a key component of the Semantic Web infrastructure. SPARQL thus forms the central interface between ontology-based data representation and practical data access. It enables semantically rich, machine-interpretable information to be queried and integrated across diverse domains, thereby realizing the foundational vision of the Semantic Web [12].

2.4.5 Vocabularies

Within such ontologies, vocabularies are core. A vocabulary is essentially a collection of terms and their interrelations, designed to be reused in annotations and metadata. Terms in a vocabulary provide names for types of things and relations and often include definitions, labels, comments and constraints, for instance saying that property P links subject of class A to object of class B. In effect, vocabularies give structure to metadata, they allow metadata to be more than simple key-value pairs, by situating metadata terms in a structured scheme, often with hierarchical relationships, domain and range constraints, etc. To put it more bluntly, a vocabulary is a set of symbolic building blocks, which is used by an ontology, in order to describe it's domain.

A significant challenge in ontology engineering is the lack of standardized vocabularies for describing metadata about ontologies themselves. While the Semantic Web provides formal standards for modeling data (RDF, RDFS, OWL), there is no universally accepted schema to annotate ontologies with information such as creator, version, license, provenance or domain coverage. According to Peroni et al. and dAquin and Gangemi, this absence of a single standard hinders ontology discovery, interoperability and reuse across repositories and applications [7] [19].

Different communities have thus developed their own metadata vocabularies to cover overlapping needs. Some widely used ones are:

Name	Purpose	Typical Terms
DC Terms (Dublin Core)	General-purpose metadata for digital and semantic resources	`dcterms:title`, `dcterms:description`
		`dcterms:creator`, `dcterms:license`
		`dcterms:issued`, `dcterms:modified`
		`dcterms:version`
RDFS / OWL	Basic annotation and structural metadata for ontologies	`rdfs:label`, `rdfs:comment`
		`rdfs:isDefinedBy`, `rdfs:seeAlso`
		`owl:versionInfo`, `owl:imports`
		`owl:deprecated`
PROV-O	Modeling provenance, derivation, and attribution of resources	`prov:wasDerivedFrom`, `prov:wasAttributedTo`
		`prov:generatedBy`
PAV	Lightweight provenance, authoring, and versioning metadata	`pav:createdBy`, `pav:createdOn`
		`pav:version`, `pav:derivedFrom`
SKOS	Modeling and documenting controlled vocabularies	`skos:prefLabel`, `skos:altLabel`
		`skos:definition`, `skos:scopeNote`
Ontology-specific vocabularies (OMV, DOOR, VOAF)	Ontology-level description, usage, and metric metadata	Ontology domain, intended use
		Version history, ontology metrics

Table 2.1: Purpose And Typical Terms Of Commonly Used Metadata Vocabularies

Name	Strengths	Limitations
DC Terms (Dublin Core)	Widely adopted; simple; interoperable across domains	Generic scope; limited domain specificity; overlapping properties
RDFS / OWL	Universally supported; core Semantic Web standards	Mostly informal annotations; limited provenance; versioning support
PROV-O	Expressive and formally defined provenance model	High modeling effort; comparatively complex; often excessive for small ontologies
PAV	Lightweight and easy to apply; complements DC and PROV-O	Less widely adopted; partial overlap with existing standards
SKOS	Well-suited for terminology management and labeling	Limited expressiveness; not intended for complex OWL axioms
Ontology-specific vocabularies (OMV, DOOR, VOAF)	Rich ontology description and analysis capabilities	Heterogeneous modeling; weak tool support; limited adoption

Table 2.2: Strengths And Limitations Of Commonly Used Metadata Vocabularies

The two tables above were created with ChatGPT in order to get an overview of commonly used metadata vocabularies as well as their corresponding purpose, strengths and limitations [4].

Because these vocabularies overlap, there is still no harmonized framework ensuring semantic consistency across ontology repositories. Consequently, ontology metadata often remain heterogeneous and incomplete, limiting automation in ontology discovery and alignment [16].

So why is sharing and reusing ontologies a central task in the semantic web at all? And why does annotating metadata to ontologies facilitate this task?

The goal of the Semantic Web is to enable data to be understood and linked not only by humans, but also by machines. To achieve this goal, ontologies must create standardized meanings: when different systems use the same ontologies, they understand terms such as person, organization, or location in the same way. Shared ontologies enable data sources and applications to communicate with each other without having to manually align their data structures or meanings. Existing ontologies can be reused and expanded instead of developing new semantic models for each project. This saves effort and increases consistency. Annotating metadata for ontologies means describing additional information about the ontology itself, such as author, version, license, scope, language, related ontologies, intended use or quality characteristics. This metadata facilitates sharing and reuse because it improves discoverability: metadata enables ontologies to be searched for and identified in reposito-

[4] https://chatgpt.com/

ries like BioPortal [5]. This allows developers and researchers to assess whether an ontology meets their requirements. In addition, information about origin and version increases transparency and traceability. And, of course, references to related ontologies or import relationships help with integration into existing systems.

2.4.6 Documentation Tools

Ontologies are typically created using dedicated editors and subsequently exported in formats such as Turtle or RDF/XML, which are difficult for humans to read and navigate. Researchers often address this limitation by referring to papers or technical reports that describe the ontology. However, such publications generally focus on presenting scientific contributions rather than providing detailed definitions of each ontological concept. A more suitable approach is to produce comprehensive documentation for all terms contained in the ontology. Because this is a time-consuming process, the Semantic Web community has developed a variety of tools to support ontology documentation. These tools take an ontology file as input and automatically generate HTML documentation based on the metadata embedded within the ontology, producing structured sections for all classes, properties and named individuals. Many tools also offer visualizations of the ontology, including classes and their relationships, which is particularly useful for gaining an overview of larger or more complex ontologies. Widely used tools for ontology documentation include pyLODE, WIDOCO, Parrot and OwlDoc, the latter being integrated directly into the Protégé ontology editor.

Tools parse the ontology file, extract the triples where the subject is the ontology (or class or property) and predicate is an annotation property. Also, many ontologies specify metadata at the ontology level, e.g., in OWL files, an ontology element with annotations. Tools find that and read it's annotations. Some properties are built into OWL or RDFS (e.g., *owl:imports, owl:versionInfo, owl:priorVersion, owl:deprecated*) and are standardized. Tools that are ontology-aware know to look for these. If the ontology uses external vocabularies like Dublin Core, tools may import or interpret those annotation predicates. Though not always labeled "metadata", things like subclass hierarchies, property domains, ranges, cardinalities, restrictions and disjointness are part of the structural content. Documentation tools often combine structural metadata plus narrative/descriptive metadata for a complete view.

Nevertheless there are limitations, not every tool is able to read in every vocabulary, however, most of the documentation tools can recognize standardized RDFS/OWL annotations without any problems. Extern or user-specified vocabularies are only manageable to read if the tool is designed for that purpose or extendable with some kind of plugin for instance. Widoco even suggests missing metadata when an ontology is imported and is used for documenting the resulting vocabulary of this work. Some structural features may not be visually or textually represented (e.g. complex class expressions, certain kinds of

[5] https://bioportal.bioontology.org/

restrictions) by some visualization tools [6] [7] [8] [9] [13] [17].

