Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo 35

Settore Concorsuale: 01/B1
Settore Scientifico Disciplinare: INF/01

Empirical Ontology Design Patterns

Presentata da: Valentina Anita Carriero

Coordinatore Dottorato
Ilaria Bartolini

Supervisore
Valentina Presutti

Esame finale anno 2023

‘Sul sentiero tortuoso riesce a immaginare di camminare verso qualcosa,
e non in mezzo a qualcosa.’
(Dimora di Ruggine, Khadija Abdalla Bajaber)

To the beautiful souls that held my hand during this winding path.

Abstract

In recent years, knowledge graphs (KGs) and ontologies have been widely adopted for modelling any kind of domain. Many of them are released openly, which benefits those who start new projects because they have a wide choice for ontology reuse and for linking to existing data. Nevertheless, understanding the content of an ontology or a knowledge graph is far from straightforward, and existing methods only partially address this issue. For example, exploring and comparing multiple ontologies is a tedious manual task. This thesis is based on the assumption that identifying the Ontology Design Patterns (ODPs) used in an ontology or a knowledge graph contributes to address this problem. Most of the time the reused ODPs are not explicitly annotated, or their reuse may be unintentional. Therefore, there is a challenge to automatically identify ODPs in existing ontologies and knowledge graphs, which is the main focus of this research work. To the best of our knowledge, there is lack of tools to effectively support this task. This thesis contributes to the state of the art by analysing the role of ODPs in ontology engineering, through experiences in real-world ontology projects, placing this analysis in the wider context of existing ontology reuse approaches and implementations. Moreover, this thesis introduces (i) a novel method for extracting *empirical* ontology design patterns (EODPs) from ontologies, and (ii) a novel method for extracting EODPs from knowledge graphs, whose schemas are implicit.

The first method is able to group the extracted EODPs in clusters that are named *conceptual components.* Each conceptual component represents a generalised modelling problem, for example *representing collections*. As EODPs are fragments possibly extracted from different ontologies, some of them will fall in the same cluster, meaning that they are expected to be implemented solutions to the same modelling problem, e.g. different solutions to model collections. Hence, EODPs and conceptual components enable the empirical observation of modelling solutions to common modelling problems adopted by different ontologies, therefore supporting their comparison.

The second method extracts EODPs from a knowledge graph as sets of probabilistic axioms and constraints involving the classes and the properties instantiated in the KG. The probabilistic axioms are annotated with relevant provenance information, e.g. the KG they were observed from. These EODPs may support KG inspection and comparison, as they provide insights on how certain entities are described in a KG and linked to other resources. Both methods are applied to ontologies and knowledge graphs largely adopted and reused, such as Wikidata.

An additional contribution of this thesis is an ontology for annotating ODPs in ontologies and knowledge graphs, which can be used as a basis for both manual and automatic annotation.

Contents

List of Tables **xi**

List of Figures **xiii**

1 Introduction **1**

1.1 The semantic web: principles and standards 1

1.2 Context and research questions 3

1.2.1 The lack of support to choose and implement an effective ontology reuse strategy 4

1.2.2 The role of ontology design patterns in supporting ontology engineering in real-world projects 5

1.2.3 Modelling solutions to common modelling problems 6

1.2.4 How to annotate ontology design patterns 8

1.2.5 Identifying patterns emerging from knowledge graphs 9

1.3 Summary of the chapters and contributions 9

2 Pattern-based ontology design and reuse **15**

2.1 Current approaches to ontology selection and reuse: benefits, limits and challenges 15

2.1.1 Motivations guiding ontology selection 16

2.1.2 Policies and implementation strategies for ontology reuse . . . 18

2.2 Pattern-based ontology design and reuse: real-world projects 20
2.2.1 eXtreme Design 21
2.2.2 ArCo: a pattern-based ontology network and knowledge graph on cultural heritage 23
2.2.2.1 An overview 23
2.2.2.2 Mapping competency questions to ODPs 23
2.2.2.3 Advancements and support to ontology reuse 30
2.2.3 Polifonia: a pattern-based ontology network and knowledge graph on musical heritage 37
2.2.3.1 An overview 37
2.2.3.2 Mapping competency questions to ODPs 37
2.2.3.3 Advancements and support to ontology reuse 41

3 Observing patterns from ontologies 47
3.1 Motivation and problem addressed 48
3.2 Related work 51
3.3 Input ontology corpora 53
3.3.1 Cultural Heritage ontology corpus 53
3.3.2 Conference ontology corpus 55
3.4 Approach 56
3.4.1 Intensional ontology graph 58
3.4.2 Community detection 61
3.4.3 Clustering and catalogue generation 63
3.5 Experiment and results 66
3.6 Evaluation and discussion 68
3.6.1 Manual inspection of communities 68
3.6.2 Clustering: similarity 69
3.6.3 Clustering: manual inspection 69

3.6.4 Evaluation against an ontology engineering task 70
3.6.5 The ArCo use case 73

4 Annotating patterns in ontologies and knowledge graphs 79
4.1 Motivation 80
4.2 Related work 82
4.3 OPLaX 85
4.3.1 Pattern level 85
4.3.2 Conceptual component level 86
4.3.3 Pattern instance level 87
4.4 Use cases 87
4.4.1 ArCo ontology network 88
4.4.2 Conceptual components and ODPs catalogue from a corpus of ontologies 89
4.4.3 Pattern-based visualization of knowledge graphs 90

5 Observing patterns from data 93
5.1 Method for extracting empirical patterns from a KG 94
5.1.1 Related work 94
5.1.2 Method 97
5.2 Empirical patterns from Wikidata 100
5.2.1 Motivation 100
5.2.2 Input 103
5.2.3 Wikidata empirical patterns on music 104
5.2.4 Wikidata empirical patterns on art, architecture, and archaeology (AAA) 110
5.2.5 Discussion and evaluation 115
5.2.5.1 Music 115
5.2.5.2 Art, architecture, and archaeology (AAA) 121

5.2.5.3 Music and AAA 125
5.2.5.4 Evolution of music and AAA patterns across two versions of Wikidata 127

6 Conclusion and future work 131
6.1 How we answered our research questions 134
6.2 Future work 136

List of Tables

3.1 Corpora of ontologies: statistics 56
3.2 The number of hierarchical relations among clusters per level of strength. 67
3.3 Correlation between reference alignments (*AA, CA, AML,* $AML_{.90}$ and $AML_{.99}$) with the sets I, E, $E_{0.1}$, $E_{0.2}$, $E_{0.3}$, $E_{0.4}$. 72

5.1 Most populated classes in the Wikidata music subKG. 105
5.2 Statistics of selected properties and triplets for each music pattern. . 106
5.3 Most populated classes in the Wikidata art, architecture, and archaeology subKG. 111
5.4 Statistics of selected properties and triplets for each AAA pattern. . . 112
5.5 Percentages of coverage of the music patterns properties in the KG. . 117
5.6 Comparison between properties recommended by WikiProject Music and properties included in our pattern for record labels. 119
5.7 Comparison between properties recommended by WikiProject Music and properties included in our patterns for releases. 120
5.8 Percentages of coverage of the AAA patterns properties in the KG. . 122
5.9 Comparison between properties recommended by WikiProject Museums and properties included in our pattern for museums. 126

List of Figures

2.1 Prefixes used in the next figures. 24
2.2 Time indexed situation ODP reused for modelling different types of locations of a cultural property. 26
2.3 Situation ODP reused for representing the coin issuance. 28
2.4 Situation ODP reused for the authorship attribution. 28
2.5 An instance of the model in Figures 2.3 and 2.4: a numismatic property involved in a coin issuance and a preferred authorship attribution. 29
2.6 The new pattern Recurrent Situation Series as implemented in ArCo. 31
2.7 Time indexed situation ODP reused for modelling time indexed names of entities. 40
2.8 Time indexed situation ODP reused for modelling the creation of pieces of art. 42

3.1 Implementations of the Membership CC from two different ontologies. 50
3.2 Approach for conceptual components extraction. 57
3.3 Example of OWL EODPs and their corresponding intensional ontology graphs. Blue rectangles indicate object properties, green rectangles data properties. 59
3.4 Example of communities detected from two ontologies of the CH corpus. 62
3.5 Virtual document disambiguation, frame detection and clustering on the communities from Fig. 3.4. 64

3.6 Top-down ODP and Empirical ODP about copyright from ArCo ontology network. 78

4.1 OPLaX, the Ontology Pattern Language eXtended. 86
4.2 An example of a specialized ODP annotated with OPLaX. 89
4.3 An example of a conceptual component annotated with OPLaX. . . . 91
4.4 An example of a pattern instance annotated with OPLaX. 92

5.1 Method for extracting empirical patterns from a KG. 97
5.2 The album pattern. 106
5.3 Classes and properties used for annotating the probabilistic pattern. . 109
5.4 The museum pattern. 112

Chapter 1

Introduction

In 2001, Tim Berners-Lee, the inventor of the web, envisioned an extension of it, in which content would become meaningful to computers: the *semantic web* [7]. While the web of documents was originally designed for humans to read, the semantic web would make data available on the web machine-readable, i.e. processable by computers, by encoding its semantics. Thus, a computer would be able to process not only the structure of a web page (e.g. this is a paragraph, this is an image), but also the meaning of its content (e.g. this is a person, this person was born in this country, this person currently works for this company), unleashing a “revolution of new possibilities” [7].

1.1 The semantic web: principles and standards

Linked Data (LD) lies at the heart of the semantic web: it can be defined as a set of principles for sharing and interlinking machine-readable data(sets) on the web [47]. When combined with open data, that is, when this data can be freely (re)used and distributed by anyone, it is called Linked Open Data (LOD). The Linked Data (LD) initiative enables the easy exposing, sharing, and connecting of structured data on the web [90].

The World Wide Web Consortium[1] (W3C) takes care of standardisation for semantic web, and the Resource Description Framework[2] (RDF) is the basic layer of the semantic web. It has been originally designed as a metadata data model, consisting of a set of W3C specifications for modelling information. With RDF, "things" on the web are identified using web identifiers (i.e. *Uniform Resource Identifiers*, URIs), and are described through statements about resources.

RDF statements are called *triples*, and consist of a *subject* (a resource), a *predicate* (a relationship between the subject and the object), and an *object*, which can be either a resource or a value. For example, the triples

`ex:Momo` `rdf:type` `ex:Dog` .

`ex:Momo` `ex:livesIn` `ex:Bologna` .

state that Momo is a dog and lives in Bologna. RDF triples can be seen as a graph: nodes are resources (e.g. real-world entities, concepts) or values (e.g. strings), and edges represent semantic relations (properties) between nodes. While the property `ex:livesIn` has been invented for this example, the `rdf:type` property is part of the RDF built-in vocabulary[3] and states that the subject is an instance of a class.

RDF Schema (RDFS)[4] vocabulary is an extension of the basic RDF vocabulary, and allows for a better structuring of resources by e.g. supporting the definition of hierarchies of both classes and properties (`rdfs:subClassOf`, `rdfs:subPropertyOf`).

Complementary to RDF and RDFS, OWL (Web Ontology Language)[5] is a logic-based language for representing richer and more complex knowledge about the meaning of things, and relations between things. So, RDF, RDFS and OWL allow to formally represent some knowledge through the definition of concepts within a domain, and relations that can formally exist between those concepts [44], providing

[1]https://www.w3.org/
[2]https://www.w3.org/RDF/
[3]https://www.w3.org/1999/02/22-rdf-syntax-ns#
[4]https://www.w3.org/TR/rdf-schema/
[5]https://www.w3.org/TR/owl-ref/

a shared and formally defined vocabulary (an ontology) to talk about real-world entities. A knowledge graph (KG) is usually built when an ontology is applied to a set of individuals, so it combines the formalization of the domain with the actual data that instantiate the concepts that have been formally defined. The SPARQL[6] query language makes it possible to retrieve information from the web of data, i.e. from data represented using RDF as data format.

1.2 Context and research questions

"The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data."[7].

Exploration is a key point for the semantic web vision. In recent years, there was an explosion of knowledge graphs and ontologies, that have been widely adopted for modelling any kind of domain. Many of them are released openly, providing ontology engineers – and any user of semantic web resources – with a wide choice of ontologies to be reused as well as knowledge graphs to reuse and link to. This explosion has led to the need to support the exploration and understanding of such structured data, in order to be able to effectively exploit it [25]. Indeed, understanding the content of an ontology or a knowledge graph is far from being straightforward, and existing methods only partially address this issue. For example, comparing several ontologies is a tedious manual task. Being able to understand and compare existing ontologies is crucial for performing various ontology engineering tasks, including ontology reuse [56]: reusing existing ontologies across different knowledge graphs is recommended in order to foster interoperability and facilitate knowledge reuse in the semantic web. However, if there is no actual support to understand what is modelled inside an ontology, the selection and reuse process becomes more error-prone and time-consuming.

Starting from the argument that there are classes of problems in ontology design

[6]https://www.w3.org/TR/rdf-sparql-query/

[7]https://www.w3.org/DesignIssues/LinkedData.html

that can be addressed by using common solutions, Ontology Design Patterns (ODPs) can support reusability of ontologies [12], and of knowledge graphs relying on such ontologies. Indeed, ontologies may be (un)intentionally designed as a composition of small, reusable building blocks, i.e. the ODPs, that provide modelling solutions to recurrent modelling problems, e.g. *modelling collections*. Hence, they inherently have a modular structure, and even large ontologies can be more easily explored, compared and reused based on these smaller fragments.

However, most of the time the ODPs used in an ontology or knowledge graph are not explicitly annotated, and their reuse may be unintentional. Therefore, there is a challenge to automatically identify ODPs in existing ontologies and knowledge graphs. To the best of our knowledge, there is lack of tools to effectively support this task [20].

Therefore, our main research question is the following:

> **RQ**$_0$: *What ontology design patterns can be observed in existing ontologies and knowledge graphs?*

In the remainder of this chapter, we will formulate more detailed research questions, after discussing their background and context.

1.2.1 The lack of support to choose and implement an effective ontology reuse strategy

Ontology reuse is one of the main ontology engineering tasks our research work may potentially impact on. Indeed, we start from the assumption that identifying ontology design patterns in ontologies (and knowledge graphs) make them more understandable in terms of what is inside such ontologies (and KGs). A good understanding of existing ontologies is needed in order to effectively select them for reuse. Based on these premises, we considered relevant to analyse the state of the art of ontology reuse practices and implementations.

There exists a vast literature on ontology reuse, which provides definitions and requirements. However, there are many factors that may affect the decision for an ontology selection and reuse policy, and its implementation. On the one hand, ontology reuse policies are usually intertwined with ontology design methodologies. Methods originating in software engineering, philosophy and cognitive science have emerged, and include: competency questions [45], foundational ontologies [36], ontology design patterns [34, 37, 51]. Such methods are based on the preliminary expression of the general cognitive requirements that should lead an ontology project. Furthermore, social motivations may influence the reuse strategies adopted by an ontology designer. International institutions and standardisation bodies tend to support the reuse of (their) standard and shared ontologies, and adopt their own common practices; domain experts may suggest the reuse of popular and/or standard ontologies as more straightforward and authoritative; ontologists tend to give priority to what they consider the best ontology modelling practices based on their expertise, etc. So, even if ontology reuse is a crucial aspect for the evolution of the semantic web, and should support semantic interoperability, it is still sometimes difficult to define shareable good practices and pragmatic guidelines about which ontology reuse approach should be adopted when developing a new ontology. This often leads to taking solutions on a case-by-case basis. Thanks to this investigation of the current landscape of ontology selection and reuse approaches, and based on existing ontology projects that worked as use cases (see the following section), we were able to outline some useful guidelines for implementing a pattern-based reuse approach.

1.2.2 The role of ontology design patterns in supporting ontology engineering in real-world projects

As previously mentioned, ontology reuse approaches are usually motivated by ontology design methodologies. There exist many methodologies, including [32, 71, 85, 68], but some of them lack actual guidelines and support for collecting requirements that should guide the ontology development, others do not define a specific workflow for developing the ontology, or they do not include evaluation/testing steps.

An ontology engineering methodology, currently used in different ontology projects, is eXtreme Design (XD) [9, 12], which is iterative and incremental. XD provides methodological support for incrementally addressing small sets of requirements, thus minimising the impact of changes in incremental releases. eXtreme Design addresses all main phases of the ontology design process: from the collection of requirements to the testing activities. Its guidelines have been implemented, and updated and integrated, in the context of actual ontology projects. Specifically, XD focuses on, and provides guidelines for, the use of ontology design patterns (ODPs): as previously mentioned, an ODP provides a modelling solution to a recurrent modelling problem, in the form of a small template ontology. The recommended ontology reuse process is also based on ontology design patterns, which can be found in ODPs catalogues[8], or extracted from existing ontologies and replicated in the ontology under development. The claim of XD is that ODPs support, and make it more effective, ontology selection and reuse, and this has been proven through multiple experiments [9, 12].

Two ontology projects we contributed to during the PhD, i.e. ArCo [20] and Polifonia[9], allowed us to experience firsthand the usefulness of the (re)use of ontology design patterns in ontology engineering activities, including ontology reuse. This motivated our aim to extract ontology design patterns from ontologies and knowledge graphs. We report in the thesis our direct experience and contributions to these two projects, with a special focus on the role of ontology design patterns for ontology reuse.

1.2.3 Modelling solutions to common modelling problems

As previously mentioned, being able to explore and understand (large) ontologies has an impact on many ontology engineering tasks, including (but not limited to) ontology selection and reuse.

[8]E.g. https://ontologydesignpatterns.org/

[9]https://polifonia-project.eu/

According to [29], existing ontology visualisation tools fail in offering overviews of big ontologies – which are the ones that would most benefit from effective visualisations for their exploration. None of them allow to compare different ontologies simultaneously. Most summarisation methods only address the data level, e.g. to return a summary of a knowledge graph in the form of a subset of triples. In this way, they reduce its size and give an insight into what is inside the knowledge graph through a relevant subgraph [73, 25]. There exist some summarisation methods that focus on the ontology level. However, they are based on *extractive* approaches that select and return a subset of *key* nodes (classes, properties) from the original ontology, presented as an ontology summary [73].

We believe that an overall understanding of an ontology requires more than viewing its key concepts: it would need to consider all the *facts* an ontology models, all the modelling issues it addresses. That is, for example, not only knowing that the two main concepts of an ontology are e.g. *musician* and *musical composition*, but also being aware that there is a fragment of the ontology answering the question about *which musical compositions have been played by a musician during a musical performance*. Additionally, there is no method actually supporting a comparison between multiple ontologies [29], in order to understand which solutions may be proposed by different ontologies to a specific modelling problem, e.g. different solutions for modelling collections. When multiple ontologies are to be analysed, e.g. for selecting the more pertinent ontology – or ontology fragment - to be reused, this activity would benefit from the possibility of exploring, and comparing, all ontologies at once, rather than exploring one ontology at a time, and comparing them as a separate step.

Based on these premises, our first research question is:

> **RQ**$_1$: *Are there different modelling solutions implemented in multiple ontologies addressing the same modelling problem? Is it possible to automatically identify and classify them?*

1.2.4 How to annotate ontology design patterns

Regardless of whether ontology design patterns are (re)used in an intentional or unintentional way in ontologies[10], ontology selection and reuse would benefit from annotations of the patterns included in such ontologies and knowledge graphs. Indeed, using existing ontologies as input to build new ontologies is a difficult, non-trivial and time-consuming process, and is often hampered by an inadequate, if not even missing, documentation accompanying such ontologies. Based on the assumption that ontology design patterns play a role in supporting ontology reuse, pattern-based ontology development and reuse should go through this annotation task, which may ease the process of exploring and understanding an ontology (and, consequently, a knowledge graph). Other ontology engineering tasks would benefit from these annotations, like a pattern-based ontology matching, as the process of finding correspondences (i.e. mappings) between ontology fragments (rather than individual entities) belonging to different ontologies, producing a set of alignments.

The need for a language for annotating ontology design patterns has been already acknowledged [11], and led to some proposals. This language can work as a basis for both manual and automatic annotation. [53] and [50] propose complementary, simple representation languages for ontology design patterns, which make use of OWL annotation properties for documenting them, while [77] introduces a framework aiming at supporting goal-oriented reuse of ontologies. However, these languages' expressiveness requires further improvements: for instance, it is not possible to indicate the modelling problem addressed by an ODP at a more abstract level, nor to annotate the data that instantiate an ODP.

This leads to our second research question:

RQ$_2$: *What are the relevant concepts and relations to describe the ontology design patterns used in ontologies and knowledge graphs?*

[10]And possibly instantiated in knowledge graphs.

1.2.5 Identifying patterns emerging from knowledge graphs

While a better understanding and exploration of ontologies is crucial for making more accessible and usable the knowledge graphs that rely on such ontologies, not all knowledge graphs are based on explicitly modelled ontologies. In some cases, knowledge graphs may be built directly from actual data, without going through an ontology modelling process: the ontology emerges from the usage of *blurred* concepts and relations in the knowledge graph. Moreover, even if such an explicit ontological resource does exist, the population of the ontology through one (or more than one) knowledge graph may give important insights about the actual usage of the ontology in real-world use cases. There exist a number of approaches for generating sets of contraints/formal definitions/patterns from a set of data, which may exploit the graph structure (e.g. [31, 15]) or rely on knowledge graph profiling, i.e deriving sets of constraints (shapes) from summaries of knowledge graphs, as in [82].

However, as highlighted in [76], all existing approaches supporting the automatic creation of shapes generate a great number of shape constraints such that it may be difficult to verify their soundness and completeness. Moreover, most generated shapes do not suggest specific classes that the objects of a property should be member of. Considering that existing methods leave room for improvements, our last research question is:

RQ$_3$: *Is it possible to automatically identify the patterns that emerge from a knowledge graph without relying on an explicit ontology?*

1.3 Summary of the chapters and contributions

We summarise the content and contributions of each chapter of the thesis[11], reporting the publications this thesis is based on, and extends, in the following list:

- **Chapter 2:** We report in this chapter our investigations on current approaches to ontology selection and reuse, analysing their motivations and strategies.

[11]This thesis has been reviewed by professor Enrico Motta and professor Marta Sabou.

We argue that solutions for ontology reuse are often adopted on a case-by-case basis, hindering the definition of good and shared practices, and that to date there are no effective solutions for supporting the developers' decision-making process when deciding on an ontology reuse strategy. Analysing their benefits and limits, we provide some guidelines on the choice for the most appropriate ontology reuse practice. Moreover, we present two real-world use cases of how the eXtreme Design methodology has been applied for the development of domain-specific knowledge graphs (cultural heritage and music), showing how a pattern-based ontology engineering can actually support ontology engineering activities, including ontology selection and reuse. Part of the content of this chapter was published in the following publications:

– Valentina Anita Carriero, Marilena Daquino, Aldo Gangemi, Andrea Giovanni Nuzzolese, Silvio Peroni, Valentina Presutti and Francesca Tomasi. 'The Landscape of Ontology Reuse Approaches'. In: *Applications and Practices in Ontology Design, Extraction, and Reasoning.* Edited by Giuseppe Cota, Marilena Daquino and Gian Luca Pozzato. Volume 49. Studies on the Semantic Web. Amsterdam: IOS Press, 2020, pages 21–38. DOI: 10.3233/SSW200033 [18]
Individual contribution to the paper: collaboration to the analysis of the state-of-the-art, and to one of the use cases presented.

– Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Andrea Giovanni Nuzzolese, Valentina Presutti and Chiara Veninata. 'Pattern-based design applied to cultural heritage knowledge graphs'. In: *Semantic Web* 12.2 (2021), pages 313–357. DOI: 10.3233/SW-200422 [20]
Individual contribution to the paper: collaboration to the design and testing of the ontologies, and to the generation of the knowledge graph.

– Valentina Anita Carriero, Aldo Gangemi, Andrea Giovanni Nuzzolese and Valentina Presutti. 'An Ontology Design Pattern for Representing Recurrent Situations'. In: *Advances in Pattern-Based Ontology En-*

gineering, extended versions of the papers published at the Workshop on Ontology Design and Patterns (WOP). edited by Eva Blomqvist, Torsten Hahmann, Karl Hammar, Pascal Hitzler, Rinke Hoekstra, Raghava Mutharaju, María Poveda-Villalón, Cogan Shimizu, Martin G. Skjæveland, Monika Solanki, Vojtech Svátek and Lu Zhou. Volume 51. Studies on the Semantic Web. IOS Press, 2021, pages 166–182. DOI: 10.3233/SSW210013 [22]
Individual contribution to the paper: collaboration to the design of the ontology design pattern.

– Valentina Anita Carriero, Fiorela Ciroku, Jacopo de Berardinis, Delfina Sol Martinez Pandiani, Albert Meroño-Peñuela, Andrea Poltronieri and Valentina Presutti. 'Semantic integration of MIR datasets with the Polifonia ontology network'. In: *Proceedings of Late Breaking Demo Session of the 22nd Int. Society for Music Information Retrieval Conference (ISMIR 2021).* 2021 [17]
Individual contribution to the paper: collaboration to the design of the ontology network.

- **Chapter 3:** In this chapter, we propose a method to identify which are the conceptual components, and the patterns implementing them, used for modelling KGs on the web through ontologies. A conceptual component is a complex (cognitive) relational structure that a designer implements in an ontology by using ontological constructs, and that may be implemented by means of different ontology fragments - the Empirical ODPs - across different ontologies. The method is applied to two domain-specific ontology corpora. The results show the potential of extracting patterns from ontologies, and grouping them based on the common modelling issues they address. This method offers an answer to $\mathbf{RQ}_1$. The chapter is based on the following publication:

  – Luigi Asprino, Valentina Anita Carriero and Valentina Presutti. 'Extraction of Common Conceptual Components from Multiple Ontologies'. In:

Proceedings of K-CAP '21: Knowledge Capture Conference 2021. Edited by Anna Lisa Gentile and Rafael Gonçalves. ACM, 2021, pages 185–192. DOI: 10.1145/3460210.3493542 [4]
Individual contribution to the paper: design of the method and the experiments.

- **Chapter 4:** In this chapter, we address $\mathbf{RQ}_2$ by developing an ontology for annotating ontology design patterns implemented in ontologies and knowledge graphs, which reuses and extends existing languages. This annotation ontology can be the basis for both manual and automatic annotations, and supports the annotation of ODPs at both conceptual component, pattern, and instance level. This chapter is based on the following publication:

  - Luigi Asprino, Valentina Anita Carriero, Christian Colonna and Valentina Presutti. 'OPLaX: annotating ontology design patterns at conceptual and instance level'. In: *Proceedings of the 12th Workshop on Ontology Design and Patterns (WOP 2021) co-located with 20th International Semantic Web Conference (ISWC 2021).* Edited by Karl Hammar, Cogan Shimizu, Hande Küçük-McGinty, Luigi Asprino and Valentina Anita Carriero. Volume 3011. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pages 1–13 [3]
    Individual contribution to the paper: collaboration to the design of the ontology, and to two of the use cases presented.

- **Chapter 5:** This chapter contains an answer to $\mathbf{RQ}_3$ in the form of a method for empirically extracting patterns that emerge from a knowledge graph, without relying on an explicit ontology. These patterns are extracted as statistically frequent domain-property-range triplets, which are then translated into probabilistic OWL ontology design patterns. The axioms inferred from statistics in the data are generated and associated with a probability. This method has been applied to two domain-specific portions of the Wikidata knowledge graph[12]. The results of our experiments show how these patterns can integrate the current support in Wikidata to the understanding and reuse of the informally defined Wikidata ontology. The chapter is adapted, with relevant extensions[13], from the following publication:

  – Valentina Anita Carriero, Paul Groth and Valentina Presutti. 'Towards improving Wikidata reuse with emerging patterns'. In: *Proceedings of the 3rd Wikidata Workshop 2022 co-located with the 21st International Semantic Web Conference (ISWC 2022).* Edited by Lucie-Aimée Kaffee, Simon Razniewski, Gabriel Amaral and Kholoud Saad Alghamdi. Volume 3262. CEUR Workshop Proceedings. CEUR-WS.org, 2022 [23]
    Individual contribution to the paper: design of the method and the experiments.

- **Chapter 6:** This chapter presents a summary of the overall research activity, along with some open questions and future directions of research.

[12]https://www.wikidata.org/

[13]A new publication extending [23] has been submitted to the Special Issue on Wikidata of the Semantic Web Journal on 14 February 2023 [24].

Chapter 2

Pattern-based ontology design and reuse

Ontology Reuse (OR) aims at fostering semantic interoperability and integration between different data sources, and constitutes a main step of the ontology design process, strongly related to the ontology selection (OS) activities.

To date, several approaches to OR do exist, with different motivations and implementations. However, despite the vast literature, a tendency to opt for a solution on a case-by-case basis, and without following a specific methodology, can hamper the definition of shareable good practices.

In this Chapter, we (i) provide an account of existing OR solutions, addressing their benefits and limits (Section 2.1), and (ii) discuss two real-world scenarios, thoroughly describing the experience in applying a pattern-based ontology engineering methodology (eXtreme Design) to the development of two domain-specific knowledge graphs, showing how we implemented and extended it, with a special focus on the ontology reuse approach (Section 2.2).

2.1 Current approaches to ontology selection and reuse: benefits, limits and challenges

The decision about what OR approach to adopt usually depends on the project's requirements, so it is made by ontology engineers when bootstrapping a project. Firstly, ontology developers search and select candidate ontologies to be reused, and secondly, they choose the more suitable strategy for implementing ontology reuse. In Section 2.1.1 we introduce common motivations guiding the selection of ontologies to reuse, while Section 2.1.2 discusses policies for ontology reuse.

2.1.1 Motivations guiding ontology selection

Some guidelines for Linked Open Data and vocabularies publication [54, 8] provide support to ontology designers in selecting valid and documented ontologies to be reused, promoting either the reuse of standards, or the reuse of popular ontologies. Some research communities [45, 36, 86, 52, 75] and some W3 Consortium working groups (such as the Ontology Engineering and Patterns Task Force[1]) foster *explicit cognitive analysis* in OR, which may result in the reuse of ontology patterns rather than ontology terms.

OR by standardisation. OR by standardisation consists in the practice of reusing ontologies released by authoritative organisations, like ISO, W3 Consortium, and professional/community consortia. Examples include ISO standard ontologies such as CIDOC-CRM [43], W3C approved ontologies such as PROV-O [59], and community standard ontologies such as FRBRoo [78]. Standard ontologies can work as reference models for representing cross-domain information, e.g. events, people-related information, provenance, but they can also model domain-specific requirements, e.g. cultural heritage attributes as in CIDOC-CRM. Usually, standard ontologies are recommended for direct reuse to the members of a community, or can be used as reference models that can be then specialised and extended.

OR by popularity. OR by popularity consists in reusing ontologies that are considered popular, usually because they are reused in (i) many third-party ontologies [54] by importing, specialising or extending them, and in (ii) existing datasets by populating their ontology entities [79]. Popular ontologies might be designed as standards, or might be based on a cognitive analysis, but the reason they are reused is often related to their popularity. For instance, FOAF[2] is the result of an early

[1]https://www.w3.org/2001/sw/BestPractices/OEP/
[2]http://xmlns.com/foaf/spec/

exemplification (1999) of a vocabulary for the semantic web, which did not result from any standardisation or formal analysis. However, it is now a reference model for social network knowledge graphs. A further example is Good Relations [49], which is a lightweight ontology for exchanging e-commerce information. It is not the result of a standardisation process, but is based on formal requirements collected from industry and derived from a cognitive analysis. Other popular ontologies include DBpedia[3], a shallow cross-domain ontology resulting from a crowd-sourcing effort, and schema.org[4], an initiative launched by an industrial standardisation body to develop and support a common set of schemas for structured data markup on web pages, fostering search engine optimisation via semantics.

OR by cognitive analysis. OR by cognitive analysis is based on the collection and definition of the requirements of an ontology project as primary source of decisions. This may lead to design novel models, or to use existing design components, such as foundational ontologies [36] or ontology design patterns [52]. The cognitive analysis justifies and supports decisions about the reuse or the novel design of ontology terms and axioms. Specifically, whether to reuse something or not is supposed to depend on the requirements of the project. That is, if a conceptualisation has already been specified in an existing ontology, it can be directly reused or aligned. If not, reuse is not recommended, and generic solutions might be specialised. Cognitive analysis inherits methods from philosophy, cognitive science and software engineering, including *competency questions* (CQs) [45], that is questions expressing the requirements the ontology should satisfy, formal ontological analysis reusing patterns already encoded in foundational ontologies, e.g. DOLCE [36], task-oriented quality assessment [35], etc. A research community has grown to integrate and support cognitive analysis by means of Ontology Design Patterns (ODPs) [34, 37, 52].

Benefits and limits. Popularity-based metrics are widespread when looking for ontologies to reuse, and can avoid time-consuming selection activities when searching

[3]http://dbpedia.org/ontology/

[4]https://schema.org/

for general purpose ontologies. Moreover, reusing ontologies that are already popular can increase the chances that data will be reused by other applications [48]. Such common practice in the Linked Data community is motivated by the assumption that the semantics of an ontology is clear and intuitive if it is popular and commonly used in the web of data. Reusing ontologies that are well-known, considering that their ontological solutions are well established and possibly subject to a shared understanding, does foster semantic interoperability and homogenization between different resources, making it easier to build rational agents able to automatically share, reuse, exchange, and reason on the knowledge at web scale.

However, selecting the ontologies that fit for the purpose is, to date, a highly subjective and error-prone task, usually performed manually and in an intuitive way. Moreover, the only metrics for ontology selection currently available are based on popularity, which may induce biased behaviours in OR practices, like boosting only OR by popularity or OR by standardisation. Additionally, more detailed information about at what extent these ontologies are actually reused in other ontologies, along with information of the domain of application, would help to guide the reuse of ontologies across domains of knowledge. In any case, an ontology selection should always start from clear ontological requirements related to the modelling problems and the domain being modelled, and these requirements are not necessarily compatible with popular or standard ontologies. Moreover, ontology selection criteria should take into account internationalisation [42] and licensing for reuse, as well as other possible statistics, that e.g. could show how ontologies are combined (by composition or merging), providing important information for their reuse.

2.1.2 Policies and implementation strategies for ontology reuse

International institutions, community consortia and standardisation bodies tend to encourage *direct* reuse of (their) existing standard or popular ontologies, also driven by social motivations. Instead, research communities may suggest *indir-*

ect approaches, i.e. drafting new ontologies tailored specifically to the collected requirements of the project, and aligning them to existing ones when considered appropriate, or *hybrid* approaches that mix direct and indirect approaches.

Therefore, we distinguish 3 policies for ontology reuse: direct reuse, indirect reuse, and hybrid reuse.

Direct reuse. This policy may be implemented in three ways: (i) importing the whole ontology to be reused (by means of the property `owl:imports` in the new ontology under development); (ii) embedding fragments (possible implementations of ontology design patterns) from a reused ontology in the local ontology; and (iii) embedding individual and selected ontology entities in the new ontology, possibly referencing the reused ontology by annotating their terms with the property `rdfs:isDefinedBy`. In the first case, the new ontology directly includes the semantics of the reused (via import) ontology, while in the last two cases the semantics of individual terms is delegated to the external ontology containing its formal definition. Directly reusing fragments of ontologies – as in (ii) – produces ontologies that can be seen as a composition of modules [70], intended as sets of terms and axioms that address a specific subset of requirements. This allows to clearly identify the specific *areas* that have been reused from source ontologies. Instead, reusing individual ontology terms – as in the third case – from different ontologies, leads to an ontology that does not allow to immediately understand which source ontology contributes, and how, to address a modelling issue (e.g. a competency question), as individual concepts, axioms, and statements, when merged into a new model [70], may loose or change their semantics in the context of the new ontology.

Indirect reuse. In this policy, terms and patterns from external ontologies are reused as templates in the new ontology. This means that ontology terms, and their semantics (i.e. their axioms), are reproduced in the local ontology, with possible changes or extensions. Thus, the local ontology is self-defined, and does not depend on external resources, but alignment axioms (such as `rdfs:subClassOf`, `owl:equivalentClass`, `owl:equivalentProperty`) are needed in order to support interoperability with other ontologies and make it explicit which parts have been reused.

Hybrid reuse is a reuse policy that combines direct and indirect reuse, guided by different motivations. These may be related to the characteristics of the reused ontologies, e.g. if they are considered reasonably stable and with a nearly absent or slow evolution, as in the case of important reference standards, or not. Or they may be motivated by the specific requirements of the ontology project, e.g. if an ontology is recommended as standard by designers involved in the development of the new ontology, or not.

Benefits and limits. On the one hand, `owl:imports` axiom is included in OWL, which provides a clear semantics, and is supported by most ontology development frameworks, such as Protégé. On the other hand, using an annotation property like `rdfs:isDefinedBy` allows for more flexibility, since it is possible to annotate the ontological entities that have been reused from an external ontology, rather than importing the whole ontology.

Directly reusing third parties ontologies can be convenient thanks to the possibility of delegating the issue of dealing with ontology preservation, versioning, storage and evolution. However, it generates a strong dependency on the reused vocabulary and its semantics: any (small) change in the reused ontologies, that is outside the control of the ontology designer reusing them[5], up to a reused ontology that is no longer available due to maintenance issues, could introduce inconsistencies in the local ontology and jeopardize its stability.

2.2 Pattern-based ontology design and reuse: real-world projects

As it can be noticed, reuse of existing ontologies is not in focus of most ontology engineering methodologies, but is often an activity that is performed ad-hoc and as a kind of *appendix* to the chosen methodology. However, whether and how to reuse

[5]Unless the reused ontologies are directly developed and maintained by the same team.

external ontologies is usually much related to the ontology design methodology. As for a pattern-based ontology engineering, it is not surprising that ontology design patterns play a role also in the reuse practice, and are recommended as a best practice for the design activity as a whole, including reusing existing resources. This is the case of the eXtreme Design (XD) methodology.

In Section 2.2.1 we describe the eXtreme Design methodology [74], with a special focus on the pattern-based ontology design and reuse, while in Sections 2.2.2 and 2.2.3 we present two ontology and knowledge graph projects we have been contributing to, showing how eXtreme Design can be applied to real-world scenarios, the usefulness of ODPs in ontology design and reuse, and specifically how with these two projects (that are both still live and ongoing) we enriched the methodology itself thanks to specific requirements and lessons learned.

2.2.1 eXtreme Design

eXtreme Design is an iterative, incremental and test-driven methodology, and puts the reuse of ontology design patterns at the core of its process, providing guidelines for such activity. Experiments and real use cases have proved the positive impact of ODPs on the ontology quality [12, 9, 80]. The intensive use of ODPs, modular design, and collaborative approach are the main characterising guidelines of the method [74].

When the project has started, XD is implemented by iterating a set of steps, each involving one or more teams: a *customer team*, that provides the requirements that guide the design and testing processes; a *design team*, which identifies and implements the most appropriate ODPs to the given requirements; a *testing team*, which takes care of testing the produced ontology components; an *integration team*, which is in charge of integrating the different components. In XD, the same requirements are used as input by the design team, for building the ontology, and by the testing team, for transforming them into unit tests.

Requirements engineering. The first step is the collection of requirements. XD suggests to collect them in the form of *user stories*, provided by the customer team, intended as sets of sentences that describe by example the kind of facts that the resulting knowledge graph should be able to encode.

Competency Questions. After possibly generalising the stories provided, by identifying the main concepts they exemplify, one or more competency questions (CQs) [45] are derived from each of them. CQs are the natural language counterpart of structured queries that the resulting knowledge graph should answer to. Besides deriving CQs, the design team works with the customer team for extracting from user stories possible general constraints, which express possible inferences or rules that apply to the concepts involved in the story.

Matching CQs to ODPs. CQs drive the selection of ODPs to reuse, so CQs are *matched* to ODPs. Possible existing solutions (ODPs in online catalogues or as fragments in existing ontologies) are analysed in order to find the most suitable ones to be reused in the local ontology. This is a complex cognitive task that, currently, lacks proper tool support. However, a published ODP that is thoroughly defined and documented should, in principle, also describe the modelling issue(s) that it addresses, along with the related competency questions. In this way, the ontology designers can assess whether an ODP matches (even at a different level of generalisation) the CQs that they have at hand by comparison.

Testing and integration. The unit testing approach followed by XD is described in [74, 13]. The testing team uses the CQs and the general constraints to design unit tests: CQs are translated into SPARQL queries, while general constraints are used as input to create sample triples that should provoke either consistency and coherence errors, or correct inferences. The results of the tests, when positive, are reported to the design team, which fixes the problems and resubmits to the testing team. Integration tests have a similar process, but focus on running regression tests, that is re-running all previously defined unit tests and additional tests on the whole ontology to check whether the ontology performs the same after an update.

2.2.2 ArCo: a pattern-based ontology network and knowledge graph on cultural heritage

2.2.2.1 An overview

ArCo project [20, 19] has as its main source of requirements the General Catalogue of Italian Cultural Heritage[6] (GC), which is the official institutional database of Italian cultural heritage (CH), maintained and published by ICCD-MiC (Italian Institute for Catalogue and Documentation - Italian Ministry of Culture). The catalogue contains hundreds of thousand catalogue records, which are XML files recording information about specific cultural properties. The maintenance of ArCo, which is an evolving project we contributed to, along with its produced resources, is guaranteed by MiC's commitment to evolve the resource, by following the XD methodology, with the support of researchers of the CNR (Italian National Research Council) and the University of Bologna.

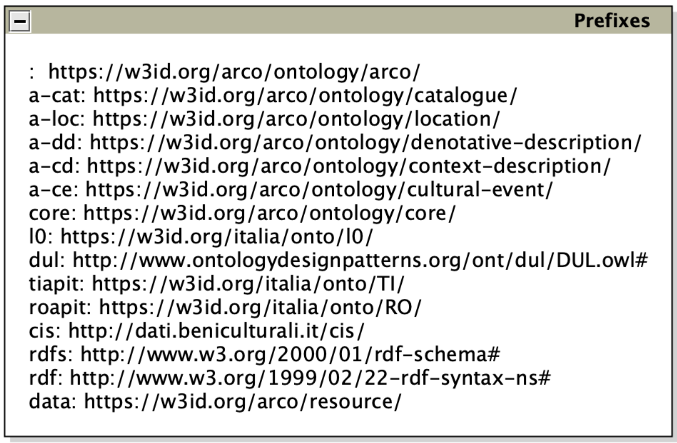
ArCo includes as main resources: (i) a knowledge graph consisting of a network of ontologies modelling the CH domain[7], and a LOD dataset with the data transformed from the available catalogue records; (ii) a software for the automatic conversion of catalogue records to ArCo-compliant LOD[8]; (iii) a detailed documentation reporting the ontological requirements (user stories and CQs), diagrams and examples of usage of the ontology, example SPARQL queries to be run on the KG[9]; (iv) a test suite that has been used for validating ArCo KG[10]; (v) a SPARQL endpoint[11].

2.2.2.2 Mapping competency questions to ODPs

In this subsection, we use some of the main modelling issues that have emerged from ArCo's requirements as driving examples to describe the process of matching

[6]https://catalogo.beniculturali.it/
[7]https://github.com/ICCD-MiBACT/ArCo/tree/master/ArCo-release/ontologie
[8]https://github.com/ICCD-MiBACT/ArCo/tree/master/ArCo-release/rdfizer
[9]https://dati.beniculturali.it/arco-rete-ontologie
[10]https://github.com/ICCD-MiBACT/ArCo/tree/master/ArCo-release/test
[11]https://dati.cultura.gov.it/sparql

requirements to Ontology Design Patterns (ODPs), as part of the XD methodology. In this way, that is by sharing its "behind the scenes", the intellectual and methodological processes that have been performed during the design and reuse of ontology design patterns, we aim at supporting the use of such ontology engineering methodology, and at showcasing by example a pattern-based ontology reuse. Figure 2.1 shows all the prefixes used in the next diagrams.


Prefixes

: https://w3id.org/arco/ontology/arco/
a-cat: https://w3id.org/arco/ontology/catalogue/
a-loc: https://w3id.org/arco/ontology/location/
a-dd: https://w3id.org/arco/ontology/denotative-description/
a-cd: https://w3id.org/arco/ontology/context-description/
a-ce: https://w3id.org/arco/ontology/cultural-event/
core: https://w3id.org/arco/ontology/core/
l0: https://w3id.org/italia/onto/l0/
dul: http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
tiapit: https://w3id.org/italia/onto/TI/
roapit: https://w3id.org/italia/onto/RO/
cis: http://dati.beniculturali.it/cis/
rdfs: http://www.w3.org/2000/01/rdf-schema#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
data: https://w3id.org/arco/resource/

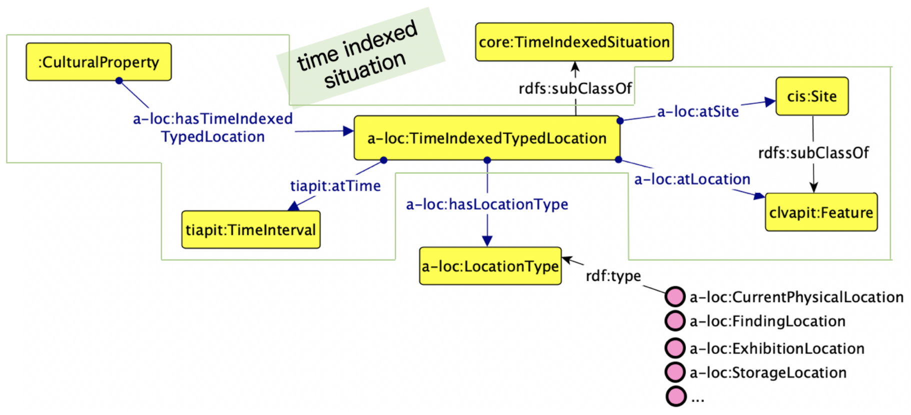
Figure 2.1: Prefixes used in the next figures.

Multiple time-indexed and typed locations for one cultural property. Dynamic concepts, such as situations that evolve over time, are present in almost every domain. Cultural property locations, which may change over time, are an exemplification of such dynamics in the cultural heritage domain. A tangible cultural property, i.e. a physical object, is placed in a physical location, which can be identified by e.g. its country, region, city, address. The location of an immovable cultural property (e.g. a monumental park) overlaps with the area occupied by it. Instead, the data about the address of a movable cultural property (e.g. a statue) refers to the building in which it is situated, and the possible related cultural institute. While an immovable cultural property, due to its nature, will be related to a unique geographical location during its life cycle, a movable cultural property can change multiple places, at different time intervals. Therefore, it is important to represent the situations in which a cultural property is located at some places, along with the temporal indexing of the locations associated with it. This requirement, if generalised, can be mapped to the existing *Time Indexed Situation*[12] ODP [39], which represents a situation that has an explicit time parameter, so this ODP has been reused and specialised for the specific use case, leading to the creation, inside the ArCo ontology network, of the *Time Indexed Typed Location* ODP. An additional requirement, not included in the more general pattern, is the need of modelling the *reason* why a cultural property is linked to a place: it can be the location where it was found/created, where an exhibition involving the cultural property has been held, where it was temporarily stored, etc. One location can play different *roles* with respect to different cultural properties, and this is why we want to specify this *role* in the time indexed situation. Specifically, as represented in Figure 2.2a, the core class of this pattern is `a-loc:TimeIndexedTypedLocation`, which is a a *situation* of a cultural property that is *located* in some place, at a certain *time* (`tiapit:atTime`), and where the location plays a specific *role* (`a-loc:LocationType`) in the situation (e.g. *finding location*, *exhibition location*, etc.). `a-loc:atLocation` represents the geographical entity involved in the situation, while its subproperty `a-loc:atSite` is used when this geographical entity corresponds to a physical building that hosts cultural properties, and is linked to at least one cultural institute. For instance, the Pitti Palace is the `cis:Site` of multiple cultural institutes, including the "Palatine Galleries" and the "Museum of Custom and Fashion", and hosts many cultural properties.