[6] https://dgarijo.github.io/Widoco/
[7] https://protege.stanford.edu/
[8] https://pypi.org/project/pylode/
[9] https://github.com/dayures/parrot

Chapter 3

Design Of A Metadata Vocabulary For Ontologies

In the following chapter the actual vocabulary will be described in detail after introducing a first design draft. But beforehand, an analysis is carried out to determine which vocabularies are already in frequent use.

3.1 Analysis Of Existing Metadata Vocabularies

To get an overview, firstly the work from Biswanath Dutta et al., wherein the creation of MOD 1.2 is documented as well as an analysis of existing metadata vocabularies to describe ontologies, was evaluated [8]. Early and still broadly used standards include the Dublin Core (DC) and DCMI Metadata Terms (DCT), which originated in the late 1990s from the DCMI initiative for describing electronic objects. These were followed in the early 2000s by the major W3C Recommendations RDFS, OWL and SKOS, which rapidly became foundational and widely used across the Semantic Web. In 2005, the Ontology Metadata Vocabulary (OMV) was created within several EU research projects. Although it introduced a reasonably rich schema (16 classes, 33 object properties, 29 data properties), OMV saw only limited adoption and was abandoned around 2007. Its most significant limitation was the lack of alignment with prevailing standard vocabularies. This weakness inspired the development of MOD 1.0, which reused elements from SKOS, FOAF, DC as well as DCT, and introduced additional classes and properties. Despite this improvement, MOD 1.0 still lacked several relevant metadata elements. Also in 2005, the lightweight but practical VANN vocabulary for annotating vocabulary descriptions emerged and has since seen frequent usage. In contrast, the Descriptive Ontology of Ontology Relations (DOOR), published in 2009 as a highly formal vocabulary, defining 32 relations between ontologies, never gained traction beyond the NeON project. Later, the VOAF vocabulary was introduced to describe vocabularies used in the Linked Data cloud. Although VOAF captures relationships between vocabularies, it does not reference OWL or DOOR, despite overlapping functionality. Ontologies share characteristics with datasets and data catalog assets, and for this reason several dataset-oriented vocabularies have become relevant. VoID, created in the late 2000s, is widely used in the Linked Data ecosystem to describe datasets. Even more influential is DCAT, a W3C Recommendation designed for dataset and catalog metadata, along with its profile Asset Description Metadata Schema (ADMS) for semantic assets such as code lists and taxonomies. Schema.org has also contributed a broadly adopted dataset class. Additional

vocabularies relevant to ontology metadata include FOAF and DOAP (for people and projects), Creative Commons (for rights information), SPARQL Service Description (for endpoints) and provenance vocabularies such as PROV and PAV. OboInOwl offers mappings for OBO ontology headers and is commonly used within the OBO community, although it is not a standard.

Recent developments have significantly modernized metadata practices for ontologies, particularly through the evolution of MOD. MOD 2.0 represents a substantial redesign and is built explicitly as an extension of DCAT2, the 2020 revision of the W3C Data Catalog Vocabulary. By adopting DCAT2 as its foundation, MOD 2.0 integrates ontology metadata into broader dataset catalog practices and becomes fully aligned with FAIR principles. MOD 2.0 introduces a clear model centered on classes such as *mod:SemanticArtefact* and *mod:SemanticArtefactDistribution*, incorporates provenance information, and systematically reuses established vocabularies including DCAT2, DCT, FOAF, Schema.org and ADMS. Building on this, newer releases — referred to collectively as MOD 3.x — further consolidate MOD as a DCAT2-aligned profile. The latest specifications, including MOD 3.2.x, expand the set of classes beyond those in MOD 2.0 and refine the modeling of semantic artefacts, distributions and related services [8].

As the researchers at MOD 1.2 recognized back then, there is a strong overlap in all the vocabularies studied. It shows that no currently existing vocabularies really covers enough aspects of ontologies to be used solely. Despite a few exceptions, metadata vocabularies do not rely on one another and redefine things that have already been described several times before. This makes it all the more important to implement this in the resulting vocabulary, especially enhancing those things MOD 2.0 / 3.0 misses.

3.2 Use Of Those Current Existing Vocabularies

Here is what Biswanath Dutta et al. found out in their study while analysing 222 ontologies distributed over several sources (108 from NCBO BioPortal, 53 from AgroPortal, 61 from a simple google search):

- Without descriptions/annotations: 23
- With metadata: 199 - number of properties from 1 to 20
- 53 ontologies retrieved from AgroPortal
  - 2 ontologies with only 1 metadata property
  - 8 ontologies with 10–20 properties
  - 21 ontologies with 2–9 properties
- 32 vocabularies used in total
  - 12 most frequently used, half of which are W3C or Dublin Core standards
  - 20 less common vocabularies, mostly used once or a few times, including PROV, SCHEMA, VOID, ADMS, DOAP, PRISM, EFO, IRON, CITO, etc.

Prefix	Number	Properties used (number)
dc	294	creator (60), title (51), contributor (34), description (32), rights (20), date (19), subject (15), publisher (14), format (10), identifier (10), license (10), language (9), source (5), coverage (2), issued (1), modified (1), type (1)
rdfs	196	comment (110), seeAlso (23), label (58), isdefinedby (5)
owl	194	versionInfo (105), imports (70), versionIRI (16), priorVersion (3)
oboInOwl	181	hasOboFormatVersion (38), date (35), default-namespace (35), savedBy (31), auto-generated-by (27), namespaceIdRule (3), synonymtypedef (3), hassubset (2), typeref (2), data-version (1), id_space (1), subsetdef (1), treat-xrefs-as-genus-differentia (1), treat-xrefs-as-is_a (1)
dct	105	license (15), modified (15), creator (12), description (12), created (9), issued (8), title (8), subject (6), rights (4), contributor (3), identifier (3), publisher (3), alternative (1), available (1), hasPart (1), hasVersion (1), language (1), lastModified (1), type (1)
skos	27	definition (8), altLabel (6), prefLabel (6), editorialNote (4), historyNote (2), changeNote (1)
vann	21	preferredNamespacePrefix (11), preferredNamespaceUri (10)
cc	12	license (12)
protege	11	defaultLanguage (11)
dcat	9	landingpage (5), downloadURL (2), contactPoint (1), mediaType (1)
foaf	5	primaryTopic (2), homepage (1), maker (1), page (1)
pav	5	version (5)
void	5	subset (2), dataBrowse (1), dataDump (1), sparqlEndpoint (1)

Figure 3.1: Frequently Used Metadata Vocabularies

Most of the 32 vocabularies examined are general-purpose, and ontology-specific metadata vocabularies (e.g., OMV, DOOR) are entirely absent from the selected sample. Two widely used vocabularies, oboInOwl and Protégé annotations, appear because they are automatically included by ontology development tools. The most frequently used metadata elements are *rdfs:comment*,

owl:versionOf, and *owl:imports*, likely due to their easy availability in editors such as Protégé, where selected RDFS and OWL properties are readily accessible via the annotation tab.

The study also observes several issues: multiple properties often capture the same information, e.g., some ontologies use *dc:license*, others *cc:license*. Also there is confusion between DC and DCT, since DCT refines the 15 core DC properties and generic properties such as *rdfs:comment* or *dc:date* are frequently used instead of more specific alternatives, like *dc:description* or *dc:created/dc:modified*.