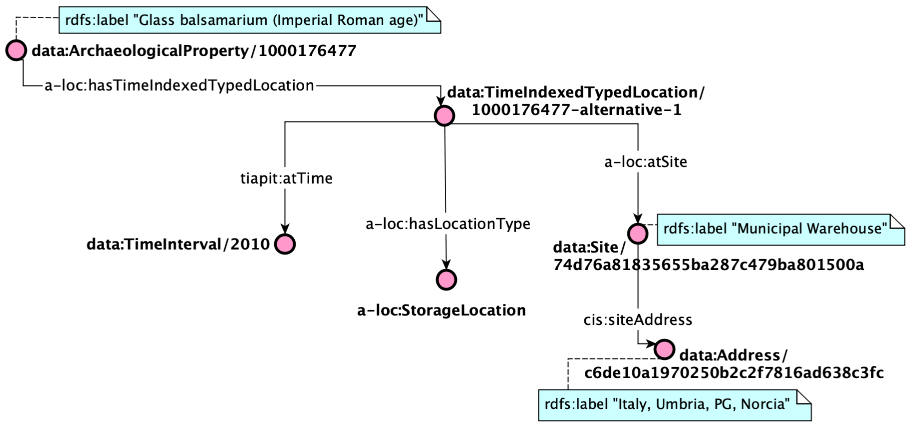
Figure 2.2b depicts an example: the place where a *balsarium* glass from the Imperial Roman age[13] was temporarily stored, i.e. one of its time-indexed typed locations. Indeed, the location type of `data:TimeIndexedTypedLocation/10001 76477-alternative-1` is `a-loc:StorageLocation`. From the other pieces of information about the time interval and cultural site, we can assert that this archaeological property has been stored in a Municipal Warehouse in the city of Norcia (Italy) in 2010.

[12]http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl

[13]https://w3id.org/arco/resource/ArchaeologicalProperty/1000176477



(a) The model for time indexed typed locations.


rdfs:label "Glass balsamarium (Imperial Roman age)"
data:ArchaeologicalProperty/1000176477
a-loc:hasTimeIndexedTypedLocation
data:TimeIndexedTypedLocation/ 1000176477-alternative-1
tiapit:atTime
a-loc:atSite
a-loc:hasLocationType
data:TimeInterval/2010
rdfs:label "Municipal Warehouse"
data:Site/ 74d76a81835655ba287c479ba801500a
a-loc:StorageLocation
cis:siteAddress
data:Address/ c6de10a1970250b2c2f7816ad638c3fc
rdfs:label "Italy, Umbria, PG, Norcia"

(b) An instance of the model in Figure 2.2a, with the site where an archaeological property has been temporarily stored in 2010.

Figure 2.2: Time indexed situation ODP reused for modelling different types of locations of a cultural property.

Situations and their descriptions. Besides its locations, a cultural property can be involved in many other different situations during its life cycle: it can be commissioned, bought, obtained, used (e.g. a garment that is wore by one person for a particular event), it can be part of a collection, or a numismatic series, can change its availability as a result of theft, destruction or rescue, etc. Each one of these situations defines a contextual relation between the cultural property and other entities involved in the situation. The *Situation*[14] ODP [39] is relevant in this

[14]http://www.ontologydesignpatterns.org/cp/owl/situation.owl

case because it models the concept of a contextual $\geq$ 2-ary (usually called n-ary) relation.

When a coin is issued, there are many entities that play a role: the coin itself, the issuer, the issuing State, the mint and the minter. The *coin issuance* is a situation representing the relation between all these entities, for that purpose. Figure 2.3 shows how we model the `a-cd:CoinIssuance` by specialising the *Situation* ODP, representing all the entities involved.

Another important situation in which a cultural property can be involved is the authorship attribution, a specific type of `a-cd:Interpretation`, that is a situation in which data about a cultural property are processed by an agent, generating explicit knowledge, thanks to specific sources or criteria. An `a-cd:AuthorshipAttribution` (see Figure 2.4) is a situation in which one author is attributed to a cultural property, based on an `a-cd:InterpretationCriterion`, e.g. an inscription, a bibliography, a documentation that motivates this attribution. Each cultural property has at least one preferred (more accredited) authorship attribution and/or a cultural scope attribution (e.g. Swedish workshop), and can have one or more alternative (previous and obsolete) authorship attributions.

Let us take as an example a coin[15] (Figure 2.5) from the 20th century, issued by Victor Emmanuel III of Italy: the relation of the coin with the issuer is expressed as both an *n-ary* relation (`a-cd:AgentRole`), involving the King and the role he played as issuer, and as a *shortcut* object property (`a-cd:hasIssuer`). Moreover, this coin has a preferred authorship attribution `data:PreferredAuthorshipAttribution/-1400019640-1` with Calandra Davide as attributed author; this attribution is based on a 'stylistic analysis' (`data:InterpretationCriterion/analisi-stilistica`). The object property `a-cd:hasAuthor` directly relates the cultural property to the preferred author(s).

Recurrence in cultural events and intangible cultural heritage. Let us think about different editions of an annual painting award, as the repetition of a cultural event: usually, we informally use the term *recurrent event* as we are referring

[15]https://w3id.org/arco/resource/NumismaticProperty/1400019640

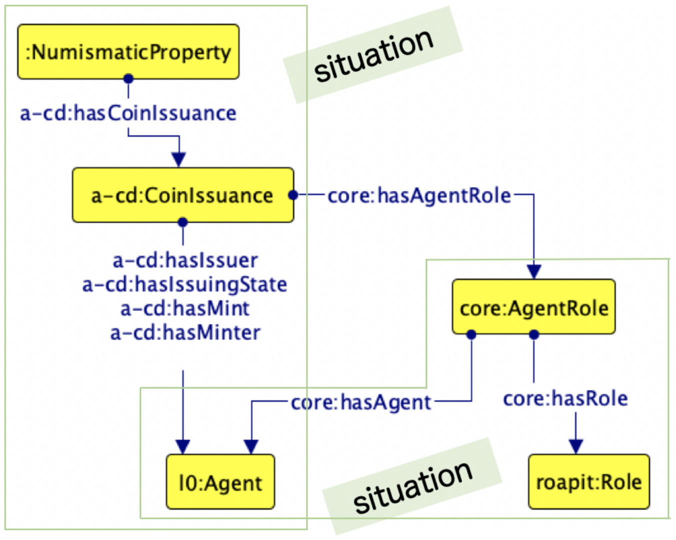


Figure 2.3: Situation ODP reused for representing the coin issuance.

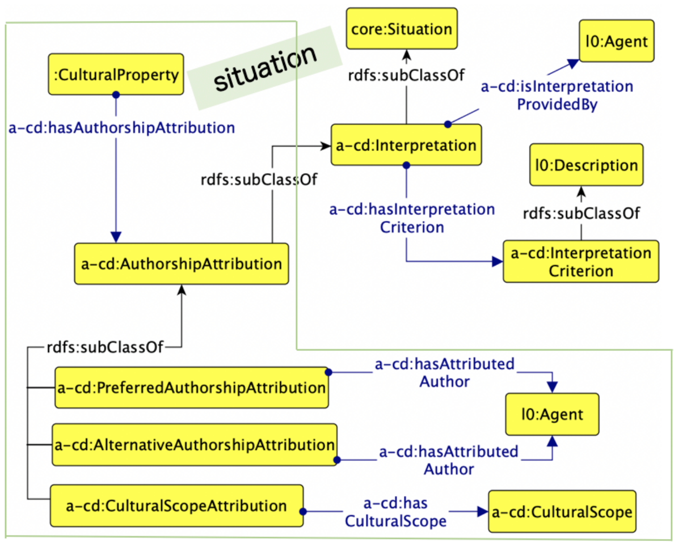

core:Situation
l0:Agent
:CulturalProperty
situation
rdfs:subClassOf
a-cd:isInterpretation ProvidedBy
a-cd:hasAuthorshipAttribution
a-cd:Interpretation
l0:Description
rdfs:subClassOf
a-cd:hasInterpretation Criterion
rdfs:subClassOf
a-cd:AuthorshipAttribution
a-cd:Interpretation Criterion
rdfs:subClassOf
a-cd:PreferredAuthorshipAttribution
a-cd:hasAttributed Author
l0:Agent
a-cd:AlternativeAuthorshipAttribution
a-cd:hasAttributed Author
a-cd:CulturalScopeAttribution
a-cd:has CulturalScope
a-cd:CulturalScope

Figure 2.4: Situation ODP reused for the authorship attribution.

to an event that occurs more than once. Instead, we are implicitly talking about a *series* of conceptually unified situations: for example, the Art Biennale[16] is a series of consecutive situations that can be somehow considered as part of the same collection of situations. If we consider that a traditional ceremony related to the *year*

[16]https://www.labiennale.org/en/art/

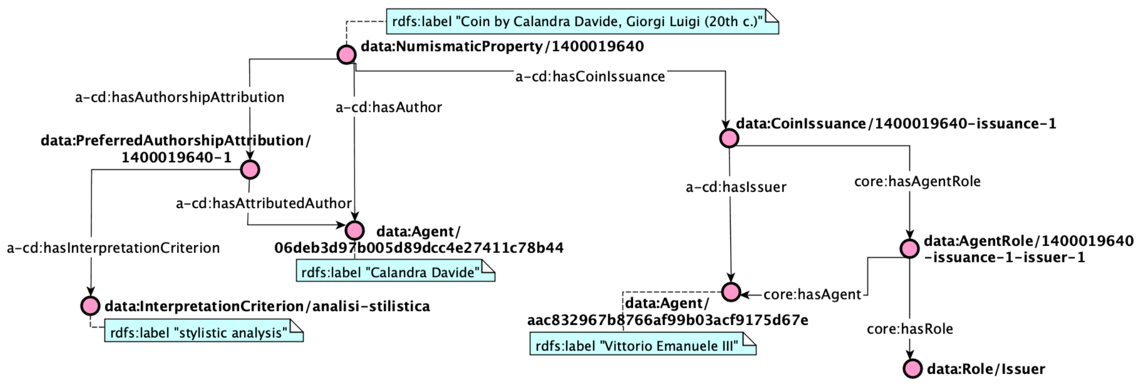

rdfs:label "Coin by Calandra Davide, Giorgi Luigi (20th c.)"
data:NumismaticProperty/1400019640
a-cd:hasAuthorshipAttribution
a-cd:hasAuthor
a-cd:hasCoinIssuance
data:PreferredAuthorshipAttribution/1400019640-1
data:CoinIssuance/1400019640-issuance-1
a-cd:hasIssuer
core:hasAgentRole
a-cd:hasAttributedAuthor
data:Agent/06deb3d97b005d89dcc4e27411c78b44
rdfs:label "Calandra Davide"
a-cd:hasInterpretationCriterion
data:AgentRole/1400019640-issuance-1-issuer-1
core:hasAgent
data:Agent/aac832967b8766af99b03acf9175d67e
rdfs:label "Vittorio Emanuele III"
data:InterpretationCriterion/analisi-stilistica
rdfs:label "stylistic analysis"
core:hasRole
data:Role/Issuer

Figure 2.5: An instance of the model in Figures 2.3 and 2.4: a numismatic property involved in a coin issuance and a preferred authorship attribution.

cycle (e.g. Carnival) can be considered a cultural property itself (an intangible one), we can say that even cultural properties can be recurrent, when they have regular time intervals between their repetitive occurrences. In their iteration, there needs to be a recognisable pattern: e.g. an exhibition that has different editions over years usually follows a pattern in scheduling consecutive editions at regular time intervals (e.g. one edition every two years). Moreover, there are specific attributes that give all different occurrences a unity: for the *Art Biennale*, these include a general topic that unites them and does not change i.e. contemporary art, and a place that always hosts the situations i.e. Venice, etc.

Recurrent situations are usually modelled as a special type of events (e.g. in Wikidata[17]), while the fact that they are multiple situations belonging to a series, and the nature of such a unifying entity, is neglected or confused with the concept of event in the literature (see for instance the DBpedia resource for Venice Biennale[18]). In order to be fully expressive, we need to model both the unitary series of situations, e.g. the *Art Biennale*[16] intended as something that occurs biennially under certain conditions, and its individual member situations, e.g. the *Art Biennale 2019* intended as a particular edition of the series that has a start date and an end date. This requirement brought us to the development of a new ODP for modelling

[17]https://www.wikidata.org/wiki/Q15275719
[18]http://dbpedia.org/page/Venice_Biennale

Recurrent situation series[19] [21, 22]. When modelling this pattern, other 5 patterns have been reused and specialised: Classification, Collection, Description, Sequence, Situation.

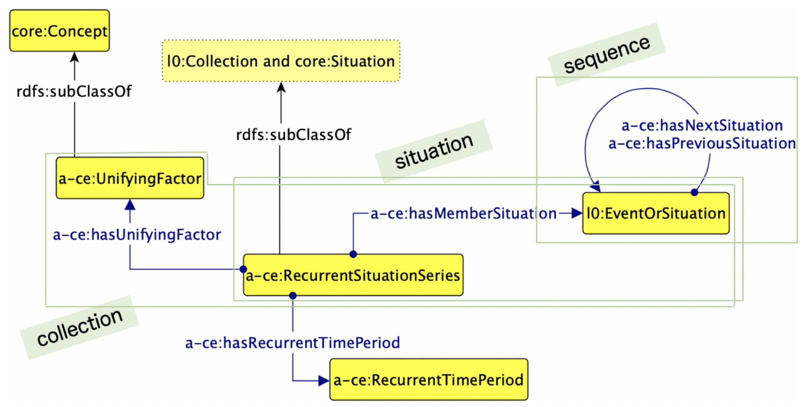
Indeed, a recurrent situation series (see Figure 2.6a) is represented as a collection, whose members are the multiple situations (`a-ce:hasMemberSituation`). Member situations need to share at least one common property (e.g. the topic, the organiser), so that they are conceptually unified by what we call *unifying factors* (`a-ce:has-UnifyingFactor`) of the series. At the same time, a recurrent situation series is also represented as a situation, since it represents the relational context of all the member situations. Each member situation is related to its own time interval and is part of a *sequence* that links it to the other member situations of the same series (e.g. `a-ce:hasPreviousSituation`). The time period between two consecutive member situations is (approximately) regular and is represented as a distinctive attribute of the series (`a-ce:hasRecurrentTimePeriod`).

An instance of this pattern, depicted in Figure 2.6b, is the `ex:ArtBiennale` (*Biennale d'arte di Venezia*), which is a contemporary visual art exhibition, whose member situations are conceptually unified by: the promotion of new contemporary art trends as mission, the Biennale Foundation as organiser, Venice as place. The time period between two consecutive situations of the Art Biennale series is approximately `ex:2Years`. All situations member of the series are in a temporal sequence: for example, the immediate next situation of `ex:ArtBiennale1895` is `ex:ArtBiennale1897`, while the object property `a-cd:hasImmediatePrevious Situation` relates the 1899 edition to the one held in 1897.

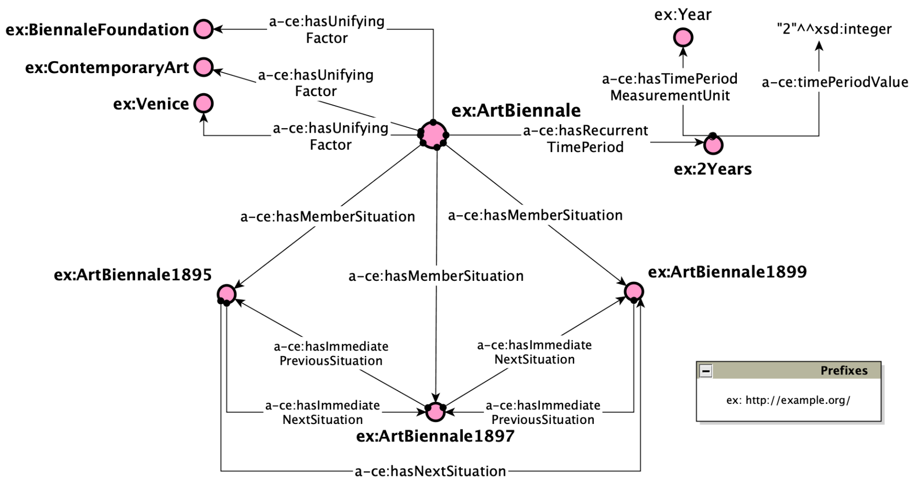
2.2.2.3 Advancements and support to ontology reuse

Requirements from an evolving heterogeneous community. The main *customer* of the project is ICCD, i.e. the institute that collects and preserves the data of the General Catalogue, and takes care of releasing and updating the cataloguing standards. ICCD domain experts composed the customer team and guided the

[19] http://www.ontologydesignpatterns.org/cp/owl/recurrentsituationseries.owl



(a) The model for recurrent situation series.



(b) An instance of the model in Figure 2.6a, with the 3 first situations member of the Art Biennale.

Figure 2.6: The new pattern Recurrent Situation Series as implemented in ArCo.

design team in selecting and prioritising the requirements. They also supported the design team in the understanding of the cataloguing standards. ArCo ontologies reflect – and are compatible with – ICCD standards. However, with ArCo project, ICCD wanted to address the needs of a diverse community of potential stakeholders and consumers of its data, while managing the development process, thus ArCo ontologies are not exclusively modelling their interpretation of the cultural heritage domain, which would limit them to the point of view of ICCD cataloguing practices. For this reason, the collection of requirements has been handled as an open process by directly involving external actors, and in particular an "Early Adoption Program" has been launched, with the aim of engaging a group of representatives of potential consumers, who would provide both requirements and validation. This approach led to the building of an open community as well as widening the scope of the project requirements. The members of the Early Adoption Program (Early Adopters) are provided with assistance and support, and are guaranteed that the fulfillment of their issues and requirements is given a high priority. Many different means of interaction have been used, in order to guarantee a lively interaction within this project community. First of all, regular meetings, such as webinars and in-person meetups, have been held, and an open mailing list supported the discussion of specific issues. Moreover, GitHub issues[20] can be submitted for proposing improvements and reporting bugs. Thanks to this approach, the customer team became an evolving creature, which can extend over time, involving additional representatives of potential producers and consumers of CH data. During its first main phase, this ArCo community, actively providing requirements, involved private companies, public administrations, researchers and creative developers. Following XD guidelines, these requirements are collected in the form of small stories. A story is a narrative text that exemplifies a possible scenario or reports an actual use case. A Google Form supported the submission of such stories by the customer team, and the maximum number of characters for the text was 250[21]. User stories are also useful for defining the ontology vocabulary. The submission form requires that each story is associated with one of three proposed categories, that indicate the type of project motivating the story, namely: (i) publishing cultural heritage data as a knowledge graph, (ii) linking an existing CH KG to ArCo KG, (iii) feeding some applications with ArCo data or providing services based on ArCo KG. Additional stories and materials (such as a sample of the original data) can be uploaded.

Based on the methodology, the requirements that can be extracted from these user stories, as well those extracted from ICCD standards, are translated into Competency Questions.

As an example, we report one of the stories collected through the Google form

[20]https://github.com/ICCD-MiBACT/ArCo/issues
[21]https://goo.gl/forms/zCixt3B1ABYbj9JS2

in the first iteration with the customer team:

Type: Linking my data to ArCo data

Title: Cultural heritage and residential property

Story: I am looking for a residential property to buy, and I want to filter the results based on the type of cultural heritage nearby.

From this story the following CQs can be extracted: "What are the types of cultural properties located in a certain area?", "Which is the current location of a cultural property?", "Which are the geographic coordinates of the current location of a cultural property?", "Which is the type of a cultural property?". Other user stories were focused on linking cultural properties to multimedia resources (e.g. photographic documentation); modelling specific properties of music heritage; reporting the changes over time of the availability of cultural properties that have been confiscated from organised crime; linking catalogue records to the related heritage protection agencies.

Having a clear strategy for collecting such stories, and interacting with the customer team with the appropriate tools is a precious activity for starting with clear, well defined and representative stories. This is one of the prerequisites for performing an effective mapping of the extracted CQs to the issues possibly addressed by existing ODPs and existing ontologies, i.e. for implementing ontology reuse activities.

An architectural pattern for large ontology networks. Managing big ontologies is a non-trivial activity for ontology designers, reasoners and users. A modular design, as opposed to a monolithic one, i.e. one ontology module addressing all requirements, has proved to favour readability, reusability and maintainability of an ontology [84, 67]. Ontologies are usually split in modules based on conceptually coherent sub-parts of the domain. However, eXtreme Design does not contain explicit guidelines on how to approach a modular and networked design of potentially large ontologies. Starting from our experience in developing the ArCo ontology network, we provide some guidelines, and propose an architectural pattern, that can be applied in other contexts with similar characteristics as the ArCo's project.

The ArCo ontology network follows what we name a *root-thematic-foundations* architectural pattern. It can be described as follows:

- a root module represents the *entry point* of the network: it imports all main *thematic modules*, thus causing the whole network to be loaded. In ArCo this is named the *arco* module, and, besides importing the network, it also models the ontology top-level hierarchy of classes, with `:CulturalProperty` as its root class.

- the second layer of the network consists of the main *thematic modules*, the ones that are directly imported by the root module. These thematic modules may import additional, secondary thematic modules, that depend on them (e.g. they are a specialisation of them) and may form additional layers in the network. The version 1.0 of the ArCo ontology network includes one layer of thematic modules: *denotative-description*, *context-description*, *cultural-event*, *location* and *catalogue*.

- a leaf module contains foundational, and not domain-specific, concepts, such as agent, physical object, role, part-whole, etc. This module needs to be imported by all main thematic modules. The name of this module in ArCo is *core*.

The implementation of the *root-thematic-foundations* pattern needs to follow a preliminary and clear conceptual organisation of the domain into separate coherent subdomains, that is a kind of clustering of the requirements based on thematic areas. Of course, there are not general criteria to be followed for identifying thematic areas and defining their granularity: these can vary depending on the project scope, design choices and the size of the domain. New thematic modules can be added over time, when a set of new requirements make up new subdomains.

In the context of ArCo, from the very beginning of the project, we analysed the ICCD General Catalogue data, along with their standards, and the user stories provided by the customer team at a very early stage of development. We initially focused on concepts and relations relevant to all types of cultural properties: this activity gave us an overview of the CH domain, without diving into concepts specific to particular types of cultural properties. We manually clustered these attributes, which led us to identify five topics that would then drive the modelling of the 5 main thematic modules of the ArCo ontology network.

The General Catalogue of Italian CH contains both data directly describing a cultural property and its contextual information (e.g. techniques, materials, surveys, exhibitions), data about the records that catalogue the cultural properties (e.g. the date of creation of the catalogue record, their version, etc.); data about other entities related to cultural properties (e.g. documents, inventories, bibliography).

Based on this analysis, one thematic module of the network (*catalogue*[22]) has been dedicated to the ICCD General Catalogue (GC), and in particular to its catalogue records and their properties. Cultural properties can be described by means of both measurable, intrinsic aspects such as weight, height, material(s), state of conservation, as well as properties that result from an interpretation activity, such as the attribution of one (or more than one) author, the definition of the date of creation, etc. This distinction led us to the definition of two additional thematic modules, respectively: *denotative description*[23], and *context description*[24]. Finally, other two thematic modules are dedicated to two major components of the life cycle of a cultural property: (i) the different locations associated with it (e.g. the current location, the place where it was found, where it was exhibited, where it was created, temporarily stored, etc.), and information about the physical sites (e.g. a building), geometry and related cadastral entities, and (ii) the cultural events in which it takes part, including events that recur over time, such as festivals and festivities. These modules are the *location*[25] module and the *cultural event*[26] module.