3.3 Initial Design Considerations

First of all, consideration was given to how the vocabulary is fundamentally structured. What was clear from the outset was that there should be a superclass *Resource* that contains all possible things, i.e., in principle, everything is a resource — ontologies, but also persons or versions of an ontology. The actual resulting classes are described in Chapter 3.5. It is important to note that the vocabulary aims to capture all possible metadata that someone might want to use. This was achieved by taking a large number of metadata properties from existing, widely used vocabularies and supplementing them with new properties created for metadata that is new and does not yet exist anywhere else. The first draft was a simple sketch created in draw.io [1] 3.2, containing the classic metadata that usually comes to mind first, as well as a rough overview of the abstract components and their relationships, which, incidentally, does not follow any strict technology such as UML, but is simply intended to provide an abstract, initial overview of the components of the system, not tied to any specific modeling technology.

Each component, such as a class or property, must be described in detail by using properties such as rdfs:label, rdfs:isDefinedBy, dcterms:issued, or dcterms:descriptions, in order to specify all meta information for each component, which contributes to better understanding and reusability of the whole system. In addition, the URI of an existing property that was used should always be specified. Another fundamental question was whether existing properties that are to be reused should simply be adopted directly, or whether they get defined as an own property that then refer to the existing ones. For example, if the property *http://purl.org/dc/terms/title* [2] should be directly imported into the vocabulary, or an own property should be created which then refers to it with the use of *rdfs:subPropertyOf* [3]. However, this would be unnecessary for relatively simple properties, as we want to use exactly this property and do not want to achieve a more narrow specialization. The property already fits semantically, so if the range shall not be restricted, for example, this is the better solution in terms of reusability and maintenance of the vocabulary. A reference would express that the title property is more specific than <*http://purl.org/dc/terms/title*> and inherits its meaning, but as mentioned above, this would only make sense if the standard property is too general and additional restrictions are needed or a different name is desired. However, common practice is to use suitable properties directly, as is the case in MOD2.0 [8]. In general, existing properties were reused whenever they precisely matched the intended use case. In cases where no suitable property could be identified, new properties were defined without reference to existing vocabularies. Furthermore, documentation tools and metadata portals then recognize the properties immediately.

But why not simply creating own properties everytime? It is, in principle, possible to define a custom data-type property, and this would function technically. However, there are strong reasons to reuse established properties from well-known vocabularies: tools such as reasoners, SPARQL queries, or other ontologies can automatically recognize the corresponding data because they are familiar with these properties. Vocabularies such as VANN are also well es-

[1] https://www.drawio.com/
[2] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/title/
[3] https://www.w3.org/TR/rdf-schema/#ch_subpropertyof

tablished, using them corresponds to speaking the "common language" of the community. Ontologies or datasets that rely on VANN can directly read and interpret the relevant information without requiring additional explanation.

As the file format for the vocabulary Turtle was choosen, since it's the most used representation form, as well as the simpliest, foremost in contrast to RDF/XML. Turtle allows direct representation of OWL ontologies while remaining fully compatible with RDF. All OWL constructs, i.e., classes, properties, individuals, hierarchies and restrictions, are ultimately expressed as RDF triples, making Turtle a flexible format for ontology serialization. Due to its size, the complete file content is not included in this thesis, nevertheless short sections are briefly presented, the entire ontology is available on GitHub [4].

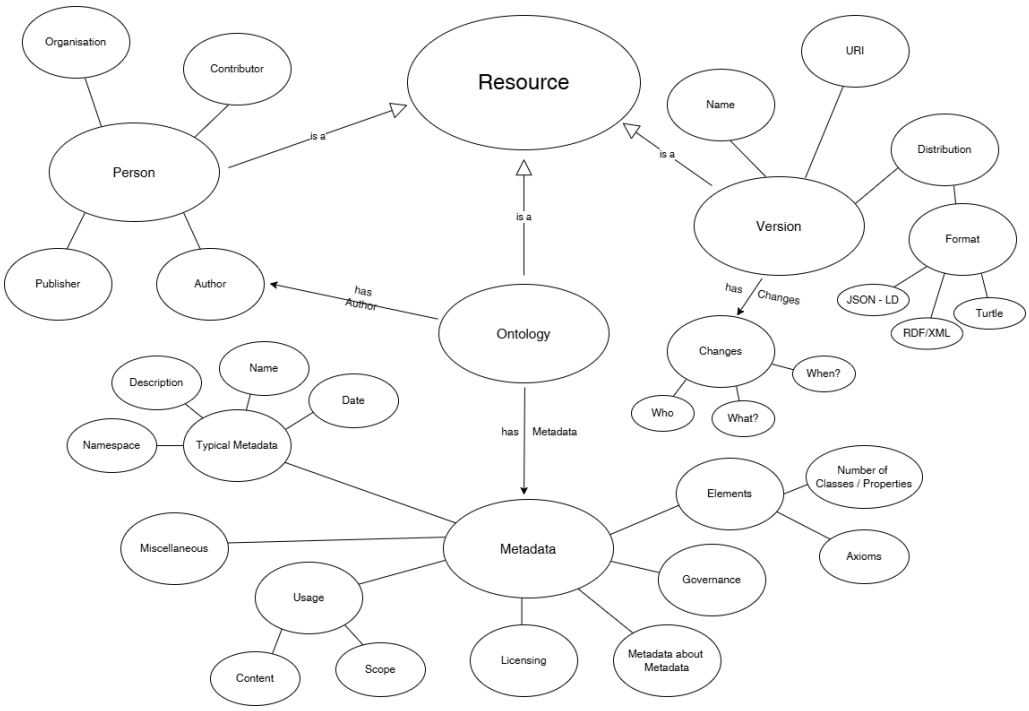

Figure 3.2: First Draft Of Vocabulary Structure

[4]https://github.com/moritzhaider/aowm

3.4 Discovering Concrete Properties

In the previous sections a first overview was developed, to get a feeling which vocabularies exist and which are frequently used. To select according properties, foremost the checklist for complete vocabulary metadata was a very good resource, in order to know some property types which definitely should occur [5]. Also, the *Metadata for Ontology Description and Publication Ontology* was a great inspiration to get an overview, which properties are in use in an actual metadata vocabulary [6]. Many other vocabularies like the FOAF, VANN or DCMI were investigated as well [7] [8] [9]. In addition, consideration was given to what kind of metadata information is not yet available in existing vocabularies, but where it makes sense to annotate this as metadata for an ontology. These were created as separate/new properties, using the class structure that was defined.