Finally, as in the architectural pattern, the *core*[27] module captures foundational

[22]`a-cat:` https://w3id.org/arco/ontology/catalogue/

[23]`a-dd:` https://w3id.org/arco/ontology/denotative-description/

[24]`a-cd:` https://w3id.org/arco/ontology/context-description/

[25]`a-loc:` https://w3id.org/arco/ontology/location/

[26]`a-ce:` https://w3id.org/arco/ontology/cultural-event/

[27]`core:` https://w3id.org/arco/ontology/core/

concepts, and is reused (by direct import) in all thematic modules, and the *arco*[28] module is the entry point of the network, importing all thematic modules, and defining the top-level hierarchy of CH concepts in ArCo.

This kind of design, i.e. splitting a big set of requirements into clusters, which will then correspond to separate interconnected modules of an ontology network, can support ontology reuse. Indeed, each module can be explored and understood more easily thanks to its more limited size, and it is made explicit which (sub)topic and subset of requirements that module addresses. For instance, a user interested in (cultural property) locations, for direct or indirect reuse, will have the opportunity to avoid the exploration of the whole network. She can limit her exploration to the dedicated module. Of course, the creation of an ontology network with a clear architecture, following specific criteria/patterns (as the architectural pattern we propose), will make the network itself more readable and reusable.

Indirect reuse of existing ontologies. During the process of incrementally selecting the CQs representing ArCo's requirements, and then matching them with one or more existing ODPs, we also inspected state-of-the-art relevant ontologies, such as CIDOC CRM[29], EDM[30], BIBFRAME[31], FRBR[32], etc, in order to find possible reusable fragments. In any case, this reuse and matching activity was incremental and manual, and the most significant effort to look for reusable ontology fragments involved large ontologies such as CIDOC CRM. This experimented issue represents one of the motivations to the development of the method in Chapter 3. Such method aims at helping to make the inspection of ontologies clearer and more understandable by extracting ontology design patterns, and grouping them based on the modelling issues they address, hence easing ontology reuse, and may possibly contribute to supporting automatic matching procedures.

[28]: https://w3id.org/arco/ontology/arco/
[29]https://www.cidoc-crm.org/
[30]https://pro.europeana.eu/page/edm-documentation
[31]http://id.loc.gov/ontologies/bibframe.html
[32]http://vocab.org/frbr/core

2.2.3 Polifonia: a pattern-based ontology network and knowledge graph on musical heritage

2.2.3.1 An overview

Polifonia[33] is a project funded by the EU Horizon 2020 Programme, that will run from January 2021 until April 2024. Its main result is the ongoing development of a digital ecosystem for European Musical Heritage, intended as music objects, their cultural and historical contexts, possibly expressed in different languages and styles, and across centuries. The ecosystem includes both methods, tools, resources, guidelines, and creative designs. The development of the project is driven by ten pilots, that provide requirements to the project and validate its developed technologies. Knowledge graphs, and ontologies, are the *background* technologies that have been chosen for integrating, representing, and interlinking music-related data with heterogeneous and distributed provenance. We are contributing to this project by developing, within the ontology design team, an ontology network that, based on the collected requirements, aims to cover a wide variety of musical aspects (musical features, instruments, emotions, performances) [17]. This ontology network is also developed following the eXtreme Design methodology principles, and adopts the same architectural pattern as implemented in ArCo, and presented in the previous Section.

2.2.3.2 Mapping competency questions to ODPs

In this subsection, we will use two modelling issues that have emerged from one of the pilots of Polifonia, namely MusicBO, as examples to show how, in Polifonia as in ArCo, we matched the collected requirements to Ontology Design Patterns (ODPs), following the XD methodology.

MusicBO[34] has the objective to support the enhancement and dissemination of the musical heritage of the Italian city of Bologna, by publishing easily consumable

[33]https://polifonia-project.eu/

[34]https://polifonia-project.eu/pilots/musicbo

data that documents its historical role as a creative city for music. The main output of MusicBO are multilingual (English, French, Spanish and Italian) digital textual corpora, containing documents (letters, reports, news, etc.) that provide an evidence of scholars, journalists, travelers, writers and students related to the European musical landscape from medieval to modern times. The resulting MusicBO corpus undergoes a process that transforms it automatically into a knowledge graph. First, sentences from the corpus are parsed into semantic graphs by relying on Abstract Meaning Representation (AMR)[35] [6]. Then, the AMR2Fred tool [63] transforms AMR graphs into RDF/OWL KGs based on FRED [40], having as a result a knowledge graph whose design is based on frame semantics and ontology design patterns. A frame is usually expressed by verbs, so all occurrences of frames from an input text are formalised as OWL n-ary relations, all being instances of some type of event or situation [40].

The domain experts involved in MusicBO defined some useful stories in order to provide requirements. Some MusicBO stories are also data-driven, i.e. they have been extracted by looking at the resulting knowledge graph, e.g. by retrieving and analysing the most instantiated frames in the MusicBO KG, and then modelling an appropriate ontology design pattern to be included in the Polifonia ontology network, and aligning it to the original frame.

(Nick)names with temporal and contextual validity. An interesting frame instantiated in the MusicBO KG is the *name.01* frame[36]. From the comment of the frame: 'this frame contains words that talk about how Speakers name Entities'. The main roles included in this frame are: (i) *named*, (ii) *name_of_arg1*, and (iii) *time*. Indeed, this frame represents a situation in which an entity (*named*) is given a name (*name_of_arg1*), and this name may have a temporal validity (*time*). For example, a person may decide to change her name over time, or may be given a name that is valid within a certain organisation she is member of for a cer-

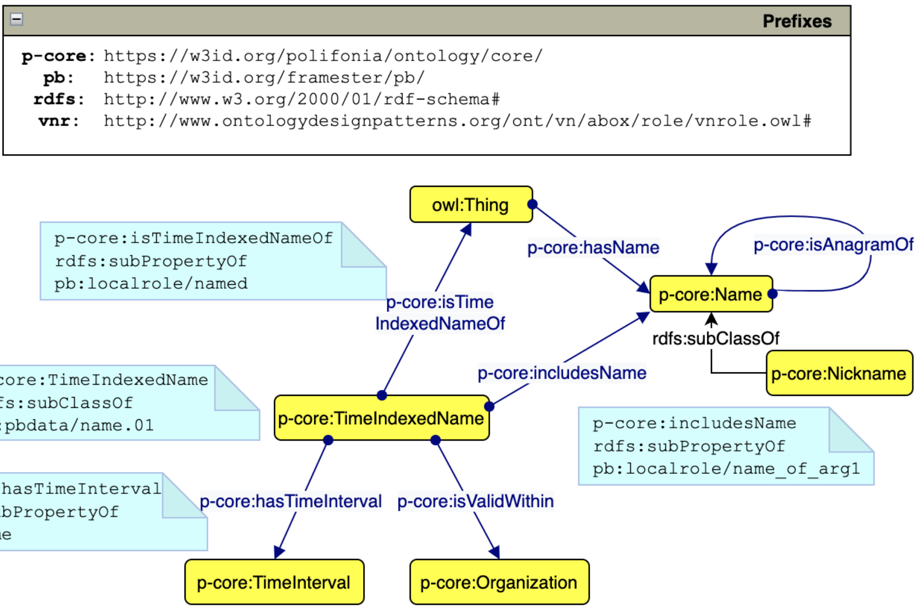
[35]This pipeline has been described in the deliverable D4.5 of the Polifonia project [88]

[36]See https://w3id.org/framester/framenet/abox/frame/Name_conferral, aligned with https://w3id.org/framester/pb/pbdata/name.01

tain period of time, or a street may have multiple denominations over time, etc. Starting from this requirement, and based on this frame, we modelled in the Polifonia ontology network the *Time Indexed Name* ODP, which is a specialisation of the more general ODP *Time Indexed Situation*[37]. Specifically, as represented in Figure 2.7a, the main class of the pattern is `p-core:TimeIndexedName`, that is a situation involving an entity (`p-core:isTimeIndexedNameOf owl:Thing`) that has a name (`p-core:includesName p-core:Name`) for a certain period of time (`p-core:hasTimeInterval`). The properties and classes of this pattern, when possible, have been aligned with the frame *name.01* and its roles (see the blue rectangles in the Figure). An additional requirement, that emerged from the examples in the MusicBO corpus, consists in modelling the organisation within which a certain name, given to an entity, may be valid: we modelled a relation `p-core:isValidWithin` with `p-core:Organization` as range. Moreover, the class `p-core:Name` has been specialised by the subclass `p-core:Nickname` for specific use cases. Finally, a name can be an anagram of another name: this has been modelled with the relation `p-core:isAnagramOf`. Figure 2.7b depicts an example of instantiation of this pattern, based on an actual sentence extracted from the corpus: 'In 1615 he [Adriano Banchieri] helped to found the Accademia dei Floridi, the first such society in Bologna; his name in it was *Il Dissonante*'. In this sentence, we have a person (`ex:Agent/AdrianoBanchieri`), which had a (nick)name (`ex:Nickname/TheDissonant`, from the Italian *Il Dissonante*) that was given to him inside an organisation, i.e. an academia (`ex:Organization/AccademiaDeiFloridi`). It is not mentioned in the sentence that Adriano Banchieri was given this name starting from 1623, while we could not find any information about until when this name was considered still valid, therefore in the example we just added `ex:TimeInterval/1623-unknown` as time interval.

Time-indexed art creation. Another frame frequently occurring in the MusicBO KG is *compose.02*, i.e. 'to create, produce art'. This frame includes the roles: (i) *entity_composed*, (ii) *artist*, (iii) *beneficiary*, (iv) *time*, and (v) *location.*

[37]http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl

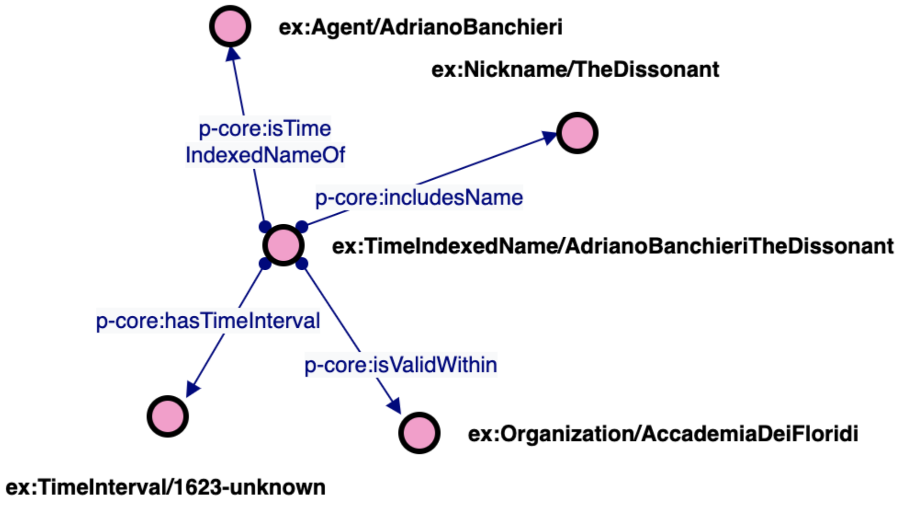


Prefixes

p-core:	https://w3id.org/polifonia/ontology/core/
pb:	https://w3id.org/framester/pb/
rdfs:	http://www.w3.org/2000/01/rdf-schema#
vnr:	http://www.ontologydesignpatterns.org/ont/vn/abox/role/vnrole.owl#

owl:Thing
p-core:hasName
p-core:isAnagramOf
p-core:Name
rdfs:subClassOf
p-core:Nickname
p-core:isTimeIndexedNameOf rdfs:subPropertyOf pb:localrole/named
p-core:isTime IndexedNameOf
p-core:includesName
p-core:TimeIndexedName rdfs:subClassOf pb:pbdata/name.01
p-core:TimeIndexedName
p-core:includesName rdfs:subPropertyOf pb:localrole/name_of_arg1
p-core:hasTimeInterval rdfs:subPropertyOf vnr:Time
p-core:hasTimeInterval
p-core:isValidWithin
p-core:TimeInterval
p-core:Organization

(a) The model for time indexed names.



ex:Agent/AdrianoBanchieri
ex:Nickname/TheDissonant
p-core:isTime IndexedNameOf
p-core:includesName
ex:TimeIndexedName/AdrianoBanchieriTheDissonant
p-core:hasTimeInterval
p-core:isValidWithin
ex:Organization/AccademiaDeiFloridi
ex:TimeInterval/1623-unknown

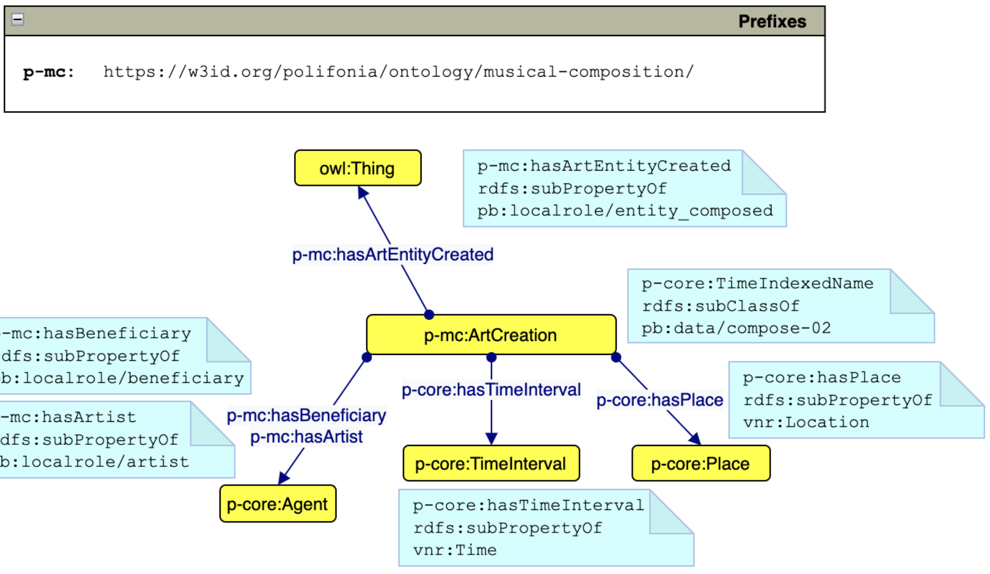
(b) An instance of the model in Figure 2.7a, with the nickname of Adriano Banchieri inside the Accademia dei Floridi.

Figure 2.7: Time indexed situation ODP reused for modelling time indexed names of entities.

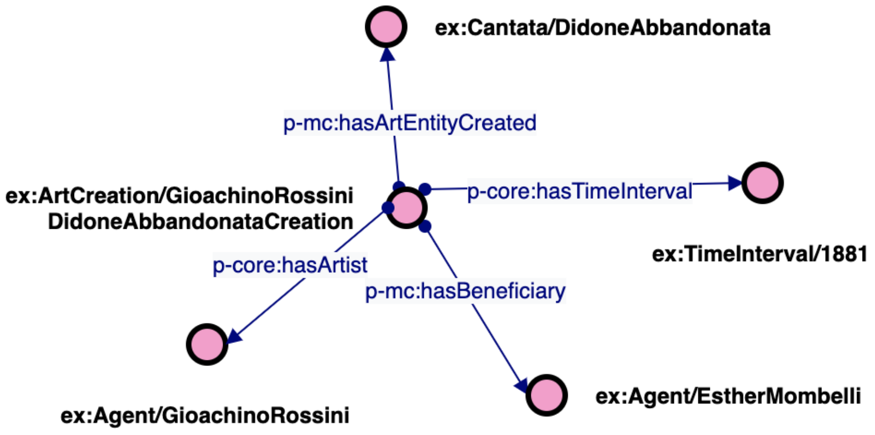
This frame represents the process of an artist that creates a piece of art, with a possible agent (person, organisation) that benefits from it, i.e. it is a beneficiary, and this situation happens at a certain time and at a certain place. This requirement led to the modelling of the *Art Creation* ODP, which is also a specialisation of the *Time Indexed Situation* ODP. The core class of the pattern (see Figure 2.8a) is `p-mc:ArtCreation`, that is a situation in which an agent (`p-mc:hasArtist p-core:Agent`) creates an entity (`p-mc:hasArtEntityCreated owl:Thing`), at some place (`p-core:Place`) and time (`p-core:TimeInterval`). This *art creation* can be linked to a beneficiary with the property `p-mc:hasBeneficiary`. When applicable, the ontological entities of the pattern are aligned with the frame *compose.02* and its roles (see the blue rectangles in the Figure). Figure 2.8b shows an example instance of the pattern in 2.8a, derived from the corpus' sentence 'the *cantata* of *Didone Abbandonata* which Rossini composed for a relation, the afterwards celebrated Esther Mombelli, in 1811'. `ex:Agent/GioachinoRossini` is the agent that, in the context of the situation `ex:ArtCreation/GioachinoRossini DidoneAbbandonataCreation`, is creating the *cantata* titled 'Didone Abbandonata'. The creation happened in 1881 (`ex:TimeInterval/1881`), the art entity created (`ex:Cantata/DidoneAbbandonata`) is dedicated to `ex:Agent/EstherMombelli`.

2.2.3.3 Advancements and support to ontology reuse

Requirements from a big heterogeneous community, and for developing user interfaces. The digital ecosystem for European Musical Heritage implemented by Polifonia has many different outputs: along with ontologies and knowledge graphs, relevant resources include also highly interactive (visual) user interface tools, to allow consumers of musical content to browse, access and explore the resources developed by the project, and interfaces that will exploit gestures and haptic sensation for supporting musical exploration and engagement. These tools will also rely on developed ontologies and knowledge graphs, and all pilots will have as output both ontologies and knowledge graphs, and interaction components. For this reason, it was needed to look for a methodological convergence between the ontology engineering and the interaction design tasks. We contributed to the definition of an integrated approach for collecting requirements for both ontology design activities, and interaction components, by collaborating with the user interaction team. As


Prefixes
p-mc: https://w3id.org/polifonia/ontology/musical-composition/
owl:Thing
p-mc:hasArtEntityCreated rdfs:subPropertyOf pb:localrole/entity_composed
p-mc:hasArtEntityCreated
p-core:TimeIndexedName rdfs:subClassOf pb:data/compose-02
p-mc:hasBeneficiary rdfs:subPropertyOf pb:localrole/beneficiary
p-mc:ArtCreation
p-core:hasTimeInterval
p-core:hasPlace
p-core:hasPlace rdfs:subPropertyOf vnr:Location
p-mc:hasArtist rdfs:subPropertyOf pb:localrole/artist
p-mc:hasBeneficiary
p-mc:hasArtist
p-core:TimeInterval
p-core:Place
p-core:Agent
p-core:hasTimeInterval rdfs:subPropertyOf vnr:Time

(a) The model for art creations.



(b) An instance of the model in Figure 2.8a, with the creation of a cantata dedicated by Gioachino Rossini to Esther Mombelli.

Figure 2.8: Time indexed situation ODP reused for modelling the creation of pieces of art.

in eXtreme Design, the collection of requirements is still based on the creation of *stories*, from which Competency Questions can be derived. However, the *Persona* and *Scenario* have been included in the process, *borrowed* by User eXperience (UX) Design practices. UX Design focuses on the design of the interface and of the human experience in which that interface can play a role. This is why UX developers need to focus on a wide range of characteristics of the people for whom the design is being developed (such as their interests, background) and the relevant contexts into which the design will be situated. *Personas* are basically descriptions of typical users (including their name, age, possible occupation). They can evolve during the design process. A *Scenario* is a description of how the Persona's main task or problem is solved before, during and after the interaction with the resource/software developed. The Scenario gives some insight into the Persona's activities and goals. Each Persona may be associated with multiple stories, intended as in eXtreme Design methodology, and examples of data to be modelled in the ontology. This clear template, besides supporting the collection of requirements for both ontology and interaction design, eases the creation of useful documentation for supporting the reuse of the project's resources.

Guidelines for publishing and documenting ontologies and knowledge graphs. As Polifonia involves a high number of different partners, and has a big and evolving network of ontologies and knowledge graphs as output (in addition to other types of resources and softwares), it was crucial to define clear guidelines for publishing such resources. A publication process that is clear and consistent over the project, and a detailed and complete documentation that follows the same structure and includes the same pieces of information per type of resource, contribute to support not only the management of the resources developed inside Polifonia, but also the reuse of such resources by users both internal and external to the project. Polifonia widely uses GitHub[38] as a code hosting platform for version control and collaboration, and defined a *rulebook*[39] in order to provide guidelines and recommendations on how to contribute to the Polifonia Ecosystem. In particular, we contributed to define rules and guidelines for the publication and documentation of ontologies and knowledge graphs developed by Polifonia ontology designers. More guidelines will be added to the current list, as new possible requirements emerge.

Guidelines for ontologies. These guidelines are about both the development of

[38]https://github.com/polifonia-project

[39]https://github.com/polifonia-project/rulebook

the ontology, and its documentation. As for the development, the ontology should: (i) have a namespace that can be dereferenced (relying on the w3id.org[40] service) and that follows a specific rule (https://w3id.org/polifonia/ontology/[name-of-the-ontology]); (ii) should be annotated with labels and comments; (iii) should be explicitly annotated with alignments to possible reused ODPs (using OPLaX ontology, see Chapter 4) and to indirectly reused external ontologies. As for the documentation, the ontology needs to be stored as an RDF/OWL file in a dedicated GitHub repository. A rule also defines how this repository should be named, in order to guarantee consistency across repositories, and what pieces of information the *readme* file of the repository should contain, namely: (i) a brief description of the scope of the ontology; (ii) statistics about the ontology, like the number of classes; (iii) examples of relevant Competency Questions with respective SPARQL queries; (iv) a graphical representation of the main classes and properties; (v) licence for the reuse; (vi) ontology tests run by following the eXtreme Design methodology. Moreover, an additional repository represents the whole network, by providing links to all ontology modules: therefore, information about the specific ontology must be added/updated in this ontology-network GitHub repository.

Guidelines for knowledge graphs. Similarly, these guidelines are split into those related to the development of the knowledge graphs, and those about the documentation. As for the development, the knowledge graph should: (i) contain individuals with a namespace that follows a specific rule (https://w3id.org/polifonia/resource/ followed by the local name of the class and by the SHA-1 hash function of the unique attribute(s) of the individual); (ii) be available on the web through a SPARQL endpoint. As for the documentation, each knowledge graph should: (i) be documented in a GitHub repository, that needs to follow the rules already defined in the Polifonia rulebook valid for all GitHub repositories; (ii) be accompanied by a *readme* file containing a brief description of its scope, the link to the SPARQL endpoint, useful statistics e.g. number of triples, the data sources from where the KG was derived from, examples of CQs and SPARQL queries to be run, licence for data

[40]https://w3id.org/

reuse. Moreover, this repository should contain a copy of the KG using a standard serialisation (e.g. turtle, RDF/XML).

Chapter 3

Observing patterns from ontologies

With the rise of the Linked Open Data (LOD) initiative, we are faced with an unprecedented amount of distributed knowledge about any kind of general-purpose or specialised domain. This knowledge is expressed in the form of knowledge graphs (KGs), usually leaning on formally defined ontologies. The widespread adoption of KGs in various domains, and the different processes for generating both ontologies and KGs, which often overlook a proper and intuitive documentation, have made the contents of these resources complicated [61]. Users, regardless of their objectives, would need to gradually discover, and understand, the content of a (possibly large) unfamiliar ontology or knowledge graph, i.e. to effectively explore it.

As for the ontology level, users may need to perform a progressive analysis of the content of the ontology with the goal of (i) understanding the structure of the ontology, (ii) learning about the domain(s) and main questions the ontology can answer, and consequently (iii) identifying whether the ontology can satisfy possible questions constituting their modelling requirements. A competency question may be (partially) addressed by an ontology fragment – a possible ontology design pattern (ODP) – intended as a subset of concepts connected by relations. As already proven by previous experiments [12, 9], ODPs support the reusability of the ontology: this is the reason why we believe that the exploration of an ontology would be more effective if it was based on the patterns contained in such ontology. Moreover, when multiple ontologies are to be analysed for e.g. choosing the more pertinent ontologies

– or ontology fragments – to be reused, the user would benefit from the possibility of exploring all ontologies at once, thus comparing them in a straightforward manner, rather than explore one ontology at a time, and try to compare the results of the analysis at a later time.

In this Chapter, we present a method aiming at empirically extracting ontology design patterns from ontologies, which may possibly support a pattern-based exploration and understanding of a (domain-specific) corpus of ontologies, as a preliminary step to different ontology engineering tasks including ontology reuse.

3.1 Motivation and problem addressed

There exist some solutions that aim at supporting users wanting to learn an ontology, to inspect it, and understand its content. The ability of visualisations to give an overview of a resource is something that a user could benefit from: an effective visual overview can be already useful for users to have a quick understanding of the concepts, relations and structure of an ontology and the domain it addresses. However, while this result can be easily obtained with small (a few tens of classes) ontologies, in the case of larger ontologies, for which a user particularly needs a support in the exploration, a satisfying solution – including sophisticated visualisation and filtering techniques – has not yet been found [29]. Moreover, a user may need, as a second step, to get to the details from the overview, while keeping the context of the ontology: this is another challenge that none of the existing tools attempted to solve yet [29]. Most tools use a node-link visualisation, a few of them allow to choose different types of visualisation. As stated in [29], it seems that only few developers of visualisation methods/tools have specific use cases in mind when they design them. Indeed, no visualisations designed so far offer a solution to the common use case of an ontology designer that needs to understand and compare different ontologies at once, for learning, for instance, which one covers a specific domain/modelling requirement, and with which specific ontology fragment.

Summarisation methods – those focused on the ontology level, rather than on the data level – can be useful for exploring an ontology too, and could be used as input to visualisation tools. These methods return a *summary* of the ontology in the form of a set of *relevant* classes (and possibly properties) from the source ontology [73, 25]. The importance of a node in an ontology can be computed by applying different measures (e.g. various types of centrality, density, popularity in search engines, etc.). While summarisation methods can be useful for some individual use cases and user goals (e.g. the user just needs to know which are the main entities modelled in the ontology), they may not fit other needs, e.g. when a user wants to understand all *facts* addressed in the ontology, not only the most important, without having to inspect all ontology entities one by one.

For instance, we may identify, using a summarisation technique (e.g. the one [69] is based on), that in an ontology on the cultural heritage domain the concepts *Cultural Object* and *Collection* are *key* ones; however, we would not be able to understand if that ontology allows to answer whether a cultural object has been in a collection. Moreover, it can not be automatically inferred that two ontologies address the same modelling problem only based on the fact that they have the same key concepts.

For instance, an ontology O_1 may model the relation of "being a member of a collection" (*membership*) between an object and a collection, as an object property `hasMember` with a class `Collection` as domain and a class `Object` as range, while an ontology O_2 may implement it as an n-ary relation, with the class `Membership` related to its arguments: the classes `Collection`, `Time`, and `Object` (see Figure 3.1). These specific implementations of relations can be seen as *empirical ontology design patterns*[1] (EODPs) [34], intended as implemented modelling solutions that can be empirically observed in actual ontologies or knowledge graphs, regardless their correctness or quality level. EODPs may or may not be considered as – or reuse – reference ontology design patterns (ODPs). Therefore, different ontology fragments, i.e. the empirical ontology design patterns, may implement the same

[1]What we term here *empirical ontology design pattern* corresponds to what we termed *observed ontology design pattern* in [4].

relational structure across different ontologies. We name *conceptual component* (CC) this complex, cognitive relational structure that an ontology engineer implements in an ontology as a set of ontology constructs (classes, properties, axioms, etc.). In our previous example, as depicted in Figure 3.1, *membership* is the conceptual component, implemented in different ways in two ontologies.

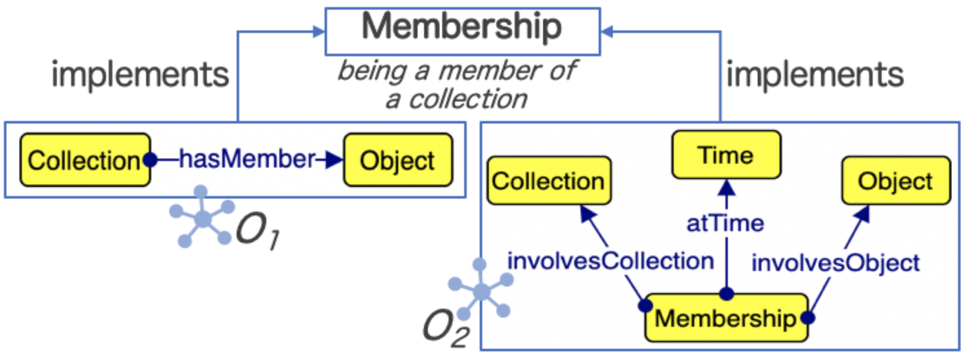


Figure 3.1: Implementations of the Membership CC from two different ontologies.

The notion of conceptual component takes origin from the concept of *knowledge pattern* presented in [38], which is in turn inspired by the concept of *frame*. The intended meaning of frame across the different theories can be summarised as "a structure that is used to organize our knowledge, as well as for interpreting, processing or anticipating information" [38]. Conceptual components are cognitive objects: they are the *intensional* counterparts of RDF/OWL implementations in ontologies of the semantic web[2]. If we are able to extract a number of conceptual components from an ontology, or a corpus of multiple ontologies, we have an indication of which types of facts, rather than which types of concepts/relations, an ontology (corpus) can represent. So, CCs provide a complete overview of the content of an ontology. Then, from this kind of summary in the form of a set of CCs, the user has access to all the EODPs that implement a specific CC, i.e. the modelling solutions adopted in specific ontologies, which answer to one (or more than one) competency question [46] and support specific inferences.

Available approaches for ontology summarisation, as well as other relevant meth-

[2]OWL has purely extensional semantics.

ods for partitioning an ontology into modules, making it more easily explorable (see Section 3.2), apply extractive techniques, meaning that the final summaries/modules contain as selected nodes the exact classes or properties from the source ontology. Instead, as part of the method presented in this Chapter, we apply a non-extractive technique. Indeed, the conceptual components that our approach identifies from a corpus of ontologies are not part of the original sources. After identifying the EODPs from each ontology, we group them in semantically-meaningful clusters: these clusters, as sets of EODPs from multiple ontologies, which are given a meaningful name, are our conceptual components. They provide a conceptual classification over the different implementations (the empirical ODPs) across various ontologies, and by being linked to their implementations and being organised as a hierarchical network, they may support understanding and comparison of the ontologies of a corpus. This method addresses a task, i.e. the exploration and understanding of existing ontologies relevant to a user, that is preliminary to other ontology engineering tasks, such as pattern-based ontology reuse, visualisation, matching and evaluation, having a potentially strong impact on semantic web interoperability.

3.2 Related work

Ontology selection and understanding. General catalogues of ontologies (e.g. vocab.org), catalogues of domain ontologies (e.g. BioPortal[3] for biomedicine), ODPs catalogues (e.g. ontologydesignpatterns.org), and semantic search engines (e.g. prefix.cc) aim at supporting ontology selection and reuse. With these catalogues, users can usually search individual ontology terms, but an actual comparison between multiple ontologies, e.g. for choosing the best one to reuse, is not supported. Moreover, none of them support a pattern-based browsing or filtering, in order to e.g. select the best ontology fragment that matches the user's requirements.

Most ontology summarisation approaches, e.g. those cited in [73, 25], look for the most important nodes (usually, classes) using centrality measures, such as between-

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

ness or PageRank, and other measures, e.g. the popularity intended as the normalised number of results returned by a search engine with a concept as keyword, as in [60]. Alternatively, they extract relevant subgraphs to support the querying and testing, aimed at validating requirements against available data. In both cases, by generating a summary based on the most informative nodes [25], they do not give an overview of possible *less key*, but still relevant, areas of an ontology. To the best of our knowledge, our method is the first one that uses a pattern-based approach, which has the advantage of providing an ontology designer with relevant ontology fragments to reuse based on specific modelling issues.

Ontology partitioning. Modularisation approaches e.g. [2, 28, 41] take as input a single ontology and return non-overlapping, consistent modules, which, if combined together, form the original ontology [28]. These methods are mainly based on logical and structural modularisation, and no additional insight about the content of the modules is provided, so that each module should be looked into in order to understand if it does satisfy a specific modelling requirement. On the contrary, each cluster of EODPs (each CC) is given a name, a description, and each EODP is accompanied by a diagram.

Complex ontology matching. Complex ontology matching consists in the generation of complex alignments, i.e. alignments that contain at least one entity that is a complex expression, on which a constructor or a transformation function has been applied [87]. [33] proposed a pattern-based approach to this task, which also provides a formalisation of the common structure of two (or more) aligned patterns, which could be a potential logical characterisation of CCs. Our method may be the basis to novel approaches or implementations to address the complex ontology matching task.

Patterns discovery. Ontology patterns discovery consists in discovering structures repeating frequently. [66] proposes a clustering of repetitive structures of axioms, and then tries to generalise them. Instead, our method starts from detecting dense communities in ontologies, and as a second step clusters them based on their vocabulary. The method by [58] investigates axiom patterns frequently recurring in ontologies, by proposing a tree-mining method: it first transforms ontology axioms in a tree shape and then uses association analysis to mine co-occurring axiom patterns, which may indicate emerging ODPs. However, this method does not take into account inferences, nor the influence of the vocabulary in the process.

3.3 Input ontology corpora

As previously explained, our method takes an ontology corpus as an input (which may even consist of a single ontology) for extracting the EODPs from the source corpus and grouping them in CCs. For our experiments, we used two corpora on two different domains as an empirical basis.

3.3.1 Cultural Heritage ontology corpus

The first corpus consists of 43 Cultural Heritage (CH) ontologies[4] that we have collected.

There are two main motivations for choosing this domain: (i) as clarified in Section 2.2, we have direct experience in modelling CH ontologies and have previous knowledge about existing ontologies on CH, and (ii) the requirements of CH ontologies are usually quite complex (as demonstrated in the case of ArCo ontology), hence we hypothesise that CH ontologies represent a good testbed from which to try to generalise our method. Indeed, cultural heritage consists of many different types of cultural properties (e.g. archaeological, numismatic, musical, etc.), all sharing some features (e.g. location, author), but each one also with distinguishing features, not shared with other types (e.g. the legend on a specific side of a coin), which can involve different levels of representation [20]. As for these features, there are at least three levels of representation [20]: (i) the inherent attributes of a cultural property, e.g. its material(s) and measures; (ii) the cataloguing process,

[4]https://github.com/stlab-istc-cnr/conceptual-components/tree/main/conceptual-components-extraction/corpora

aimed at recording pieces of information which describe a cultural property; (iii) the interpretation of this information based on certain criteria for further knowledge extraction. Moreover, there are many different domains that are related to the already vast domain of CH (e.g. geology for archaeological properties, chemistry for anthropological materials). Additionally, there are various cultural institutions, such as galleries, libraries and archives, describing various cultural entities, each providing possible different views on them and adopting different classifications and terminologies. Based on these premises, we believe that ontologies from the CH domain provide us with an adequate level of complexity and variety – wrt both the content and the vocabulary – that make our method generalisable to other domains.

Ontologies that specifically focus on related domains (e.g. geometry, chemistry) and top-level ontologies have been left out from the corpus. To select the ontologies we used two main sources: an online catalogue of ontologies, and an online survey.

Queries on LOV. Linked Open Vocabularies Repository (LOV)[5] is a repository of ontologies. In this repository, each ontology can be assigned one or more topical tag. We analysed and searched the *Vocabs* section of LOV by filtering the results with tags related to cultural heritage (such as Catalogs, FRBR, Metadata).

Online survey. Furthermore, we published a call to fill a Survey[6] and disseminated it on 3 mailing lists related to cultural heritage and ontology engineering, and on social medias: the survey was filled in by 40 people, and most of them were researchers. The survey required the users to (i) check the ontologies that they already knew, choosing from the list of ontologies that we selected from LOV; (ii) recommend other possible ontologies not included in the list; (iii) specify in which ontology projects they had used any of those ontologies, if any. Almost all ontologies were known by at least one participant, and four ontologies on the cultural heritage domain have been recommended from the users, and then added to our corpus. For each ontology, we retrieved the latest version available to be included in the corpus. Four of them are not monolithic ontologies, but ontology networks, i.e. they are

[5]https://lov.linkeddata.es

[6]https://t.co/ghwk6lxCOH?amp=1

composed of multiple modules: we treat each networked ontology as one ontology. When we do not encounter any issues related to import or inconsistency, we include the inferred version of the ontologies, i.e. the version with materialised inferences. We generate the inferred version using the OWL API[7] and the HermiT Reasoner[8]. For 10 ontologies we were not able to generate the inferred version, thus we include the asserted version.
The resulting CH corpus (see Table 3.1) counts a total number of 2,707 classes (with `rdf:type owl:Class` or `rdfs:Class`), with an average of ∼63 classes per ontology. As for the properties (both object properties `owl:ObjectProperty` and datatype properties `owl:DatatypeProperty`, `rdf:Property`), they are 9,132 in total, with an average of ∼212 per ontology. The total number of logical axioms is 26,392, with an average of ∼613 per ontology.

3.3.2 Conference ontology corpus

For the second corpus, we rely on the dataset of the Conference evaluation track[9] of the OAEI 2020 campaign (*Conf* for short). OAEI (Ontology Alignment Evaluation Initiative) is a coordinated international initiative, whose main objectives are assessing strengths and weaknesses of alignment/matching systems, comparing their performances, helping improve evaluation techniques of such systems. The means to achieve these goals include the organization of a yearly evaluation event, where different datasets of ontologies are used as testbed. The Conference corpus contains 16 ontologies[10] on the specific domain of conferences, which is less vast than CH but has, in any case, a good range of subtopics and related domains (e.g. travel, price). The total number of classes within the corpus is 851 (cf. Table 3.1), with an average of ∼53 classes per ontology; the total number of properties is 714, with an average of ∼44 properties per ontology. In this case, it was possible to compute the inferred

[7]https://github.com/owlcs/owlapi
[8]http://www.hermit-reasoner.com
[9]http://oaei.ontologymatching.org/2020/results/conference
[10]https://owl.vse.cz/ontofarm/#ontologies

Table 3.1: Corpora of ontologies: statistics

Dataset	**# ontologies**	**# logical axioms**				**# classes**				**# properties**			
		tot	avg	min	max	tot	avg	min	max	tot	avg	min	max
CH	43	26,392	∼613	16	1,060	2707	∼63	5	539	9132	∼212	6	4324
Conf	16	4097	∼256	65	739	851	∼53	14	140	714	∼44	17	78

versions of all 16 ontologies. The total number of logical axioms is 4,097, with an average of ∼256 axioms per ontology.

3.4 Approach

Our approach is summarised in Figure 3.2. The intuition our method is based on is that ontologies are designed, in an either intentional or unintentional way, as compositions of conceptual components, i.e. they offer solutions to some abstract modelling issues. Indeed, these conceptual components are implemented by (empirical) ODPs (EODPs). EODPs are the adopted modelling solutions, in the form of ontology fragments, i.e. sets of ontology entities (classes and properties, along with axioms).

Our hypothesis is that these EODPs can be observed based on their distinctive vocabulary and the high density of their internal connections. Specifically, their vocabulary shows a semantic coherence with the relational meaning they represent, i.e. the combination of words used for describing an EODP (through local names and labels of classes and properties) evokes that relation. For example, in an EODP addressing the *Membership* relation, a possible vocabulary may include terms such as collection, is member of, has member, has unifying property, which can be easily traced back to the membership concept. Moreover, we can hypothesise that the density of the connections (edges) inside the specific ontology fragment (EODP) is higher than the density of the connections between different EODPs. For example, let us consider an ontology that represents the address of an object as a class `Address` related to four arguments: the object located at that address, the city of the address,

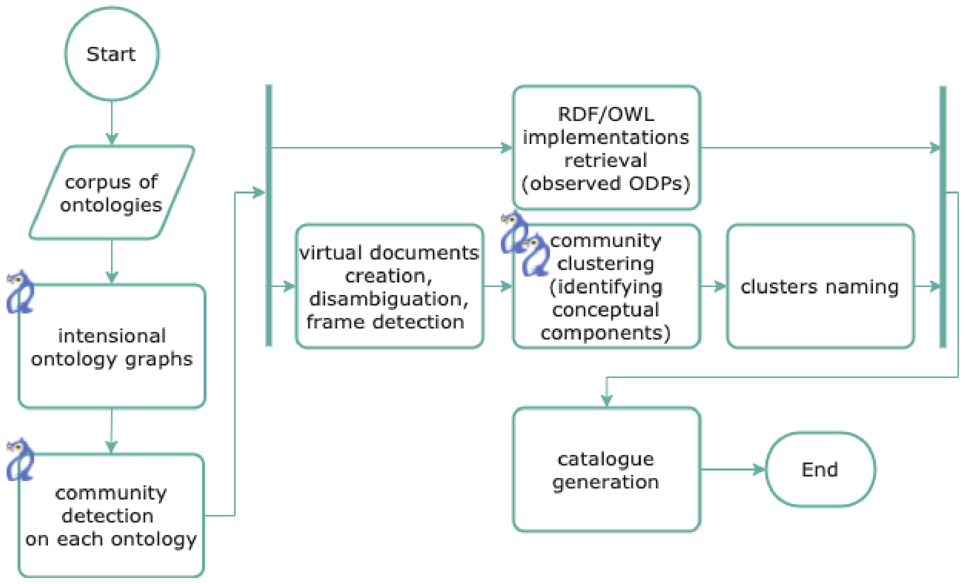

Start
corpus of ontologies
intensional ontology graphs
community detection on each ontology
virtual documents creation, disambiguation, frame detection
RDF/OWL implementations retrieval (observed ODPs)
community clustering (identifying conceptual components)
clusters naming
catalogue generation
End

Figure 3.2: Approach for conceptual components extraction.

the street and number, and the postal code. The class `Address`, the four arguments, and the properties that relate the main class to its arguments, form altogether an EODP *Address*. Let us imagine that the same ontology also includes another EODP *Event*, modelling events, their participants and other attributes (e.g. time, place). The connections between the entities included in the *Address* EODP, and the connections between the entities involved in the *Event* one, will be denser than the connections across *Address* and *Event*. This is a topological phenomenon that can be recognised by community detection algorithms, such as [27].

Our method, depicted in Figure 3.2[11], detects the communities from each ontology of a corpus of multiple ontologies (cf. Section 3.4.2), after a pre-processing step that generates what we call an *intensional ontology graph* for each ontology (cf. Section 3.4.1). Each community corresponds to a potential empirical ODP. Then, we recreate the OWL/RDF[12] fragment corresponding to a community (the actual EODP), extracting from the original ontology the ontology entities and axioms in-

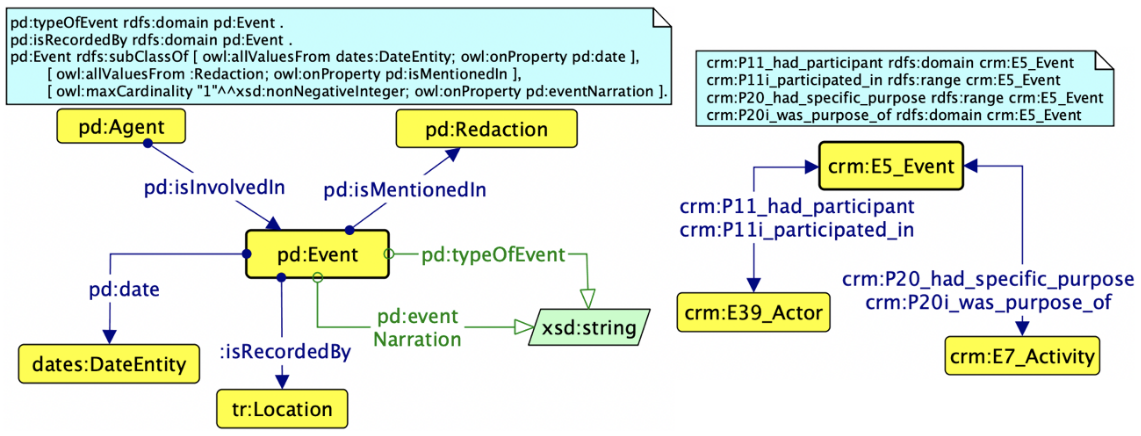
[11]In the Figure, one owl-logo means that the process works on one ontology at a time, two owl-logos that it works on the whole corpus.

[12]OWL ontologies, RDF vocabularies.

volved in that community. Additionally, from each community we also generate a virtual document, intended as a bag of words built through the concatenation of the vocabulary terms describing its entities (e.g. the values of the property `rdfs:label`). Each virtual document undergoes a disambiguation process and a frame detection step, and all documents are then given as input to a clustering algorithm (cf. Section 3.4.3). Each cluster that we obtain groups different communities (i.e. EODPs), potentially coming from different ontologies of the corpus. Based on our assumptions, each cluster corresponds to an empirically extracted conceptual component, and each EODP related to it is one of its possible OWL/RDF implementations. Each cluster is given a name based on some heuristics. Finally, the exploration of the ontology corpus based on its conceptual components and EODPs is supported by the generation of a *catalogue*, which provides an indexed summary of the whole ontology corpus.

3.4.1 Intensional ontology graph

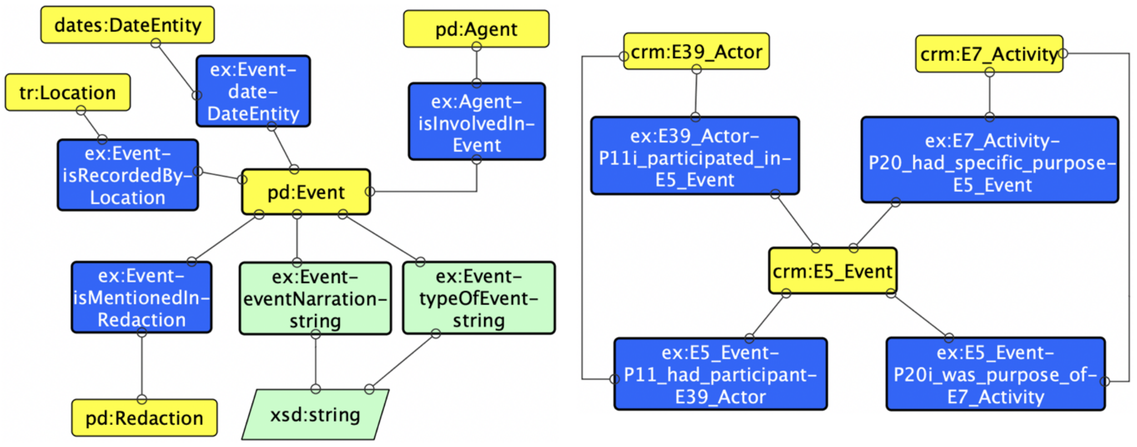
Most community detection algorithms manipulate undirected graphs, ignoring nodes and edges labels and focusing instead only on the topological structure of a network. Therefore, as a preliminary step, we transform the ontologies given in input into graph structures that preserve as much as possible the pieces of information about the formalisation of their conceptualisation, so that we can then have them processed by a community detection algorithm. We call this intermediate graph structure *intensional ontology graph*: it is a graph derived from an OWL/RDF ontology whose nodes correspond to the original classes and properties, and whose edges specify meaningful relations between the nodes (e.g. two classes, or a class and a property). Informally, with this graph we encode the intensional level of the ontology. The formal rules that we defined to translate an ontology to the respective intensional ontology graph are presented in Listing 3.1. With the notation *edge_label (source_label, target_label)*, we indicate a pair of nodes *source_label, target_label* that are connected by the edge *edge_label*, in the intensional graph. To indicate undirected and unlabelled edges we use *no_label*. A rule is a set of premises, expressed in



```
pd:typeOfEvent rdfs:domain pd:Event .
pd:isRecordedBy rdfs:domain pd:Event .
pd:Event rdfs:subClassOf [ owl:allValuesFrom dates:DateEntity; owl:onProperty pd:date ],
        [ owl:allValuesFrom :Redaction; owl:onProperty pd:isMentionedIn ],
        [ owl:maxCardinality "1"^^xsd:nonNegativeInteger; owl:onProperty pd:eventNarration ].
```

```
crm:P11_had_participant rdfs:domain crm:E5_Event
crm:P11i_participated_in rdfs:range crm:E5_Event
crm:P20_had_specific_purpose rdfs:range crm:E5_Event
crm:P20i_was_purpose_of rdfs:domain crm:E5_Event
```

(a) Two EODPs from the CH corpus.


dates:DateEntity
pd:Agent
tr:Location
ex:Event-date-DateEntity
ex:Agent-isInvolvedIn-Event
ex:Event-isRecordedBy-Location
pd:Event
ex:Event-isMentionedIn-Redaction
ex:Event-eventNarration-string
ex:Event-typeOfEvent-string
pd:Redaction
xsd:string
crm:E39_Actor
crm:E7_Activity
ex:E39_Actor-P11i_participated_in-E5_Event
ex:E7_Activity-P20_had_specific_purpose-E5_Event
crm:E5_Event
ex:E5_Event-P11_had_participant-E39_Actor
ex:E5_Event-P20i_was_purpose_of-E7_Activity

(b) Intensional graphs corresponding to the OWL EODPs in 3.3a.