3.5 Structure Of The Vocabulary

Topmost at the turtle file the prefix annotations are located, in order to simplify all the utilized URIs.

```
@prefix aowm:    <http://dqm.faw.jku.at/ontologies/aowm#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix dct:     <http://purl.org/dc/terms/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix dcmi:    <http://purl.org/dc/dcmitype/> .
@prefix vann:    <https://vocab.org/vann/> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
...
```

[5] https://dgarijo.github.io/Widoco/doc/bestPractices/index-en.html
[6] https://github.com/FAIR-IMPACT/MOD
[7] https://iptc.org/thirdparty/foaf/
[8] https://vocab.org/vann/
[9] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Below that, the header follows, where the ontology is shortly described by using it's own vocabulary. In general, sections are always marked through a corresponding comment block. *aowm* acts as a prefix for the vocabulary and the namespace <*http://dqm.faw.jku.at/ontologies/*> comes from the institute that initiated this work [10].

```
#################################################################
# Ontology Header
#################################################################
<http://dqm.faw.jku.at/ontologies/aowm>
a owl:Ontology ;
  dct:title "Vocabulary For Annotating Ontologies With Metadata"@en ;
  dct:creator "Moritz Haider" ;
 dct:description "A vocabulary designed to describe metadata about ontologies."@en ;
  dct:created "2026-01-31"^^xsd:date ;
  dct:license <http://creativecommons.org/licenses/by/4.0/> ;
  vann:preferredNamespacePrefix "aowm" ;
  vann:preferredNamespaceUri <http://dqm.faw.jku.at/ontologies/aowm#> ;
  owl:versionInfo "0.1" .
```

Afterwards section for the classes and properties can be found, whereby *Resource* works as an parent class for all other classes in the vocabulary, it is a subclass of *owl:Thing*.

```
aowm:Resource
 a owl:Class ;
 rdfs:subClassOf owl:Thing .
 rdfs:label "Resource"@en ;
 rdfs:comment "Generic parent element for all domain classes in the AOWM vocabulary.
    Used as the common superclass for ontology metadata resources (Distribution, Version, ...)."@en ;
 rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
 dct:issued "2026-01-31"^^xsd:date .
```

In general, classes are always structured as follows:

```
aowm:Resource a owl:Class .
```

Here, *a* is shorthand for *rdf:type*, declaring that *aowm:Resource* is an OWL class.

Classes can have hierarchical relationships, such as subclassing (*rdfs:subClassOf*), enabling the definition of taxonomies and ontological hierarchies.

```
rdfs:subClassOf owl:Thing .
```

[10] https://www.jku.at/en/institute-for-application-oriented-knowledge-processing/

rdfs:label assigns a human-readable name to a resource while *rdfs:comment* supplements a description or explanatory text for a resource.

```
rdfs:label "Resource"@en ;
rdfs:comment "Generic parent element for all domain classes in the AOWM vocabulary.
Used as the common superclass for ontology metadata resources (Distribution, Version, ...)."@en ;
```

To provide a reference to the source or ontology in which a resource is defined, *rdfs:isDefinedBy* is used, while *dct:issued* is utilized to indicate the creation or publication date of a document, data set or resource.

```
rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
dct:issued "2026-01-31"^^xsd:date .
```

Finally, *rdfs:seeAlso* serves as a reference to additional information or related resources [11] [12].

```
rdfs:seeAlso owl:Ontology .
```

In terms of logical structure, the central class for representing an ontology is described within the vocabulary as a class *Ontology*, accordingly a subclass of *Resource*. Moreover there is a class *Catalog*, principle a data catalog in which the ontology can appear, according to *mod:SemanticArtefactCatalog*, a dedicated web-based system that fosters the availability, discoverability and long-term preservation and maintenance of semantic artefacts, including ontologies [13]. Such a catalog can have a record, represented by the class *CatalogRecord*. Furthermore there is a class *Distribution*, which is a specific representation and/or serialization of an ontology. An ontology can have several distributions, for instance an .owl file, .ttl file or a HTML documentation. Including these distributions is crucial for fullfilling the FAIR critera. Moreover the class *Agent* is utilitzed as a resource that acts in some way or role (class *Role*), also responsible for the two subclasses *Person* and *Organization*. Finally, the classes *SerializationFormat, DegreeOfMaturity* and *MetadataCreationMethod* were created to generate enumeration types that can then be used for certain properties in the range value area. For instance, the property *aowm:degreeOfMaturity* looks as such:

```
aowm:degreeOfMaturity
 a owl:DatatypeProperty
 rdfs:label "degree of Maturity"@en ;
 rdfs:domain aowm:Ontology ;
 rdfs:range  aowm:DegreeOfMaturity ;
 dct:description: "Degree of maturity of an ontology"@en ;
 dct:issued "2026-01-30"^^xsd:date ;
 rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .
```

[11] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[12] https://www.w3.org/TR/rdf-schema/
[13] https://github.com/FAIR-IMPACT/MOD

SKOS enables the definition of concepts that each represent a specific meaning within a semantic framework and are particularly suitable for classifications and enumerations. For this property, the different degrees of maturity were modeled as SKOS Concepts. Each concept represents a clearly defined maturity level, e.g., from an initial draft state to a fully validated and productive ontology state. The concepts were grouped within their own SKOS Concept Scheme, enabling systematic organization and consistent referencing within the metadata [14].

The corresponding class:

```
aowm:DegreeOfMaturity
 a owl:Class ;
 rdfs:subClassOf aowm:Resource ;
 rdfs:label "Degree Of Maturity"@en ;
rdfs:subClassOf aowm:Resource ; rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
 dct:issued "2026-01-31"^^xsd:date .
```

The concept scheme:

```
aowm:DegreeOfMaturityScheme
 a skos:ConceptScheme ;
 rdfs:label "Degree of Maturity Scheme"@en ;
 skos:hasTopConcept aowm:Draft, aowm:Stable, aowm:Mature ;
 dct:isPartOf <http://dqm.faw.jku.at/ontologies/aowm> ;
 dct:issued "2026-01-31"^^xsd:date .
```

Containing all concepts, which display the different maturity states, referencing to the scheme as well as the corresponding class.

```
aowm:Draft
 a aowm:DegreeOfMaturity, skos:Concept ;
 rdfs:label "draft"@en ;
 skos:inScheme aowm:DegreeOfMaturityScheme ;
skos:definition "Ontology is in early development stage; incomplete and experimental."@en .

aowm:Stable
 a aowm:DegreeOfMaturity, skos:Concept ;
 rdfs:label "stable"@en ;
 skos:inScheme aowm:DegreeOfMaturityScheme ;
skos:definition "Ontology is stable, can be used for production, minor changes expected."@en .

aowm:Mature
 a aowm:DegreeOfMaturity, skos:Concept ;
 rdfs:label "mature"@en ;
 skos:inScheme aowm:DegreeOfMaturityScheme ;
skos:definition "Ontology is fully mature, stable, widely adopted, unlikely to change."@en .
```

[14] https://www.w3.org/TR/skos-reference/

The same thing was done for different serialization format kinds:

```
aowm:SerializationFormatScheme
 a skos:ConceptScheme ;
skos:hasTopConcept aowm:Turtle, aowm:RdfXml, aowm:JsonLd, aowm:NTriples,  aowm:Trig ;
 rdfs:label "Serialization format scheme"@en ;
 dct:isPartOf <http://dqm.faw.jku.at/ontologies/aowm> ;
 dct:issued "2026-01-31"^^xsd:date .

aowm:Turtle
 a aowm:SerializationFormat, skos:Concept ;
 rdfs:label "Turtle"@en ;
 skos:inScheme aowm:SerializationFormatScheme ;
 skos:definition "Serialization with Turtle."@en .

aowm:RdfXml
 a aowm:SerializationFormat, skos:Concept ;
 rdfs:label "RDF/XML"@en ;
 skos:inScheme aowm:SerializationFormatScheme ;
 skos:definition "Serialization with RDF/XML"@en .

aowm:JsonLd
 a aowm:SerializationFormat, skos:Concept ;
 rdfs:label "JSON-LD"@en ;
 skos:inScheme aowm:SerializationFormatScheme ;
 skos:definition "Serialization with JSON-LD"@en .

aowm:NTriples
 a aowm:SerializationFormat, skos:Concept ;
 rdfs:label " N-Triples"@en ;
 skos:inScheme aowm:SerializationFormatScheme ;
 skos:definition "Serialization with N-Triples"@en .

aowm:Trig
 a aowm:SerializationFormat, skos:Concept ;
 rdfs:label "TriG"@en ;
 skos:inScheme aowm:SerializationFormatScheme ;
 skos:definition "Serialization with TriG"@en .
```

However, such enumerations were not created for every property. Often, only a simple string is specified as a property range, as this allows for much greater “freedom” in specification and not just a strict selection of enumeration types, where no additional information can be specified (unless additional properties are listed). This decision was made based on own assessment of where potential users of the vocabulary would be most likely to work with an abstract classification of the range or where the annotation of arbitrary information in the form of a string would be more appropriate. All declared classes can be found in the Appendix 4.3.

These classes have several object properties, for example an Agent can have some kind of a Role, which is implemented as a class *Role* and an object property *hasRole*, within the corresponding role is specified.

```
aowm:hasRole
 a owl:ObjectProperty ;
 rdfs:label "has role"@en ;
 rdfs:domain aowm:Ontology, aowm:Catalog ;
 rdfs:range aowm:Role ;
 dct:description "Role belonging to a ontology/catalog."@en ;
 dct:issued "2026-01-30"^^xsd:date ;
 rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .
```

There is also an object property *hasAgent*, which assigns an agent to an ontology.

```
aowm:hasAgent
 a owl:ObjectProperty ;
 rdfs:label "has agent"@en ;
 rdfs:domain aowm:Ontology ;
 rdfs:range aowm:Agent ;
 dct:description "Agent belonging to a ontology/catalog."@en ;
 dct:issued "2026-01-30"^^xsd:date ;
 rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .
```

Here is a brief example regarding the usage of the mentioned properties:

```
@prefix aowm: <http://dqm.faw.jku.at/ontologies/aowm#> .
@prefix ex: <http://example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:MyOntology
 a aowm:Ontology ;
 rdfs:label "Example ontology"@en ;
 aowm:hasAgent ex:DrAlice ;

ex:DrAlice
 a aowm:Agent, aowm:Person ;
 rdfs:label "Dr. Alice Muster"@en .
 aowm:hasRole ex:CuratorRole .

ex:CuratorRole
 a aowm:Role ;
 rdfs:label "Curator"@en ;
 dct:description "Responsible for curating and quality control of the ontology."@en .
```

In addition, for own defined properties the domain is always specified with *rdfs:domain* [15], for existing properties only *dcterm:domainIncludes* [16] is used, describing the possible or rather commonly used areas of application, without a mandatory reference and therefore not overwriting the original domain. Sometimes *dcterm:rangeIncludes* is specified as well. Moreover, *owl:equivalentProperty* [17] is used in order to express properties which are semantically equal, thus they are not suggestions, but rather that they can be used instead of the original property whichin *owl:equivalentProperty* is placed. The goal of using these attributes is increasing the interoperability between metadata communities. But foremost for *owl:equivalentProperty* it is important that it is intentional that all properties should have the same semantic. For instance it would not be supporting if one of the mentioned properties has another meaning, or rather semantics. A distinction must be made between *rdf:type rdf:Property* and *a owl:DatatypeProperty*. The former expresses, in very general terms, that a resource is an RDF property without providing additional semantics. In contrast, *owl:DatatypeProperty* denotes an OWL datatype property, i.e., a property whose values are literals such as *xsd:string* or *xsd:int*. Since *owl:DatatypeProperty* is defined as a subclass of *rdf:Property*, every *owl:DatatypeProperty* is implicitly also an *rdf:Property*.

A direct usage of an existing property in the vocabulary looks as follows:

```
dct:title
 rdf:type rdf:Property ;
 rdfs:label "title"@en ;
 dct:domainIncludes
  aowm:Ontology ,
  aowm:Distribution,
  aowm:Catalog ;
 dct:description  "dct: A name given to the resource."@en ;
 rdfs:isDefinedBy <http://purl.org/dc/terms/> ;
 dct:issued "2008-01-14"^^xsd:date ;
 owl:equivalentProperty
  rdfs:label ,
  cc:attributionName ,
  omv:name ,
  schema:name ,
  skos:prefLabel ,
  foaf:name .
```

On the other hand, the definition for creating a new property follows:

```
aowm:supportPeriod
 a owl:DatatypeProperty
 rdfs:label "support period"@en ;
 rdfs:domain aowm:Ontology ;
 rdfs:range xsd:string ;
 dct:description "Planned period of support for an ontology"@en ;
 dct:issued "2026-01-30"^^xsd:date ;
 rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .
```

[15] https://www.w3.org/TR/rdf12-schema/#ch_domain
[16] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/dcam/domainIncludes/
[17] https://www.w3.org/TR/owl2-syntax/#Equivalent_Object_Properties

Ultimately, this leads to a mixture of using existing properties and creating new ones, whereby the style of property definition has generally been adopted from the current version of MOD [18], as have some properties, and the class structure is also based on it in part. As already mentioned, in the context of ontology metadata vocabularies, it is common practice to reuse existing, well-established properties rather than introducing new ones, in order to maximise interoperability and semantic consistency. For instance, the MOD ontology adopts this principle by directly reusing foaf:homepage instead of defining a new, semantically equivalent property [19]. Although the original FOAF specification assigns only minimal constraints to this property, MOD provides additional, locally valid modelling guidance by specifying *rdfs:domain* and *rdfs:range* statements as well as *dcterms:domainIncludes*. These constraints do not modify the global semantics of FOAF, rather, they constitute application-profile–level restrictions that reflect the intended use of the property within MOD. In particular, defining *rdfs:domain owl:Thing* avoids narrowing the original FOAF scope, while *dcterms:domainIncludes* indicates the expected classes (e.g., ontologies, catalogues) to which the property is typically applied.