Figure 3.3: Example of OWL EODPs and their corresponding intensional ontology graphs. Blue rectangles indicate object properties, green rectangles data properties.

turtle syntax, and a conclusion, expressed with the introduced notation, that follows the symbol "→".

Frame 3.1: Transformation rules from an OWL/RDF ontology to its corresponding intensional graph.

```
(r1) :p rdfs:domain :d  .  :p rdfs:range :r . → :p(:d, :r)

(r2) :c1 rdfs:subClassOf | owl:equivalentClass [
  owl:onProperty :p ;
  owl:someValuesFrom | owl:allValuesFrom | owl:hasValue |
  owl:maxCardinality | owl:minCardinality | owl:cardinality :c2 ]
```

```
    → :p(:c1, :c2)

(r3) :p(:n1, :n2) → no–label(:n1, :n1–p–n2) no–label(:n1–p–n2, :n2)
```

Given an ontology O, the first rule (r1) generates an edge :p linking two nodes, :d and :r, for all properties that have domain :d and range :r. We do not consider domain/range declarations that involve blank nodes. We assume that properties that do not specify their domain/range have `owl:Thing` as domain/range. Existential, universal, and cardinality property restrictions on classes generate an edge :p between the class local to the restriction and the class in the restriction expression (r2). We ignore all class expressions, that is we only consider restrictions involving named classes or datatypes. We empirically verified on our corpora that the loss of information and the impact of ignoring class expressions is almost insignificant: only 1.62% of subClassOf/equivalence axioms and 5.42% of domain/range axioms involve class expressions in the CH corpus, while 1.48% of subClassOf/equivalence axioms and 9.22% of domain/range axioms involve class expressions in the Conf dataset.

Hierarchical and equivalence relations between classes and properties are left out of the intensional graph, to avoid merely taxonomic patterns, but they are reintroduced when the OWL/RDF EODPs are retrieved (cf. Subsec. 3.4.2).

By applying rules (r1) and (r2), we obtain a labelled *multi-graph*, that is a graph having multiple labelled edges. The last rule (r3) translates the intensional graph resulting from (r1) and (r2) to the corresponding unlabeled and undirected graph, that is the input needed to the chosen community detection algorithm. For each edge :p between two nodes :$n1$ and :$n2$, rule (r3) generates two unlabelled edges. The first edge connects $n1$ to a new node :$n1 - p - n2$, the second edge connects :$n1-p-n2$ to :$n2$. The node :$n1-p-n2$ represents the meaning of the property :p, contextualised to its use for connecting :$n1$ and :$n2$. This maximises the quality of the detected communities. Indeed, communities extracted with community detection algorithms are disjoint, hence if we only store information about a property :p, this property will only end up in one community. However, a same property :p may be relevant to different contexts (and EODPs), and these contexts are captured by its local usage, i.e. the predicates it connects. With this representation we enable overlapping communities, which is crucial to capture entities that are relevant to multiple patterns.

We transform each ontology included in the two corpora into its *intensional* ontology graph. Figure 3.3 shows two EODPs (in 3.3a) from POSTDATA (on the left) and CIDOC CRM (on the right) and their corresponding intensional graphs[13] (in 3.3b).

3.4.2 Community detection

Community detection gathers the nodes of a network into groups, in a way that there is a higher density of edges within groups than between them. For detecting the communities from each ontology, we use the Clauset-Newman-Moore algorithm [27]. This community detection algorithm is based on the greedy optimization of the *modularity*, i.e. a measure of how much the computed division is good, based on the ratio between the number of edges inside the communities and the number of edges between them. At the first step, there are as many communities as the nodes, where each node is the only member of its own community, then the two communities that, if merged, most increase the modularity, are repeatedly joined, until it is no longer feasible to merge communities without decreasing the modularity.

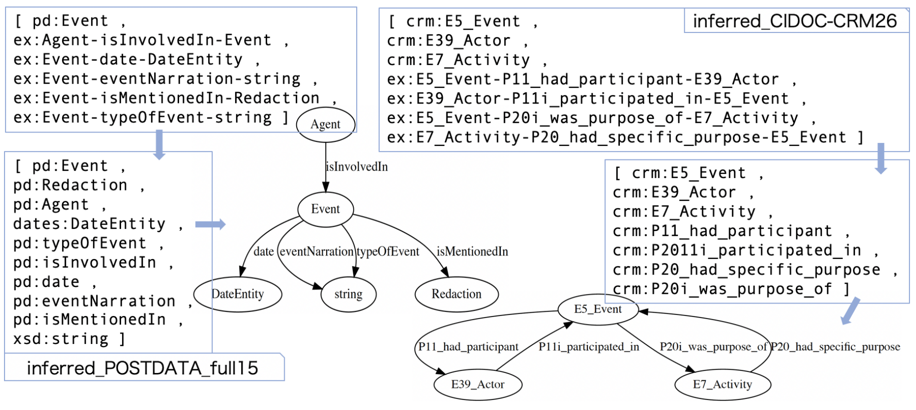
After running this algorithm on the intensional ontology graphs, we observed that there is a significant difference in the density of the detected communities, and that many communities having a lower density could be further split into more meaningful sub-communities. Based on our experiments, we found that recursively running the algorithm on communities with a density lower than the average density of all communities detected at the previous step, would improve the results. Therefore, we modified the algorithm such that it would behave in this way, until there is

[13]`pd:` http://postdata.linhd.uned.es/ontology/postdata-core# `tr:` http://postdata.linhd.uned.es/ontology/postdata-transmission# `dates:` http://postdata.linhd.uned.es/ontology/postdata-dates# `crm:` http://www.cidoc-crm.org/cidoc-crm/

no community that can be split further.

OWL/RDF EODPs. The community detection step generates communities as sets of nodes. In order to obtain the actual empirical ODPs, and to better study and present their structure and content, we retrieve the OWL/RDF ontology fragments that contain the original nodes, both classes and properties. To define what to include and determine their boundary, we use the following heuristics: for each node in the community (both classes and properties), we retrieve the triples that assert its type(s). As for properties, we include domain and range axioms, inverse properties, super-properties and equivalent properties. As for the classes, we retrieve all super- and equivalent classes, and all restrictions that involve at least one property that is inside the community. Figure 3.4 depicts the sets of nodes retrieved from the two communities depicted in Figure 3.3b (from POSTDATA and CIDOC)[14].

As it can be noticed, these communities almost perfectly overlap with the ontology fragments in Figure 3.3a.



```
[ pd:Event ,
ex:Agent-isInvolvedIn-Event ,
ex:Event-date-DateEntity ,
ex:Event-eventNarration-string ,
ex:Event-isMentionedIn-Redaction ,
ex:Event-typeOfEvent-string ]
```

```
[ pd:Event ,
pd:Redaction ,
pd:Agent ,
dates:DateEntity ,
pd:typeOfEvent ,
pd:isInvolvedIn ,
pd:date ,
pd:eventNarration ,
pd:isMentionedIn ,
xsd:string ]
```

inferred_POSTDATA_full15

inferred_CIDOC-CRM26

```
[ crm:E5_Event ,
crm:E39_Actor ,
crm:E7_Activity ,
ex:E5_Event-P11_had_participant-E39_Actor ,
ex:E39_Actor-P11i_participated_in-E5_Event ,
ex:E5_Event-P20i_was_purpose_of-E7_Activity ,
ex:E7_Activity-P20_had_specific_purpose-E5_Event ]
```

```
[ crm:E5_Event ,
crm:E39_Actor ,
crm:E7_Activity ,
crm:P11_had_participant ,
crm:P2011i_participated_in ,
crm:P20_had_specific_purpose ,
crm:P20i_was_purpose_of ]
```

Figure 3.4: Example of communities detected from two ontologies of the CH corpus.

[14]Arrows mean consecutive steps.

3.4.3 Clustering and catalogue generation

Communities are detected based on the topological features of the intensional graphs. We hypothesise that each community identifies an EODP. The vocabulary used in an ontology (pattern) – e.g. through labels – contributes, along with formal constructs, to define the relational meaning(s) represented by that ontology (pattern). We believe that the terms in their vocabulary will contribute to evoke the relational meaning captured by these EODPs. These relational meanings correspond to (possible specialisations of) the *conceptual components* that we are trying to extract. Since we start from a corpus of multiple ontologies, if we cluster all communities from the corpus according to their vocabularies, we may identify CCs that group together EODPs across all ontologies.

Clustering input. We generate a virtual document corresponding to each community by the concatenation of all `rdfs:label` values from the entities included in the community. We take all English terms in the labels and, when no label is present, we extract and use the local ID. We remove the repetitions and do not consider comments, since in many cases they may introduce noise, e.g. if they contain long explanations with examples. Entities with namespaces `owl:`, `rdf:`, `rdfs:` and `xsd:` are excluded from the virtual documents. At this point, we disambiguate all virtual documents by using the UKB[15] Word Sense Disambiguation tool, which is based on WordNet (English) version 3.0[16]. Then, we query the profile B of Framester[17], a knowledge graph that links many linguistic resources (including WordNet and FrameNet[18]), in order to extract all FrameNet frames that have a *close match* with (i.e. are evoked by) the synsets of the virtual documents, and we enrich the documents with such frames. We also exploit the hierarchy of frames to include additional, more general frames. As a result of this step, each community is represented by the concatenation of all the synsets and frames that have been retrieved.

[15]https://github.com/asoroa/ukb
[16]https://wordnet.princeton.edu/
[17]https://github.com/framester/Framester
[18]https://framenet.icsi.berkeley.edu/fndrupal/

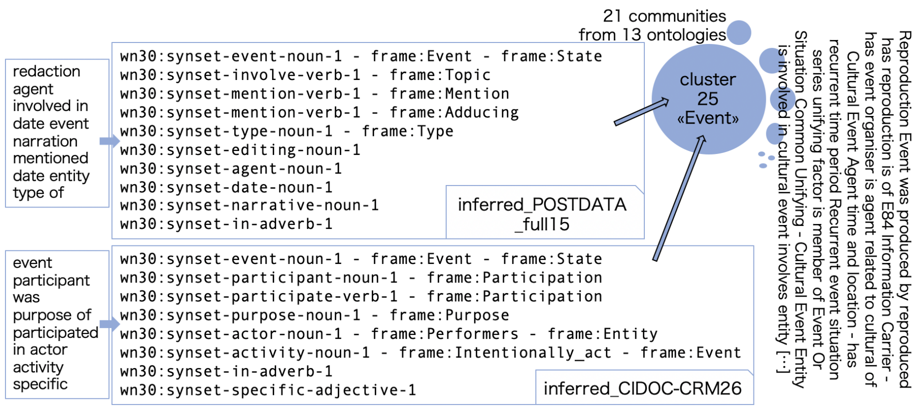

21 communities
from 13 ontologies
redaction agent involved in date event narration mentioned date entity type of
wn30:synset-event-noun-1 - frame:Event - frame:State
wn30:synset-involve-verb-1 - frame:Topic
wn30:synset-mention-verb-1 - frame:Mention
wn30:synset-mention-verb-1 - frame:Adducing
wn30:synset-type-noun-1 - frame:Type
wn30:synset-editing-noun-1
wn30:synset-agent-noun-1
wn30:synset-date-noun-1
wn30:synset-narrative-noun-1
wn30:synset-in-adverb-1
inferred_POSTDATA_full15
cluster 25 «Event»
event participant was purpose of participated in actor activity specific
wn30:synset-event-noun-1 - frame:Event - frame:State
wn30:synset-participant-noun-1 - frame:Participation
wn30:synset-participate-verb-1 - frame:Participation
wn30:synset-purpose-noun-1 - frame:Purpose
wn30:synset-actor-noun-1 - frame:Performers - frame:Entity
wn30:synset-activity-noun-1 - frame:Intentionally_act - frame:Event
wn30:synset-in-adverb-1
wn30:synset-specific-adjective-1
inferred_CIDOC-CRM26
Reproduction Event was produced by reproduced has reproduction is of E84 Information Carrier - has event organiser is agent related to cultural of Cultural Event Agent time and location - has recurrent time period Recurrent event situation series unifying factor is member of Event Or Situation Common Unifying - Cultural Event Entity is involved in cultural event involves entity [···]

Figure 3.5: Virtual document disambiguation, frame detection and clustering on the communities from Fig. 3.4.

Figure 3.5[19] shows the synsets and frames included in the virtual documents of the two communities depicted in Figure 3.4.

Clustering. At this point, we cluster the virtual documents of the communities, using the K-Means algorithm [62] and the elbow method. K-Means is a general-purpose clustering algorithm that has been tested in different application areas and domains [92]. K-Means splits the observations into a predefined number of k disjoint groups, defining, after a number of iteration, k centroids, one for each cluster.
The resulting conceptual components, i.e. the clusters, are placed in a hierarchical network. To define hierarchical relations between them we use inheritance relations existing between FrameNet frames.

Given a cluster c, we indicate with $F(c)$ the set of frames associated with c. Two clusters c_1 and c_2 have a hierachical relation $r(c_1, c_2)$, with an associated weight w, if at least one frame $f_1 \in F(c_1)$ inherits from at least one frame $f_2 \in F(c_2)$. We indicate with max the maximum number of inheritance relations between frames that occur between two clusters of the network. The weight w reflects the strength of $r(c_1, c_2)$ and is computed in the following way. Given two clusters c_1 and c_2, the strength w of $r(c_1, c_2)$ is the sum of the frames in c_1 that are subsumed under at

[19]`wn30:` https://w3id.org/framester/wn/wn30/instances/ `frame:` https://w3id.org/framester/framenet/abox/frame/

least one frame in c_2, divided by max. The values for w ranges between 0 and 1 [0,1].

Naming conceptual components. To automatically associate a meaningful name with each cluster, which aims at giving a name to the conceptual component, we derive a label, which is manually checked, from the most frequent synsets and frames that belong to the cluster, i.e. we count how many times a same synset or frame is included in the virtual documents belonging to a cluster, i.e. the number of repetitions. Additionally, each cluster is accompanied by a textual description that merely concatenates all terms (coming from the labels or the local IDs) representing its communities. This description is useful to better understand the more specific concepts included in the empirical ODPs grouped by a cluster. For example, the communities in Figure 3.5 end up in the same cluster, which is given the name *Event*: the most frequent frame within the cluster (41 times from 21 communities that belong to 13 different ontologies). The description indicates that the ontologies implementing the *Event* conceptual component cover different types of events (e.g. cultural events, reproductions), and entities related to events, such as organisers, time, etc.

Catalogue generation. The last step of the method (cf. Figure 3.2) generates a *catalogue* that connects and organises the input ontologies according to the conceptual components that have been extracted from the whole corpus, and their corresponding EODPs. From each CC in the catalogue is possible to access its associated EODPs that implement it within the ontologies. Therefore, the catalogue classifies the ontologies based on the conceptual components that they implement. We provide an HTML rendering of the catalogue[20] included in the online package[21], generated from the CH corpus.

[20]https://stlab-istc-cnr.github.io/cc-and-odps-catalogue/

[21]https://github.com/stlab-istc-cnr/conceptual-components

3.5 Experiment and results

The overall time required for producing all results with our method is about 1h15m for the CH corpus and 30m for the Conference corpus, using a commodity hardware: specifically, we used a laptop (2,3 GHz Intel Core i5, 16GB of RAM).

Intensional graphs. The average number of nodes and edges of the intensional graphs generated from the CH corpus is ∼165 and ∼217, respectively. For the Conf corpus, they are ∼91 and ∼115. The intensional graphs keep an average of 47% (CH corpus) and 54% (Conf corpus) of classes and 90% (CH corpus) and 87% (Conf corpus) of object and datatype properties. The loss of information about ontology classes is caused by the fact that the transformation rules from the ontology to the intensional graph, as defined in Listing 3.1, are biased towards ontologies with rich axiomatisation: ontologies that have poor axiomatisation are mostly affected by loss of classes and properties. However, it is important to notice that this is also due to the fact that all superclasses and superproperties are discarded in this step, while they are all re-included when the EODPs are retrieved.

Community detection. The total number of communities detected is 1,300 from the CH corpus. Only in one case, that is for the RDA ontology, the algorithm could not split the ontology in different communities. The ontology with the greatest number of communities is ArCo (363 communities). The average number of communities per ontology is ∼30. As for the Conf corpus, from 16 ontologies our algorithm detects a total of 419 communities, with an average of ∼26 communities per ontology. The minimum number of communities per ontology is 8, while the maximum is 83.

Clustering. We convert the virtual documents in numerical feature vectors and apply tf-idf in order to discard too frequently occurring tokens. Our setting ignores terms that have a document frequency higher than 90%, while a minimum value was not specified. To evaluate the optimal number of clusters k for our data, we used the elbow method and we run the algorithm with a fixed number of 100 (CH

Table 3.2: The number of hierarchical relations among clusters per level of strength.

	Strength levels														
	0.0			0.1			0.2			0.3			0.4		
	tot	max	avg	tot	max	avg	tot	max	avg	tot	max	avg	tot	max	avg
CH	6644	91	69.2	813	66	8.4	274	42	2.8	114	30	1.18	58	22	0.6
Conf	2000	47	25.9	572	30	7.4	260	25	3.3	133	15	1.72	63	11	0.8

dataset) and 81 (Conf dataset) clusters, as a result of the elbow method. Being K-Means nondeterministic, we set the *random state* parameter to a commonly used integer value (42), so that our cluster assignments is reproducible.

For the CH corpus, the average number of communities per cluster is 13, with a maximum of 111, and a minimum of 3. Each cluster includes communities that come from an average of ∼4.5 different ontologies. 11 clusters contain communities coming only from one ontology. 88 clusters group an average of ∼15 communities that belong to a range between 2 and 10 different ontologies. 1 cluster includes 111 communities from 26 different ontologies.
As for the Conf corpus, the average number of communities per cluster is 5.17, with a maximum of 13, and a minimum of 1. The communities in each cluster belong to an average of ∼2.6 different ontologies. 25 clusters include communities from the same ontology, the remaining 56 clusters group an average of ∼7.2 communities that belong to a range between 2 and 8 different ontologies.
Catalogue. Table 3.2 gives an overview of the number of hierarchical relations among the clusters per level of strength (see Section 3.4.3). The strength levels reported in the table are those used in the experiments. For each level l, it is indicated the total, maximum and average number of relations having a strength $\geq l$.

3.6 Evaluation and discussion

3.6.1 Manual inspection of communities

A manual inspection of the communities, analysing both their structures and labels, has been a preliminary step for assessing the quality and soundness of our results. In the context of this analysis, we defined four categories of communities based on their quality: bad, medium, good, ideal. A community is *bad* if it can belong to more than two CCs: this means that it lacks a conceptual coherence, it actually addresses different (more than two) modelling problems, and this is usually due to an implementation (i.e. the EODP) that is poorly axiomatised. For instance, a community from the Conf corpus contains 27 heterogeneous properties (e.g. *created by* and *has conflict type*), and these properties, in the original ontology, are not involved in any kind of restriction, e.g. domain or range. As another example, a community from the CH corpus includes unrelated properties that have the same domain and the same range (`xsd:string`). In these cases, the topology did not help to identify significant modules, while an analysis of the vocabulary highlighted the presence of different conceptual areas. The detection of bad communities, along with good ones, may be a useful result for evaluating the quality of an ontology. A community has *medium* quality if it can belong to (not more than) two different CCs. A community is *good* if it can be assigned to exactly one CC, but includes a maximum of 20% intruders entities (i.e. ontology entities that are incoherent with respect to the others, and the conceptual component). An *ideal* community has less than 20% intruders.

About 8% of communities in both the CH and Conf ontologies are *bad*. ∼7% (CH) and ∼3% (Conf) have *medium* quality, ∼17% (CH) and ∼5% (Conf) are *good*, while the majority of the communities of both corpora (∼67% for the CH corpus, ∼84% for the Conf corpus) have an *ideal* level of semantic coherence. As an example, see the two implementations of the CC *Event* presented in Figure 3.5.

Let us take two additional examples from the two corpora. A community from the Conf dataset (extracted from the cmt-2 ontology) identifies an EODP modelling the fact of *being a member of a conference*: it includes the two (explicitly declared) inverse binary relations and the concepts *conference* and *conference member*, which are, respectively, their domain and range. A CH community from CIDOC CRM implements an EODP for capturing that an object changed its ownership: it contains the core concept *acquisition* (`crm:E8_Acquisition`), and the classes and properties modelling the physical entity involved and the actors that acquired and surrendered the title over it. By inspecting the EODP, and the original ontology, we found that all properties in this fragment have domain and range restrictions; however, inverse relations between object properties are not asserted.

3.6.2 Clustering: similarity

For a first assessment of the quality of the clusters, we computed a pairwise similarity among them. In particular, we chose to adopt the Overlap Coefficient[22], which is commonly used in data mining techniques, for measuring the overlap between the sets of synsets and frames of a pair of clusters. This score shows how similar two clusters are, and its values ranges from 0.0 (i.e. dissimilar) to 1.0 (i.e. similar). We can conclude that, on average, the clusters of both corpora score very low (0.20 for CH and 0.17 for Conf): this indicates a good quality of the clusters.

3.6.3 Clustering: manual inspection

The clusters that have been extracted from both corpora identify a wide range of different conceptual components, with different levels of abstraction. There are some general conceptual components, such as *Event*, *Categorization*, *Membership*, and *Intentionally act*, that emerge from both corpora. CCs common to different corpora can support the interoperability between ontologies addressing different domains. Other components are instead more peculiar to the specific domain: for instance, *Performing arts*, *Measurement* and *Attribution* from the CH corpus; *Submitting documents*, *Respond to proposal*, *Award* from the Conf corpus. By inspecting a CC,

[22]https://en.wikipedia.org/wiki/Overlap_coefficient

it is possible to compare implementations (EODPs) from different ontologies and choose the one, if any, that best fits our requirements. For example, for a user that needs to model the acquisition process of a cultural property, there would be two candidate reusable EODPs, one from the CIDOC CRM ontology and one from ArCo. These two implementations of the CC *Acquisition* partially overlap: ArCo addresses the acquisition place and time too, while it does not represent the new owner, which is instead included in the CIDOC EODP.