[18] https://github.com/FAIR-IMPACT/MOD/tree/main
[19] https://iptc.org/thirdparty/foaf/

This combination enables MOD to tailor the property to its metadata requirements without fragmenting the vocabulary landscape or compromising compatibility with FOAF. This property was also adopted using this already excellent approach, with a few modifications such as changing the domainIncludes values.

```
### http://xmlns.com/foaf/0.1/homepage
foaf:homepage
 rdf:type rdf:Property , owl:InverseFunctionalProperty , owl:ObjectProperty  ;
 rdfs:subPropertyOf
  foaf:page ,
  foaf:isPrimaryTopicOf ;
 rdfs:label "homepage"@en , "page web"@fr ;
 rdfs:domain owl:Thing ;
 dcterms:domainIncludes
  mod:SemanticArtefact ,
  mod:SemanticArtefactCatalogRecord  ,
  mod:SemanticArtefactCatalog ;
 rdfs:range foaf:Document ;
 dcterms:description "FOAF: A homepage for some thing."@en ;
 rdfs:isDefinedBy <http://xmlns.com/foaf/0.1/> ;
 dcterms:issued "2014-01-14"^^xsd:date ;
 owl:equivalentProperty
  schema:mainEntityOfPage ,
  cc:attributionURL ,
  doap:blog ,
  <http://www.isibang.ac.in/ns/mod/1.0/homepage> ;
 pav:derivedFrom <https://w3id.org/mod/1.0> ;
 pav:importedOn  "2015-08-05"^^xsd:date ;
 #OPTIONAL STATEMENTS
  prov:wasInfluencedBy "MIRO guidelines: C.3" ,
  "FAIR principle: F2" .
```

This property was adopted as follows:

```
### http://xmlns.com/foaf/0.1/homepage
foaf:homepage
 rdf:type
  rdf:Property ,
  owl:InverseFunctionalProperty ,
  owl:ObjectProperty  ;
 rdfs:subPropertyOf
  foaf:page ,
  foaf:isPrimaryTopicOf ;
 rdfs:label "homepage"@en ;
 rdfs:domain owl:Thing ;
 dct:domainIncludes
  aowm:Ontology ,
  aowm:Catalog ,
  aowm:CatalogRecord ;
 rdfs:range foaf:Document ;
 dct:description "FOAF: A homepage for some thing."@en ;
 rdfs:isDefinedBy <http://xmlns.com/foaf/0.1/> ;
 dct:issued "2014-01-14"^^xsd:date ;
 owl:equivalentProperty
  schema:mainEntityOfPage ,
  cc:attributionURL ,
  doap:blog .
```

Also worth mentioning are the newly introduced properties in the “metadata about metadata” section, a brief excerpt from this can be found in the appendix 4.3.

3.6 The Resulting Vocabulary

The AOWM vocabulary is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), which allows reuse and extension under the condition of proper attribution [20]. This license choice aligns with common practices for publishing RDF vocabularies and metadata standards. Overall, the vocabulary is divided into 21 categories, comprising a total of 194 properties using 33 vocabularies, whereby the categories are primarily intended to improve structure and clarity, also with a view to a possible expansion of the product.

The categories are:

- general properties
- description properties
- license properties
- metrics
- usage and application context properties
- media properties
- person and organization properties
- community properties
- links
- date properties
- elements and relations
- content
- distribution
- versioning and evolution
- quality and maturity
- technical representation
- maintenance and lifecycle
- object descriptions
- methodology and provenance
- metadata about metadata
- miscellaneous

The turtle file was validated with IDLab Turtle Validator [21]. Due to it's size, the resulting vocabulary is located in a github repository under the file name *aowm.ttl* [22].

[20] https://creativecommons.org/licenses/by/4.0/deed.en
[21] http://ttl.summerofcode.be/
[22] https://github.com/moritzhaider/aowm

Chapter 4

Trends - Conclusion And Future Work

4.1 Trends

In recent years, there has been a noticeable shift in both academic and industrial settings: Knowledge Graphs (KGs) have gained prominence, often hailed as a progression beyond traditional ontologies. But the reality is more nuanced. Ontologies are far from obsolete and Knowledge Graphs are not full replacements, they are better thought of as evolutionary extensions or complements. What follows is a reflection on the differences between the two, whether KGs can be considered successors of ontologies, how relevant ontologies still are and what roles KGs currently play.

Ontologies typically provide formal descriptions of domain concepts. They define what is allowed, providing a schema or model of knowledge. Knowledge Graphs, on the other hand, include not only such schemas but also large numbers of instances - individual entities, relationships among them, potentially from multiple heterogeneous sources, with dynamic updates and often pragmatic concerns, such as query performance. Many ontologies are used without many (or any) instances, rather they are used to classify, annotate and provide metadata. As one recent paper notes [21]:

“Ontologies are often published with only the model-level layer, serving as knowledge models for a given domain, without any data.” When large numbers of individuals are represented together with the ontology as schema, we move into the realm of a Knowledge Graph.

Knowledge Graphs are generally more dynamic: new data - new entities - evolving relations and heterogeneous sources, also more tolerant of inconsistencies. Ontologies as schema tend to be more static, because changes in structure often have large ripple effects in reasoning and usage. Ontologies often carry logical semantics, enabling reasoning, consistency checking, inference and so on. Knowledge Graphs may still use these, especially in the schema part, but in many applications, the reasoning is lighter or supplemented by statistical or machine learning approaches. The focus moves from absolute correctness of the model to usefulness, connectivity, data integration and queryability [15].

4.1.1 Are Knowledge Graphs "Successors" of Ontologies?

It is inaccurate to view Knowledge Graphs as complete successors in the sense that ontologies are no longer needed. Ontologies remain essential for defining schema, vocabularies, conceptual clarity, semantic interoperability. Every mature Knowledge Graph depends in many cases on an ontology or set of ontologies for its schema part. They allow leveraging of ontologies in more data-intensive, interconnected, practical settings [15].

4.1.2 Are Ontologies Outdated?

Ontologies are still active and important, recent literature shows that ontologies remain widely used in many domains. The work from Bernabé et al. shows foundational ontologies are used for ontology construction, repair, mapping, ontology-based data analysis and that they are claimed to improve interoperability and reasoning, though the authors also note empirical evaluation is rare [3]

A bibliometric topic-modeling study, "Recent Trends and Insights in Semantic Web and Ontology-Driven Knowledge Representation Across Disciplines Using Topic Modeling" (2025), analysed over 10,000 articles from 2019-2024 and found that ontology engineering remains a core theme alongside knowledge graphs and linked data [24].

However, there are downsides: constructing full formal ontologies is laborious and needs domain expertise. Moreover they can be rigid and adoption sometimes lags. The same biomedical study notes a low number of papers actually providing formal evaluation or empirical tests for claims about foundational ontologies.

4.1.3 The Current Role Of Knowledge Graphs

Knowledge Graphs have become more prominent. Some of the things they enable and are being used for include:

- Large-scale data integration and linking: Because data is increasingly distributed, heterogeneous and unstructured, KGs are helpful to connect data, provide links and resolve semantic heterogeneity.
- Machine Learning, Embeddings, AI hybrids: There's a trend of combining KGs with ML methods: embeddings, graph neural networks, multimodal knowledge graphs. For example, "Knowledge Graphs Meet Multi-Modal Learning: A Comprehensive Survey" (2024) reviews how KGs are being used alongside multi-modal data (images, text, etc.), and how tasks like KG completion, entity alignment, etc. are being addressed [6].
- Explainability and AI Systems KGs are fertile ground for explainable or rather interpretable AI because their graph structure and schema (ontologies) enable human-readable relationships and traceability.

4.2 Conclusion

This thesis has addressed the challenges of metadata representation in the Semantic Web with a particular focus on the design and development of a domain-specific vocabulary to support consistent and interoperable annotations. The creation of this vocabulary demonstrates the feasibility and benefits of reusing established standards, while extending them with new properties tailored to describe the structure, provenance and other relevant characteristics of ontologies. This approach ensures that both domain knowledge and metadata about ontologies themselves can be represented in a coherent and machine-interpretable manner. In summary, the work demonstrates that careful vocabulary design, awareness of existing standards and thoughtful integration of semantic tools are essential for advancing the interoperability, quality and usability of ontology-based metadata, laying a foundation for future research and intelligent, data-driven applications in the Semantic Web. To conclude, I will now give an overview of current trends in this domain as well as an outlook regarding possible future work, preceded by some brief personal conclusion on my part.