In both corpora, there are some clusters that could be either split or merged. On the one side, if no frames/synsets clearly emerge from a cluster, it may be a signal that it is actually grouping different conceptual components. On the other side, the emergence of the same frame(s) as the most frequent in different components (thus, these CCs will be given the same name) may indicate that they could be merged, or that they are a specialisation of the same conceptual component: looking at less frequent frames and at their hierarchical relations could clarify this.

3.6.4 Evaluation against an ontology engineering task

Along with this manual evaluation, we evaluate our method by also analysing our results in the context of the ontology matching task, which is one of the ontology engineering tasks our work may have an impact on. While this is an indirect evaluation, we believe it is useful for assessing the quality of our method.

The hypothesis behind this evaluation is that, given a pair of ontology entities that should be aligned, through subsumption or equivalence, these entities should belong to either the same cluster or two hierarchically related clusters. Considering that a cluster groups EODPs from different ontologies based on their close semantics, a human or artificial agent performing ontology alignment activities on a corpus of ontologies, can start the identification of entities that should be aligned by looking within a same cluster or following strong hierarchical relations between clusters. The question we want to answer with this evaluation is whether a good number of these alignments can be identified with our approach.

We rely on three sets of alignments to compare our results: (i) a set AA of 224 asserted and curated alignments (1 equivalence and 223 subsumptions) from the CH corpus; (ii) a set AML of 237 alignments (all equivalences) generated by AgreementMakerLight [30], which proved to be the best ontology matching tool in most of the OAEI 2020 tracks, for all pairs of ontologies included in the CH corpus; (iii) a dataset CA of 224 alignments on the Conf corpus (all equivalences) that has been used as a gold standard in the OAEI 2020 conference track[23].

The AML dataset annotates each alignment with a confidence score cs, while for the CA and AA datasets we assume that cs is equal to 1 for all alignments. We introduce $AML_{.90} \subseteq AML$ and $AML_{.99} \subseteq AML$ which are the sets containing the alignments having a confidence score ≥ 0.9 and ≥ 0.99, respectively.

To measure the quality of our results, we want to verify whether, given a set of alignments A, the pairs of entities belonging to A are assigned to a same cluster or to related clusters, with the same cs provided for that alignment. For example, if a pair $(e1, e2)$ belongs to AML with a confidence score $cs = 0.98$, then we assume that AML would assign $(e1, e2)$ to the same cluster or to two related clusters with $cs = 0.98$. Finally, we define the sets D, I, H and E to provide an interpretatio of the results of our method. Given a set of entity pairs D from the alignment in AA, CA or AML, we introduce: (i) the set $I \subseteq D$ as the set of entity pairs in D, that are member of the same clusters; (ii) $H \subseteq D$ as the set of entity pairs that belong to hierarchically related clusters; and (iii) $E := I \cup H$. H_n (and, similarly, E_n) indicates the set of entity pairs that belong to two clusters related with strength $l \geq n$. We remind that n = [0,1] is the strength of the hierarchical relation between two clusters.

We introduce the measure *corr* (see the Formula 3.1) in order to compute the correlation between the alignment sets and the results of our method. Given two sets of entity pairs A and B, where each pair is associated with a confidence score $cs(e_i, e_j)$, we define *corr(A,B)* as the sum of all cs of the alignments in A divided

[23]http://oaei.ontologymatching.org/2019/conference/data/reference-alignment.zip

Table 3.3: Correlation between reference alignments (AA, CA, AML, $AML_{.90}$ and $AML_{.99}$) with the sets I, E, $E_{0.1}$, $E_{0.2}$, $E_{0.3}$, $E_{0.4}$.

Alignments	I	E	$E_{0.1}$	$E_{0.2}$	$E_{0.3}$	$E_{0.4}$
AA	.21	.99	.64	.46	.34	.32
AML	.47	.99	.77	.64	.58	.56
$AML_{.90}$	.46	.99	.77	.64	.57	.55
$AML_{.99}$	.51	1.0	.75	.63	.60	.57
CA	.27	.76	.51	.43	.36	.35

by the sum of all cs in B, that is:

$$corr(A,B) = \frac{\sum_{(e_i,e_j)\in A} cs((e_i,e_j))}{\sum_{(e_i,e_j)\in B} cs((e_i,e_j))} \tag{3.1}$$

The cs associated with the alignments of AA and CA is 1.0. The correlation ranges from 0.0 (which means no correlation) to 1.0 (that is strong correlation). The entity pairs from our method inherit the cs value from the comparing set. Intuitively, $corr$ results from the computation of the ratio between the pairs that should be aligned and the pairs that belong to the same or to two related clusters. Table 3.3 reports the value of $corr$ computed for comparing AA, CA, AML, $AML_{.90}$ and $AML_{.99}$ (the testing sets) with the sets I, E, E_n, $E_{0.2}$, $E_{0.3}$, $E_{0.4}$.

Discussion. Almost all cultural heritage entity pairs aligned in the testing sets ($corr \geq .99$) can be found either into the same cluster or in two related clusters, a lower number for CA pairs ($corr = .76$) (see column E of Table 3.3). In the worst case, all hierarchical relations between clusters are to be inspected (69.2/CH and 25.9/Conf on average per cluster). This task may sound inconvenient to be performed manually, however we remark that an entity-to-entity analysis of the ontologies in the two corpora would require the inspection of 43 ontologies (CH corpus) and 16 ontologies (Conf corpus) and, in the worst case, of 11839/1565 ontology entities (classes and properties), respectively. An artificial agent, that is an ontology alignment algorithm, may exploit our clusters and their hierarchical relations to be integrated in a method for ranking candidate pairs inside a corpus – at the moment, ontology alignment tools only work with a pair of ontologies at a time. By setting a threshold for l, i.e. by ignoring weaker hierarchical relations, the value of *corr* decreases, but it remains reasonably good for the CH corpus until up to $l = 0.3$ (with only 1.18 average relations per cluster). With $l = 0.4$, it is possible to find up to 57% of the most precise alignments ($AML_{.99}$) by looking into entities belonging to the same clusters. As for AA, the performance is the worst in our experiment e.g. for column I. To better understand this result we run AgreementMakerLight on the CH corpus and compared its results against AA (that are curated alignments asserted in the ontologies). Only 1.5% of the alignments are identified. Our approach does not aim to identify alignments, therefore we cannot claim to perform better than AgreementMakerLight, however we believe that this result (see Table 3.3), as compared to this extremely bad performance, supports our hypothesis that clusters and their relations may be exploited to improve the performance of ontology alignment algorithms.

3.6.5 The ArCo use case

In this Section, we use the ArCo ontology network as a use case for evaluating our communities and clusters involving ArCo ontological entities. Indeed, as already discussed in Section 2.2, this ontology we contributed to has been explicitly developed following a pattern-based methodology, and is richly documented. This evaluation shows that (i) our catalogue of CCs could contribute to support the identification of ontology fragments (EODPs) that address specific competency questions, hence possibly easing the ontology selection and reuse process; (ii) the EODPs we detect may suggest possible missing axioms in the input ontology (e.g. in the case of *bad* EODPs), and integrations or possible modifications to the ODPs documented in a *top-down* manner, thus possibly supporting evaluation/refactoring activities.

CQ-based evaluation. Competency questions, besides being a crucial means for defining requirements and driving the design activities, can also be used to document an ontology, making it clear which are the main requirements it addresses;

this contributes to support an ontology designer that wants to understand, and possibly reuse, such ontology.

In the context of the ArCo project, ArCo's ontology design team has defined a set of relevant competency questions the ontology network provides an answer to, along with the respective SPARQL queries[24]. We use these CQs, and SPARQL queries, to analyse how our catalogue of CCs would support the identification of the most appropriate ArCo fragments to be looked into, and possibly be reused, for answering any of these CQs. To this end, we generalised the CQs that were strictly related to the data in the ArCo knowledge graph, such that they would include only elements that are defined within the ontology, e.g. *What are the complex cultural properties which have a number of components greater than 2?* becomes *Which is the number of components of a complex cultural property?*. In this process, some CQs corresponded to the same generalised CQ, thus we do not consider duplicates. Moreover, we split CQs clearly including more CQs inside, e.g. *What is the conservation status of the cultural property X? Which interventions have been proposed?*. We obtain a list of 43 competency question - SPARQL query pairs[25]. We consider as (i) *totally supported* (TS) a CQ the answer of which can be found in one (or more than one) community with *ideal* or *good* quality, and this community is included in one (or more than one) CC, which groups coherent EODPs addressing the same modelling problem; (ii) *with a medium support* (MS) a CQ the answer of which can be found in multiple *ideal* or *good* communities, which are partly included in *good* CCs – intended as explained in (i) – and partly included in *bad* CCs, as grouping partially incoherent EODPs; (iii) *with a low support* (LS) a CQ the answer of which can be found in one (or more than one) *ideal* or *good* community included in *bad* CC(s); and (iv) *not supported* (NS) a CQ the answer of which can be hardly found, since it is included in a community with a *medium/bad* quality, inside a *bad* CC.

[24]https://github.com/ICCD-MiBACT/ArCo/blob/master/ArCo-release/test/CQ/CQs-SPARQLqueries.txt

[25]The 43 CQs, along with their evaluation, can be found here: https://github.com/stlab-istc-cnr/conceptual-components/blob/main/conceptual-components-extraction/results/communities-manual-evaluation/evaluationCCs-ArCoCQs.xlsx

28/43 CQs are *totally supported*: for instance, the competency question *What is the place and time of the exhibition which includes the cultural property X?* is addressed by one Empirical ODP, which is included in a CC named *Event*, grouping together 21 EOPDs related to events. 2/43 CQs have a *medium support*. E.g., the CQ *When was the catalogue record about the cultural property edited or updated?* is addressed by two EODPs, one in a good CC named *Event*, the other in a wrong CC about descriptions (*Communicate categorization*). As another example, *What are the dimensions of a photograph X?* is answered by 3 EODPs, two of which are in the appropriate CC named *Measurement*, while the third one is a CC named *Statement*. 11/43 CQs have a *low support*, e.g. the only EODP perfectly addressing the CQ *Who holds the copyright of a photograph X?* is included in the *Agent* CC, which is an example of cluster grouping together many incoherent EODPs. 2/43 CQs are *not supported*: the property needed to address the CQ *Which interventions have been proposed?* is included in a *bad* community, and this community ended up in a cluster, named *Locale*, that actually groups together incoherent EODPs.

In conclusion, we can say that our catalogue of CCs would provide an important support to retrieve the ontology fragments from ArCo addressing more than 65% of the representative CQs of ArCo, and it would provide partial (*medium*) support for 2 additional competency questions. For 11 CQs, good EODPs addressing them can be found, but in a wrong CC. Finally, for the remaining 2 CQs, it would be difficult for a user to quickly find an appropriate ArCo's EODP(s) answering them through the catalogue.

Comparison between our empirical ODPs and ArCo's top-down ODPs. As a documentation activity related to the pattern-based methodology adopted, ArCo's ontology designers have manually catalogued, through graphical diagrams, the main ontology design patterns implemented in the ontology network. We compare these top-down defined patterns with the EODPs that we extracted with our method from ArCo ontology network[26], in order to understand how much the EODPs

[26]The results of this comparison can be found here: https://github.com/stlab-istc-cnr/conceptual-components/blob/main/conceptual-components-extraction/results/

that we can automatically find match the actual intent of the ontology designers[27].

ArCo's ontology design team defined 43 relevant ontology design patterns in the version 1.0 of the ontology network. Since we included ArCo 1.0 in our CH corpus, we discarded 11 additional patterns included in later versions of the ontology.

These 43 patterns map to 110 distinct communities extracted from ArCo. On average, one top-down defined pattern corresponds to 3 communities (max: 8; min: 1). Indeed, it can be reported as a first observation that these top-down ODPs tend to be bigger than our communities, in some cases grouping together patterns that seem to have been combined: for instance, the top-down ODP *acquisition* maps to 2 EODPs, one including the *acquisition* situation and its involved entities (the cultural property, the previous owner, the location, the time), the other including the relation between the *acquisition* situation and its *acquisition type* (a possible specialisation of a general *type* pattern).

We label as *positive* (P) an EODP whose entities (classes and properties) are part of a single ArCo's top-down ODP, as *negative* (N) an EODP including entities that are reported as part of separate patterns from ArCo's ontology design team[28]. Out of the 110 EODPs corresponding to ArCo's main patterns, 95 can be classified as positive, and the remaining 15 as negative. An example of negative EODP is one including the relationship between a cultural property and the collection it is member of, along with the relationship between a cultural property and its estimate, while the ontology design team defined two separate ODPs, one for the estimate, one for the membership to a collection. 24/43 top-down patterns match to only positive communities. Instead, only 2/43 top-down patterns match to only negative communities.

In some cases, we include in our EODPs additional classes and properties that

communities-manual-evaluation/evaluation-ArCo-patterns-EODPs.xlsx

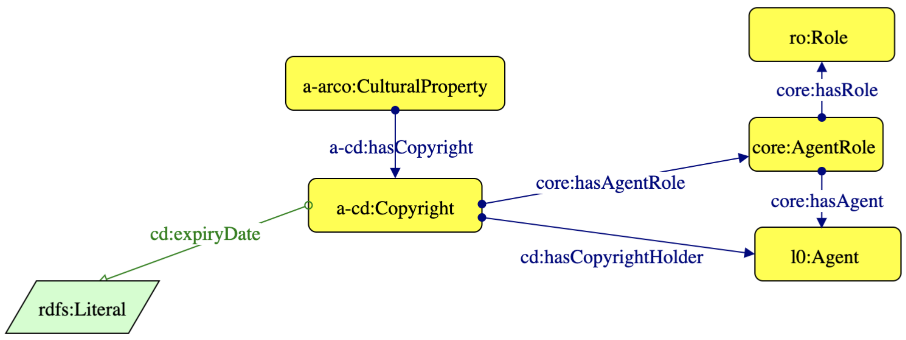
[27]The diagrams of the top-down defined ODPs of ArCo, along with the communities mapped to them, can be found here: https://drive.google.com/drive/folders/1_2SsvtkyEoauv9HaAjQ71t31TvYFFrxC?usp=sharing

[28]We are designing an additional evaluation that, rather than using a binary classification of EODPs, computes their percentage of coverage of classes and properties in the top-down ODPs.

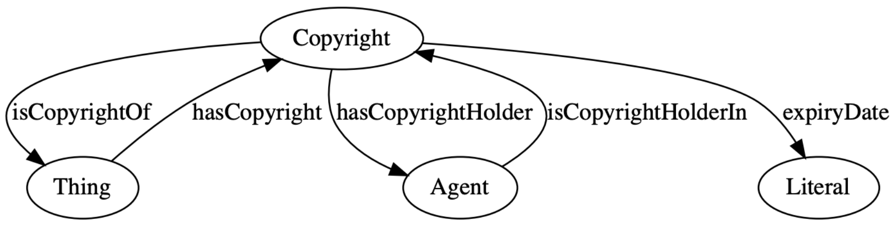
are not mentioned in ArCo's top-down pattern diagrams: for example, the diagram for *authorship attribution* does not include subclasses of `AuthorshipAttribution` (`CulturalScope`, `PreferredAuthorshipAttribution`, and `AlternativeAuthorship Attribution`), that correspond to 3 different communities, nor the property `has AuthorityFileCataloguingAgency`. Moreover, ArCo's top-down pattern diagrams tend to overlook inverse properties.

It may also happen that our communities do not include properties and classes that are depicted in ArCo's patterns, however, this is always due to missing axioms in the ontology. For example, the *use* pattern is depicted in one diagram, and corresponds to 6 different EODPs. The diagram also includes the property `hasUser`, which links the core class `Use` to the class `Agent`. However, unlike other properties, the property `hasUser` is not included in any property restriction on the class `Use`, and has `owl:Thing` as domain. Therefore, by considering the axioms in the ontology, it would not be possible to automatically include the `hasUse` property in the *use* EODPs: indeed, this property ends up in a *bad* community including a number of properties with `owl:Thing` as domain and `Agent` as range.

Let us consider an additional example, the *copyright* pattern. The top-down ODP (Figure 3.6a) corresponds to exactly one extracted EODP (Figure 3.6b). We include in our EODP 3 out of a total of 6 properties included in the top-down ODP, that is `hasCopyright`, `hasCopyrightHolder`, and `expiryDate`. However, the `hasAgentRole` property, that we do not detect, is not included in any property restriction on the class `Copyright`, and has `owl:Thing` as domain, so it is not surprising we could not be able to include it. As for the other 2 properties we do not detect (`hasAgent`, and `hasRole`), they are actually part of another top-down pattern, *agent role*. While in 3.6a the property `hasCopyright` links the class `CulturalProperty` to `Copyright`, there is no property restriction on `CulturalProperty` with the `hasCopyright` property, so in 3.6b `hasCopyright` properly links `owl:Thing` to `Copyright`, based on the domain axiom on the property. Finally, as it can be noticed, while 3.6a does not depict any inverse property, we also include all inverse properties, that is `isCopyrightOf`, and `isCopyrightHolderIn`.



(a) Top-down diagram of the ArCo's Copyright ODP.



(b) Automatically extracted EODP corresponding to Copyright ODP.

Figure 3.6: Top-down ODP and Empirical ODP about copyright from ArCo ontology network.

As is made clear in the examples, this comparison of classes and properties included in top-down patterns and our EODPs can support possible suggestions about missing axioms to add to the core classes of the patterns. In some cases, the detected EODPs may even highlight errors or oversights when manually creating the graphical diagram. For example, in the diagram of the *protective measure* ODP the properties `noticeDate` and `openingNoticeDate` have `rdfs:Literal` as range, while in our corresponding EODP they have the range `xsd:dateTime`, as in the source of the ontology. Hence, we hypothesise that our method could also support the evaluation and refactoring of an ontology.

Chapter 4

Annotating patterns in ontologies and knowledge graphs

A necessary next step to the extraction of patterns and conceptual components from ontologies, and the consequent identification of portions of knowledge graphs that populate those patterns, would consist in keeping track of the patterns extracted, so that the presence of a specific pattern is made explicit, and can effectively contribute to the interoperability between ontologies and knowledge graphs, supporting relevant tasks such as ontology selection and reuse. An ontology annotated with the ontology design patterns included in it – regardless of whether they have been intentionally developed as patterns, or they are the result of a pattern-unaware modelling and have been later extracted with a specific method – can be more easily explored through its patterns. These patterns somehow *summarise* the way the ontology addresses some modelling issues, and can be an input to ontology visualisation tools. Similarly, knowledge graph exploration and visualization can benefit from the annotation of *instances* of patterns. Moreover, when annotated with patterns, ontologies can be aligned (thus, can be made interoperable) based on the patterns they implement, and this annotation and alignment can happen at 3 different levels. It can be annotated the ontology design pattern implemented inside the ontology, as a set of classes, properties and axioms that propose a solution to a modelling problem. Secondly, an ontology designer may annotate an ontology with all the *facts* that ontology addresses (the conceptual components), irrespective of specific

RDF/OWL implementations as ODPs, that is annotating the ontology, and making it comparable with other ontologies, at a more abstract level. Finally, pattern-based interoperability can be reached even at the data level, by annotating the knowledge graph with the sets of triples that altogether populate the patterns of the ontology the KG is based on, enabling e.g. a visualization of the KG with a modular view that would ease its inspection.

In this Chapter we present a language for annotating ontology design patterns in ontologies and knowledge graphs.

4.1 Motivation

There are three main motivations to the development of a comprehensive language for a pattern-based annotation of ontologies and knowledge graphs, that we present in this Chapter.

Pattern. As already mentioned in previous chapters, a pattern-based ontology engineering is based on the creation or reuse of ontology design patterns, which are integrated in an ontology as small reusable components, following e.g. eXtreme Design methodology [12, 10]. An ontology can be modelled following a pattern-based approach since the start of the ontology project, or it can be refactored by trying to reuse existing ODPs. As previously stated, patterns can be implemented in an ontology even implicitly, e.g. by defining relations and classes that are compatible with an existing pattern. Annotating patterns (re)used within an ontology eases the process of understanding and exploring, and possibly reusing, an ontology, by making it explicit which sets of ontology entities are members of an ODP that addresses a specific modelling issue, and by allowing to represent hierarchical and other types of relations between related ODPs. Moreover, annotations can improve the interoperability between ontologies reusing the same patterns (or generalisations/specialisations of the same patterns), when these patterns are annotated and aligned.

Conceptual component. An ODP is a particular implementation of a modelling solution to a modelling problem, in the form of a small RDF/OWL ontology. An ODP can be easily reused and integrated in a larger ontology that needs to address the same modelling problem the pattern provides a solution for. However, there may exist multiple ODPs that, in different ways, address the same modelling problem, i.e. they are different solutions to the same modelling issue. As proposed in the previous Chapter, this is where the idea of conceptual component comes into play. We report again the example presented in the previous Chapter. Let us consider the modelling problem "being a member of a collection": it can be implemented as (i) a binary relation `hasMember` between an `Object` and a `Collection`, or (ii) as an n-ary relation `Membership` between the arguments `Collection`, `Time`, and `Object` (see Figure 3.1). Therefore, (i) and (ii) are two ODPs that address, in different ways and with different levels of expressiveness, the modelling issue of an object that is member of a collection. An ontology O_1 may reuse the ODP in (i), while an ontology O_2 may reuse the ODP in (ii): even if reusing different implementations, the two ontologies are addressing the same modelling problem. An ontology can be seen as a composition of conceptual components. If we annotate an ontology with the conceptual components it addresses, and we relate the RDF/OWL implementations to the respective conceptual components they implement, we may ease the exploration of, and the interoperability between, ontologies at a more abstract level, i.e. regardless of actual implementations.

Pattern instance. If one (or more than one) knowledge graph is built by reusing an ontology, this knowledge graph will contain triples that comply with that ontology – along with other possible reused ontologies. If this ontology is pattern-based, and includes a number of ontology design patterns, the knowledge graph may contain instances of that pattern. A *pattern instance* is a collection of ABox triples, including individuals and properties that comply with an ontology design pattern. Let us consider the pattern *Collection*[1]. This ODP defines two inverse properties (`hasMember` and `isMemberOf`) that have the classes `Collection` and `Thing` as

[1]http://ontologydesignpatterns.org/wiki/Submissions:Collection

domain or range. An example of instance of this ODP would be the set of the following 4 triples: (i) `:collection_X :hasMember :member_A`, (ii) `:collection_X :hasMember :member_B`, (iii) `:member_A :isMemberOf :collection_X`, and (iv) `:member_B :isMemberOf :collection_X`. Having pattern instances annotated with respect to the ODP that they comply with, would allow us to support a pattern-based exploration and visualization of a knowledge graph.

4.2 Related work

OPLa. [53] introduces a simple and extendable language for annotating information about ontology design patterns (ODPs) and the corresponding modelling process, with OWL annotation properties. With this language it is possible to indicate the patterns that an ontology (module) reuses, the ontology entities (classes, properties, individuals and axioms) that belong to a pattern (or ontology module); the possible modules an ontology can consist of; specialization or generalization relations between patterns and modules. Specifically, this annotation ontology defines the class `OntologicalEntity`, and its subclasses (namely, `Individual`, `Property`, `Class`, `Axiom`). The property `isNativeTo` links an ontological entity to the `OntologicalCollection` it is member of and is a core entity of. The classes `Ontology`, `Module` (intended as a part of an ontology that captures a conceptual sub-area of the domain) and `Pattern` are subclasses of `OntologicalCollection`. A module can be declared as native to an ontology, and an ontological collection can be annotated with the property `reusesPatternAsTemplate` for indicating the ODPs that have been reused in it. Finally, additional relations between patterns and modules are modelled with properties such as `hasRelatedPattern`, `generalizationOfPattern`, or `specializationOfModule`.