At the beginning of this Bachelor's Thesis, I had no experience with ontologies at all, only some basic knowledge of what the Semantic Web is and what it's goals are. I had never encountered the word ontology before, which honestly surprised me, given that it is a very central and important concept in relation to the entire World Wide Web. At first, it was relatively complex to delve into the subject matter, partly due to the multitude of different kinds of ontology representations, but after a certain amount of familiarization, I had a good overview. Then I wanted to get started right away and immediately began defining my own properties for my vocabulary (which always were newly defined properties specified as sub-properties of already established properties from, for example, the VANN vocabulary). This approach was then revised several times, as the reuse of existing properties is also a very essential part of the entire system. Overall, I found the work on this topic very interesting, from researching various metadata in different vocabularies to implementing them in my own vocabulary. Finally, it was also very informative to explore the current state of knowledge graphs and the usage status of ontologies.

4.3 Future Work

This thesis opens several directions for future research and practical development in ontology-based metadata modeling and knowledge representation. One potential avenue is the creation of a tool to assist ontology engineers in correctly annotating metadata. Such a tool could support semantic validation, enforce consistency with existing ontologies and reduce errors in large or complex knowledge bases. Beyond tooling, future work could focus on integrating automated reasoning and machine-assisted suggestions. Large Language Models (LLMs) have shown potential for recommending appropriate ontology classes, properties and relationships based on textual context [25]. Incorporating these techniques could streamline the annotation process while maintaining semantic accuracy. Another important direction is the enhancement of visualization and user interaction. Interactive interfaces that represent class hierarchies, property relationships and metadata dependencies could make ontology management more accessible, particularly for non-experts and facilitate collaborative editing.

Finally, systematic empirical evaluation is necessary to assess the practical impact of such approaches. Studies could investigate usability, annotation accuracy and efficiency gains in real-world workflows, providing guidance for further refinements and demonstrating the added value of supporting tools in ontology engineering.

Bibliography

[1] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. Cambridge, MA, USA: MIT Press, 2004.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. Cambridge, UK: Cambridge University Press, 2010. DOI: 10.1017/CBO9780511711787.

[3] C.H. Bernabé, N. Queralt-Rosinach, and V.E Silva Souza. "The Use of Foundational Ontologies in Biomedical Research". In: *Journal of Biomedical Semantics* (2023). Online: BioMed Central. DOI: 10.1186/s13326-023-00300-z.

[4] Tim Berners-Lee, Roy Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. Tech. rep. RFC 3986. Mountain View, CA, USA: IETF, 2005.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* (2001). New York, USA: Scientific American, Inc., pp. 34–43.

[6] Zhuo Chen, Yichi Zhang, Yin Fang, Yuxia Geng, Lingbing Guo, Xiang Chen, Qian Li, Wen Zhang, Jiaoyan Chen, Yushan Zhu, Jiaqi Li, Xiaoze Liu, Jeff Z. Pan, Ningyu Zhang, and Huajun Chen. "Knowledge Graphs Meet Multi-Modal Learning: A Comprehensive Survey". In: *arXiv preprint* abs/2402.05391 (2024). Online: arXiv. URL: https://arxiv.org/abs/2402.05391.

[7] Mathieu D'Aquin and Aldo Gangemi. "Is There a Beauty in Ontologies?" In: *Applied Ontology* (2011). Amsterdam, Netherlands: IOS Press. DOI: 10.3233/AO-2011-0093.

[8] Biswanath Dutta, Anne Toulet, Vincent Emonet, and Clement Jonquet. "New Generation Metadata Vocabulary for Ontology Description and Publication". In: *Metadata and Semantic Research*. Ed. by Emmanouel Garoufallou, Sirje Virkus, Rania Siatri, and Damiana Koutsomiha. Cham: Springer International Publishing, 2017, pp. 173–185. ISBN: 978-3-319-70863-8.

[9] Lisa Ehrlinger, Johannes Schrott, Martin Melichar, Nicolas Kirchmayr, and Wolfram Wöß. "Data Catalogs: A Systematic Literature Review and Guidelines to Implementation". In: *DEXA Workshops*. Cham, Switzerland: Springer, 2021, pp. 148–158.

[10] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. London, UK: Springer London, 2004. URL: https://doi.org/10.1007/b97353.

[11] Thomas R. Gruber. "A Translation Approach to Portable Ontology Specifications". In: *Knowledge Acquisition* 5.2 (1993). Amsterdam, Netherlands: Elsevier, pp. 199–220. DOI: 10.1006/knac.1993.1008.

[12] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web*. Berlin, Heidelberg: Springer, 2008.

[13] Matthew Horridge and Sean Bechhofer. "The OWL API: A Java API for OWL Ontologies". In: *Semantic Web: - Interoperability, Usability, Applicability* 2 (2011). Amsterdam, Netherlands: IOS Press, pp. 11–21. DOI: 10.3233/SW-2011-0025.

[14] Sayed Hoseini, Johannes Theissen-Lipp, and Christoph Quix. "A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes". In: *Journal of Web Semantics* 81 (2024). Amsterdam, Netherlands: Elsevier, p. 100819. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2024.100819. URL: https://www.sciencedirect.com/science/article/pii/S1570826824000052.

[15] Mayank Kejriwal. *Knowledge Graphs: A Practical Review of the Research Landscape*. Vol. 13. Basel, Switzerland: MDPI, 2022, p. 161. DOI: 10.3390/info13040161.

[16] Leyuan Ma and Timo Hartmann. "A proposed ontology to support the hardware design of building inspection robot systems". In: *Advanced Engineering Informatics* 55 (2023). Amsterdam, Netherlands: Elsevier, p. 101851. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2022.101851. URL: https://www.sciencedirect.com/science/article/pii/S1474034622003093.

[17] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Tech. rep. KSL-01-05. Stanford, CA, USA: Stanford Knowledge Systems Laboratory, 2001. URL: https://www.researchgate.net/publication/243772462_Ontology_Development_101_A_Guide_to_Creating_Your_First_Ontology.

[18] Tassilo Pellegrini and Andreas Blumauer. *Semantic Web – Wege zur vernetzten Wissensgesellschaft*. x.media.press. Berlin, Heidelberg: Springer, 2006.

[19] Silvio Peroni, David Shotton, and Fabio Vitali. "One Year of the OpenCitations Corpus". In: *The Semantic Web – ISWC 2017*. Ed. by Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin. Cham: Springer International Publishing, 2017, pp. 184–192. ISBN: 978-3-319-68204-4.

[20] Maria Poveda. "FAIR principles and ontologies for the Semantic Web: the meeting point". In: *Proceedings of Linked Archives International Workshop 2021*. Ed. by Carla Teixeira Lopes, Cristina Ribeiro, Franco Niccolucci, Irene Rodrigues, Nuno Freire, et al. Vol. 3019. CEUR Workshop Proceedings. Aachen, Germany: CEUR-WS.org, 2021.

[21] Marta Contreiras Silva, Patrícia Eugénio, Daniel Faria, and Catia Pesquita. "Ontologies and Knowledge Graphs in Oncology Research". In: *Cancers* 14.8 (2022). Basel, Switzerland: MDPI. ISSN: 2072-6694. DOI: 10.3390/cancers14081906. URL: https://www.mdpi.com/2072-6694/14/8/1906.

[22] Barry Smith. "Ontology and Information Systems". In: *The Blackwell Guide to the Philosophy of Computing and Information*. Ed. by Luciano Floridi. Oxford, UK: Blackwell, 2003, pp. 155–166.

[23] John F. Sowa. *Ontology, Metadata and Semiotics*. Online resource. Lexington, MA, 2000. URL: http://www.jfsowa.com/ontology/ontometa.htm.

[24] Georgiana Stănescu (Nicolaie) and Simona-Vasilica Oprea. "Recent Trends and Insights in Semantic Web and Ontology-Driven Knowledge Representation Across Disciplines Using Topic Modeling". In: *Electronics* 14.7 (2025). Basel, Switzerland: MDPI. ISSN: 2079-9292. DOI: 10.3390/electronics14071313. URL: https://www.mdpi.com/2079-9292/14/7/1313.

[25] Frank Wawrzik, Rami Dhouib, Khushnood Adil Rafique, and Christoph Grimm. "Hybrid AI Approach for Knowledge Graph Construction". In: *Conference: Proceedings of the Workshop, Poster and Demonstration Sessions at IJCKG 2023 co-located with 12th International Joint Conference on Knowledge Graphs (IJCKG 2023)* (Mar. 2024). Tokyo, Japan.

[26] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, and et al. "The FAIR Guiding Principles for Scientific Data Management and Stewardship". In: *Scientific Data* 3 (2016). Online: Nature Research, p. 160018. DOI: 10.1038/sdata.2016.18.

Appendix

Classes

```
#################################################################
# Classes
#################################################################
aowm:Resource
  a owl:Class ;
  rdfs:subClassOf owl:Thing .
  rdfs:label "Resource"@en ;
  rdfs:comment "Generic parent element for all domain classes in the AOWM vocabulary.
 Used as the common superclass for ontology metadata resources (Elements, Version, ...)."@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:Ontology
  a owl:Class ;
  rdfs:subClassOf aowm:Resource, mod:SemanticArtefact, owl:Ontology ;
  rdfs:label "Ontology"@en ;
 rdfs:comment "A formal specification of a conceptualization, described using RDF-based standards."@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date ;
  rdfs:seeAlso owl:Ontology .

aowm:Distribution
  a owl:Class ;
  rdfs:subClassOf aowm:Resource, mod:SemanticArtefactDistribution ;
  rdfs:label "Distribution of an ontology"@en ;
  rdfs:comment "A specific representation and/or serialization of an ontology."@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:Catalog
  a owl:Class ;
  rdfs:subClassOf aowm:Resource, mod:SemanticArtefactCatalog ;
  rdfs:label "Data Catalog"@en ;
 rdfs:comment "A dedicated web-based system that fosters the availability, discoverability and long-term preservat
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:CatalogRecord
  a owl:Class ;
  rdfs:subClassOf aowm:Resource, mod:SemanticArtefactCatalogRecord ;
```

```
  rdfs:label "Ontology Data Catalog Record"@en ;
 rdfs:comment "A record in a catalog, describing the registration of a single ontology."@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:Version
  a owl:Class ;
  rdfs:subClassOf aowm:Resource ;
  rdfs:label "Version"@en ;
  rdfs:comment "A specific version of an ontology or metadata record."@en ;
  rdfs:seeAlso mod:SemanticArtefactVersion ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:Agent
  a owl:Class ;
  rdfs:subClassOf aowm:Resource, foaf:Agent ;
  rdfs:label "Agent"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  rdfs:comment "A resource that acts or has the power to act."@en ;
  dct:issued "2026-01-31"^^xsd:date ;

aowm:Role
  a owl:Class ;
  rdfs:label "Role"@en ;
  rdfs:comment "A role assumed by an agent in relation to a resource."@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date ;
  rdfs:seeAlso prov:Role .

aowm:Person
  a owl:Class ;
  rdfs:subClassOf aowm:Agent ;
  rdfs:label "Person"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date >;
  rdfs:seeAlso foaf:Person .

aowm:Organization
  a owl:Class ;
  rdfs:subClassOf aowm:Agent ;
  rdfs:label "Organization"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date ;
  rdfs:seeAlso foaf:Organization .

aowm:SerializationFormat
  a owl:Class ;
  rdfs:subClassOf aowm:Resource ;
  rdfs:label "Serialization Format"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .
```

```
aowm:DegreeOfMaturity
  a owl:Class ;
  rdfs:subClassOf aowm:Resource ;
  rdfs:label "Degree Of Maturity"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .

aowm:MetadataCreationMethod
  a owl:Class ;
  rdfs:subClassOf aowm:Resource ;
  rdfs:label "Creation Method Of Metadata"@en ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> ;
  dct:issued "2026-01-31"^^xsd:date .
```

Selection Of Properties

```
###metadata about metadata####
aowm:metadataCreationMethod
  a owl:DatatypeProperty
  rdfs:label "creation method"@en ;
  rdfs:domain aowm:Ontology,
  aowm:Catalog ;
  rdfs:range  aowm:MetadataCreationMethod ;
 dct:description: "Creation method of metadata - manual, automatic, semi-automatic"@en ;
  dct:issued          "2026-01-30"^^xsd:date ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .

aowm:metadataSource
  a owl:DatatypeProperty
  rdfs:label "metadata source"@en ;
  rdfs:domain aowm:Ontology,
  aowm:Catalog ;
  rdfs:range xsd:anyURI ;
  dct:description: "Source of metadata, e.g. "repository", "person",..."@en ;
  dct:issued          "2026-01-30"^^xsd:date ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .

aowm:metadataGeneratedBy
  a owl:DatatypeProperty
  rdfs:label "metadata generated by"@en ;
  rdfs:domain aowm:Ontology,
  aowm:Catalog ;
  rdfs:range xsd:string ;
  dct:description: "System or software which created the metadata"@en ;
  dct:issued          "2026-01-30"^^xsd:date ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .

aowm:metadataDerivedFrom
```

```
  a owl:DatatypeProperty
  rdfs:label "metadata derived from"@en ;
  rdfs:domain aowm:Ontology,
  aowm:Catalog ;
  rdfs:range xsd:anyURI ;
 dct:description: "Other metadata or artifacts, from which this metadata were derived"@en ;
  dct:issued          "2026-01-30"^^xsd:date ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .

aowm:metadataCreationDate
  a owl:DatatypeProperty
  rdfs:label "metadata creation date"@en ;
  rdfs:domain aowm:Ontology,
  aowm:Catalog ;
  rdfs:range xsd:dateTime ;
  dct:description: "Time when metadata was initial created."@en ;
  dct:issued          "2026-01-30"^^xsd:date ;
  rdfs:isDefinedBy <http://dqm.faw.jku.at/ontologies/aowm> .
```