CP annotation schema. The *ODP Portal*[2] is a catalogue that aims at collecting Ontology Design Patterns. Specifically, the *CP annotation schema*[3] has been

[2]http://www.ontologydesignpatterns.org

[3]http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl

defined to annotate a Content Pattern, i.e. an ODP addressing content modelling problems, in order for the pattern to be submitted to the ODP Portal. Similarly to OPLa, this ontology makes use of OWL annotation properties to feed the information fields of each catalogue entry, that can also be exploited by Semantic Web applications [37], and allows an ontology designer to describe an ODP by specifying: (i) its intent, i.e. the overall scope of the ontology pattern; (ii) the requirement(s) addressed; (iii) related scenarios, intended as examples of instantiation of the pattern; (iv) its consequences, i.e. benefits or trade-offs in using the pattern; (v) relations between patterns (specialization, generalization, componency); (vi) requirement-based unit tests that have been run to evaluate it; (vii) ontologies or other schemas that the pattern was extracted or reengineered from.

In their current state, neither OPLa nor the CP annotation schema are able to represent the relations between a pattern and its abstract counterpart (conceptual component), and between a pattern and its instance in a KG (pattern instance).

[50] introduces some small changes and extensions to OPLa and CP annotation schema, reorganising them into three different namespaces[4]: `opla-core` for storing the OPLa original annotation properties and classes; `opla-cp`, that is an adaptation of the CP annotation schema; `opla-sd`, including new annotation properties possibly needed by tools supporting modular graphical ontology modelling (specifically, coordinates of a node in a schema diagram).

GO-FOR. The authors of [77] introduce GO-FOR, the “Goal-Oriented Framework for Ontology Reuse”, aimed at supporting a pattern- and goal-based reuse of ontologies. Its core entity is a goal-oriented ontology pattern (GOOP): an ontology fragment that is bound to a goal, that is the scope addressed by the pattern. A repository integrated in the GOOP-HUB[5] stores existing GOOPs. Based on their goals, GOOPs can be related through *part-of* relationships.

The GOOP OWL metamodel[6], besides deriving from OWL the `Class`, `Object`

[4]https://github.com/cogan-shimizu/Extended-OPLa

[5]https://github.com/nemo-ufes/goophub

[6]https://github.com/nemo-ufes/goophub/blob/master/src/main/resources/goop-meta-model.owl

`Property` and `DatatypeProperty` concepts for representing the constructs that a GOOP can consist of, models the class `Goal`, which has `AtomicGoal` and `ComplexGoal` as subclasses. A complex goal consists of other goals by either `OR_decomposition` (at least one subgoal is to be addressed) or `AND_decomposition` (all subgoals need to be satisfied). Moreover, a goal is related to the `Actor` that aims to achieve it. The concept of *goal* of a pattern in GO-FOR can be aligned to the concept of *intent* in the CP annotation schema. However, while in GO-FOR *part-of* relations are defined at the level of the goals (the designer can specify subgoals of complex goals), in both OPLa and CP annotation schema the composition, specialization and generalization relations are expressed at the level of patterns. Search for patterns in GOOPR can be based on goals and specific actors related to them (e.g. a doctor, a researcher), while the patterns of the ODP Portal are grouped based on the domain they address (e.g. multimedia, time). Like the previous ones, this metamodel has the limit of focusing only on the pattern level.

OTTR. In [81] the Reasonable Ontology Templates (OTTRs) are described: OTTRs are OWL ontology macros through which it is possible to define and instantiate ontology design patterns. An OTTR T is a parametrised ontology that can be instantiated providing arguments that fit the parameters of the template. The OTTR T is formalised as a knowledge base O_T together with a list of parameters $(p_1, ..., p_n)$ of concepts, roles or individuals from O_T. Given a list $(q_1, ..., q_n)$ of constants, concepts or role expressions that are called arguments, $T(q_1, ..., q_n)$ represents a template instance, that is an occurrence of a pattern. OTTRs can also be specialised for specific use cases and sub-domains. Moreover, specific templates can be used for generating instances of a modelled pattern. OTTRs are expressed in a specific syntax (stOTTR), however the authors developed a tool[7] that converts OTTR templates into OWL ontologies. These templates' aim is to automatize ODP-based ontology engineering and support interoperability between ontologies that use the same or related templates. Moreover, the concept of pattern instance is also used. However, as a drawback, the definition and maintenance of these templates may

[7]https://gitlab.com/ottr/lutra/lutra

result to be expensive, and to hardly support changes in already defined ODPs.

4.3 OPLaX

OPLaX[8,9] (Ontology Pattern Language eXtended) is the annotation ontology we developed, with the aim of providing the ontology designers with a language for annotating ontology design patterns at both pattern, conceptual component, and instance level (see Figure 4.1). OPLaX reuses and is aligned with OPLa (see the plain classes and properties in the figure) and with CP annotation schema (see the properties in italics). OPLaX introduces some changes with respect to these two reused ontologies, and integrates them with specific classes and properties (see the bold classes and properties in the figure) for modelling data related to conceptual components and pattern instances.

4.3.1 Pattern level

OPLaX reuses from OPLa classes and properties that allow the ontology pattern designer to define more specific types of `:OntologicalCollection`, that is `:Module`, `:Ontology`, and `:Pattern`. Moreover, patterns can be related with their member ontological entities with the `:isNativeTo` property, and it is possible to specify componency (`:componentOfPattern/Module`), derivation (`:derivedFromPattern/Module`), specialization (`:specializationOfPattern/Module`), and generalization (`:generalizationOfPattern/Module`) relations between ontological collections of the same or different types. Moreover, it can be annotated that an `:Individual`, a `:Class` or a `:Property`, that is out of scope of that particular pattern, has been used (property `:ofExternalType`). The property `:reusesAsTemplate` indicates that an ontological collection reuses as templates other ontological collections, e.g. an ontology that reuses multiple patterns as templates. Additional properties have been reengineered from the CP annotation schema: `:addressesScenario`

[8]https://w3id.org/OPLaX/
[9]https://github.com/stlab-istc-cnr/OPLaX

defines the scenario; `:hasIntent` annotates the intent; the requirement(s) covered are indicated with `:coversRequirement`; the consequence through the property `:hasConsequence` and the competency question(s) with `:hasCompetencyQuestion`. To annotate the ontological collection with possible unit tests the property `:hasUnit Test` can be used. Moreover, it is possible to explicitly annotate that an ontological collection is `:extractedFrom` another ontological collection, e.g. a pattern that has been widely or partially cloned by an ontology.

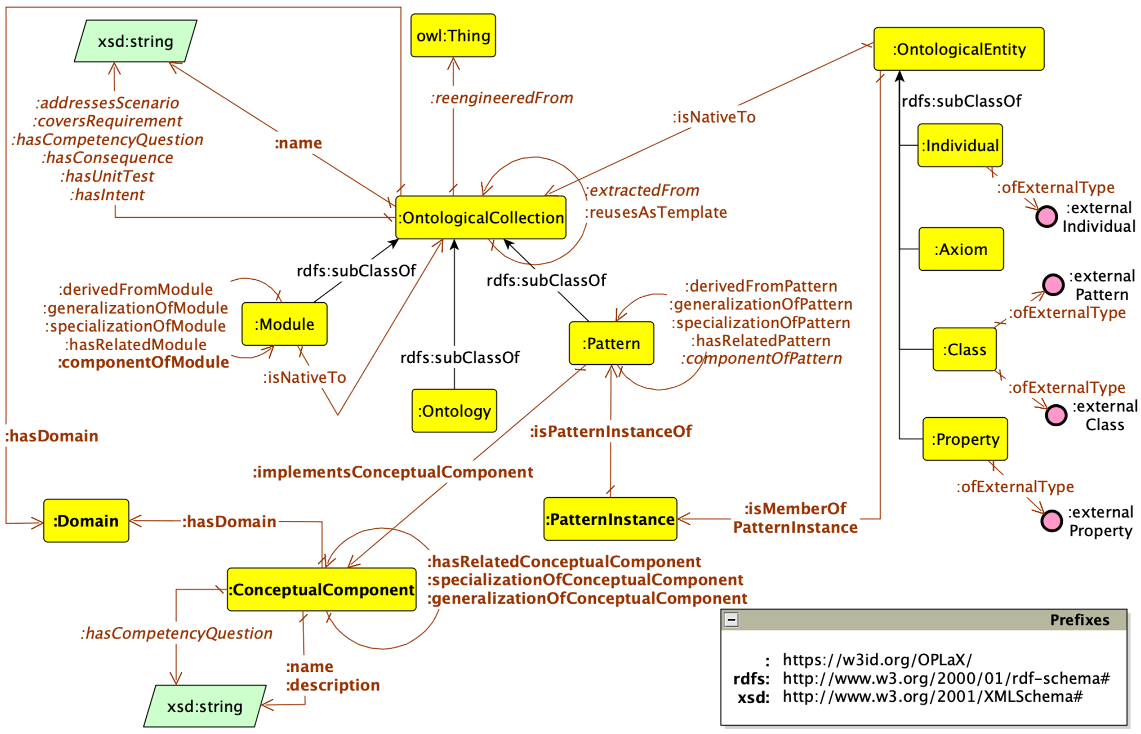


Figure 4.1: OPLaX, the Ontology Pattern Language eXtended.

4.3.2 Conceptual component level

In OPLaX, a `:Pattern` can be related to the `:ConceptualComponent` it is an implementation of with the OWL annotation property `:implementsConceptual Component`. Multiple conceptual components can be related to each other with `:has RelatedConceptualComponent`. For example, the conceptual component 'a cultural property that is located at a cultural site' would be a specialization of the more general conceptual component 'an object being located at a place' (`:specializationOfConceptualComponent`), and the other way around (`:generalizationOfConceptualComponent`). A conceptual component can be annotated with its `:name` (e.g. *locating* for the conceptual component 'an object being located at a place'), and with a more detailed description (`:description`). Moreover, a conceptual component satisfies one (or more than one) general competency question (`:hasCompetencyQuestion`), e.g. *where is an object?* for the previously mentioned component *locating*. Finally, a conceptual component has a `:Domain` (`:hasDomain`): for instance, a conceptual component *Paper award* and a conceptual component *Submitting paper* (as from the experiments presented in the previous Chapter) could be included in the *Conference* domain.

4.3.3 Pattern instance level

An instance of a `:Pattern`, i.e. an entity member of the class `:PatternInstance`, is related to the pattern it is an instance of with the annotation property `:isPatternInstanceOf`. All the individuals that are member of a pattern instance are annotated with the property `:isMemberOfPatternInstance`. By retrieving all individuals linked to a pattern instance by means of this property, it is possible to obtain the boundary of the pattern instance itself.

4.4 Use cases

In this section, we show how OPLaX has been used in practice in three use cases: the ArCo ontology (described in 2.2.2), the catalogue of conceptual components and ontology design patterns extracted from the cultural heritage corpus (see the previous Chapter), and an external work, i.e. a pattern-based visualization of knowledge graphs [5] which uses ArCo as a use case too.

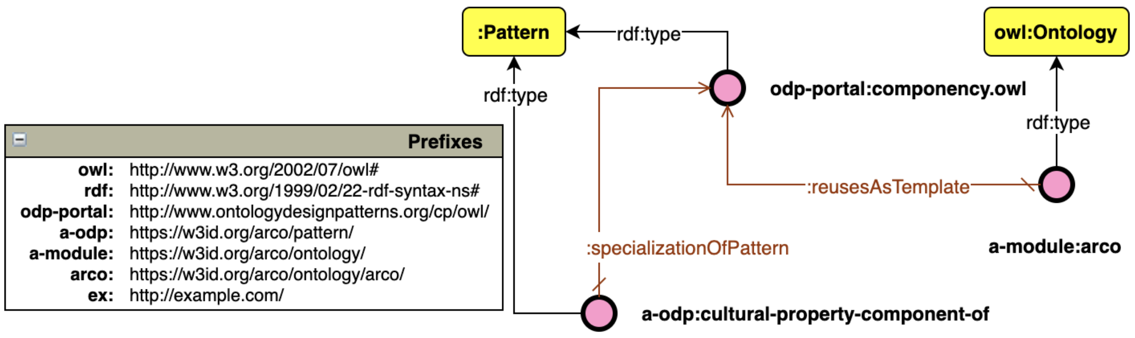
4.4.1 ArCo ontology network

As reported in Section 2.2.2, ArCo [19, 20] has been developed following the pattern-based eXtreme Design (XD) ontology engineering methodology [12, 10]. ArCo version 1.0 is composed of 7 ontology modules, across which 12 different ODPs published in the ODP Portal have been reused and specialised. ArCo directly reuses, by a direct embedding of the ontology entities in the local ontology, only two ontologies: indeed, these ontologies are considered reference standards by the Italian Government and their development involves ArCo's team. Instead, other ontologies and ontology design patterns are indirectly reused, that is they are used as templates, reproduced (and in some cases extended/specialised) in the local ontology, and aligned with `rdfs:subClassOf/subPropertyOf` and `owl:equivalentClass/ equivalentProperty` axioms [18]. These alignment axioms support ontology and knowledge graphs interoperability, by making it explicit a correspondence between single ontology entities.

With OPLaX it is possible to annotate all the ontology entities as members of an ODP, and generate alignments between different ontologies at the level of patterns. An example – taken from ArCo – of ODP specialization, annotated through OPLaX, is presented in Figure 4.2. The *arco* ontology module, that is the root module of the network (see Section 2.2.2.3), indirectly reuses the ODP *Componency*[10], published on the ODP Portal: the whole module is indeed annotated with the property `:reusesAsTemplate` for representing the reuse of the pattern. Specifically, the ArCo's pattern *Cultural Property Component of*[11], included in the arco module, specialises the pattern *Componency*, since it represents the componency relation between a complex cultural property and its components. Hence, the pattern is annotated with the property `:specializationOfPattern` (see Figure 4.2a). For expressing that individual properties (e.g. `arco:hasCulturalPropertyComponent`) and classes (e.g. `arco:ComplexCulturalProperty`) implemented in the module belong to this specialised ODP, the annotation property `:isNativeTo` is used (as in

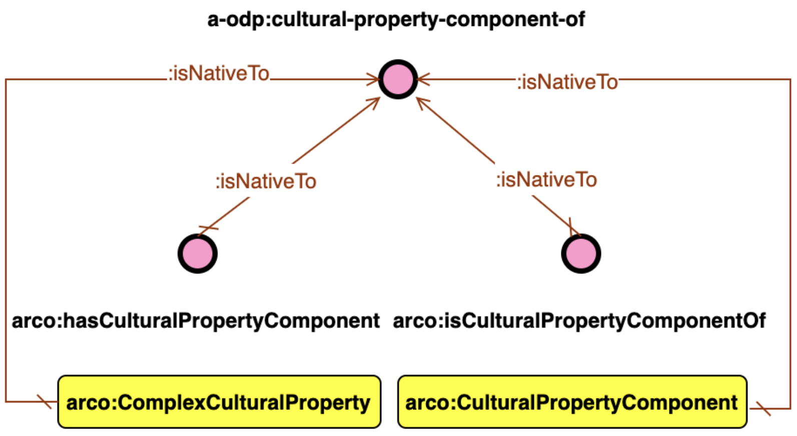
[10]http://ontologydesignpatterns.org/wiki/Submissions:Componency

[11]https://w3id.org/arco/pattern/cultural-property-component-of



Prefixes	
owl:	http://www.w3.org/2002/07/owl#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
odp-portal:	http://www.ontologydesignpatterns.org/cp/owl/
a-odp:	https://w3id.org/arco/pattern/
a-module:	https://w3id.org/arco/ontology/
arco:	https://w3id.org/arco/ontology/arco/
ex:	http://example.com/

(a) The property `:reusesAsTemplate` relates the *arco* module to the *Componency* ODP reused over the module. The property `:specializationOfPattern` expresses the specialization relation between the pattern *Cultural Property Component of* implemented in the module and the ODP *Componency*.


a-odp:cultural-property-component-of
:isNativeTo
:isNativeTo
:isNativeTo
:isNativeTo
arco:hasCulturalPropertyComponent
arco:isCulturalPropertyComponentOf
arco:ComplexCulturalProperty
arco:CulturalPropertyComponent

(b) The annotation property `:isNativeTo` relates the object properties and the classes of the *Cultural Property Component of* ODP to the ODP itself.

Figure 4.2: An example of a specialized ODP annotated with OPLaX.

Figure 4.2b).

4.4.2 Conceptual components and ODPs catalogue from a corpus of ontologies

Let us take as an example the catalogue generated from the corpus of 43 ontologies on Cultural Heritage[12], that we presented in Chapter 3. The conceptual compon-

[12]https://stlab-istc-cnr.github.io/cc-and-odps-catalogue/

ent named (annotation property `:name`) *Event* (Figure 4.3), which represents the general modelling problem of the happening of an event, is implemented by 21 observed ontology design patterns, extracted from 13 distinct ontologies. Each observed ontology design pattern is thus related to the CC *Event* with the property `:implementsConceptualComponent`. These patterns implement the general component at different levels of specialization, thus addressing also more specific intents: for instance, an ODP extracted from the Europeana Data Model (see the ODP at the top of Figure 4.3) represents the general concept of an `Event` that `happenedAt` some `Place`, and `occurredAt` a certain `TimeSpan`, while the event modelled by an ODP extracted from Cultural-ON (see the ODP at the bottom of Figure 4.3) is a more specific type of event, that is a `CulturalEvent`, which involves at least one cultural entity. This conceptual component is also associated with a description (with the property `:description`), which, in the catalogue's case, is generated by concatenating all the labels that annotate the entities of the patterns. A manually generated competency question that could be associated with this conceptual component by the property `:hasCompetencyQuestion` would be "What happened?". The conceptual components are included in a hierarchical network, that is built based on the inheritance relations between the frames that are detected from the virtual documents of the individual patterns. For instance, the CC *Event* in Figure 4.3 is related through the property `:generalizationOfConceptualComponent` to the conceptual component *Intentionally act.*

4.4.3 Pattern-based visualization of knowledge graphs

In [5], the authors present a novel pattern-based approach for visualising a knowledge graph based on the ontology design patterns it includes. ODPs become the key access point for the exploration of the knowledge graph: based on them, possible thematic paths can be built, for guiding the exploration of, and interaction with, the KG. Moreover, the user can exploit the set of the patterns that are instantiated in the KG as a concise summary of its content. The presented approach relates a visual frame, that is an intuitive standard visualisation, to an ontology design

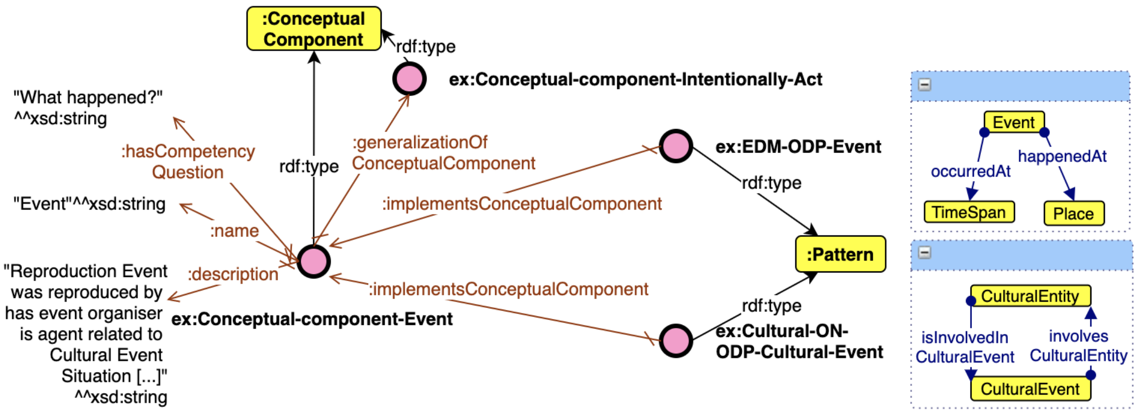

:Conceptual Component
rdf:type
ex:Conceptual-component-Intentionally-Act
"What happened?" ^^xsd:string
:hasCompetency Question
rdf:type
:generalizationOf ConceptualComponent
ex:EDM-ODP-Event
rdf:type
:implementsConceptualComponent
"Event"^^xsd:string
:name
:Pattern
"Reproduction Event was reproduced by has event organiser is agent related to Cultural Event Situation [...]" ^^xsd:string
:description
:implementsConceptualComponent
ex:Conceptual-component-Event
rdf:type
ex:Cultural-ON-ODP-Cultural-Event
Event
occurredAt
happenedAt
TimeSpan
Place
CulturalEntity
isInvolvedIn CulturalEvent
involves CulturalEntity
CulturalEvent

Figure 4.3: An example of a conceptual component annotated with OPLaX.

pattern, such that every time an ODP is used in an ontology, the data of a KG can be visualized thanks to that reusable visual frame. There are three levels of exploration. At the first level (*ODP level*), the user can access the patterns that are populated in the knowledge graph, viewing their related visual frames, along with the most important concepts (key concepts) in the KG, based on the number of occurrences. At the second level (*exploration level*), the user can view all the instances of a specific pattern in the KG, and can filter them based on predefined semantic filters specific to the ODP in question. The third level (*visualization level*) displays a single instance of an ODP that has been selected by the user.

The tool developed[13] in the paper [5] uses OPLaX annotation properties. In particular, the *ODP level* relies on OPLaX annotations that are related to the pattern level, in order to display the patterns in the graphical user interface. If any, it is possible to view specialisation and composition relations between patterns, thanks to the annotation properties `:specializationOfPattern` and `:componentOfPattern`. The `:isNativeTo` property is reused to annotate the relation between a pattern and its key concept(s). At the *exploration level*, the tool relies on the annotation property `:isPatternInstanceOf` for linking a pattern instance to the pattern that is being instantiated. Thanks to this annotation, the user can access the list of all the instances of a pattern, and then browse it for possibly selecting the instances

[13]https://github.com/ODPReactor

of interest, using predefined filters. The property `:isMemberOfPatternInstance` allows to link all the entities that are member of a pattern instance to the pattern instance itself, so that, at the *visualisation level*, an individual pattern instance, with its members, can be visualised by means of a graphical component associated with the given pattern.

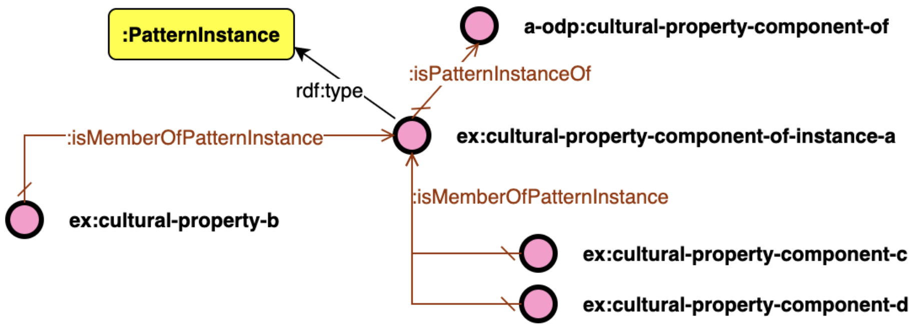

:PatternInstance
a-odp:cultural-property-component-of
:isPatternInstanceOf
rdf:type
:isMemberOfPatternInstance
ex:cultural-property-component-of-instance-a
ex:cultural-property-b
:isMemberOfPatternInstance
ex:cultural-property-component-c
ex:cultural-property-component-d

Figure 4.4: An example of a pattern instance annotated with OPLaX.

As graphically represented in the example of Figure 4.4, `ex:cultural-property-component-of-instance-a` is an instance of the ArCo's ODP *Cultural Property Component of*: indeed, it is related to the pattern with the property `:isPatternInstanceOf`. The property `:isMemberOfPatternInstance` annotates each member of this pattern instance, that is: the complex cultural property (`ex:cultural-property-b`) and its two components, i.e. `ex:cultural-property-component-c` and `ex:cultural-property-component-d`.

Chapter 5

Observing patterns from data

Most times, ontologies are developed top-down, starting from some application needs, requirements coming from domain experts, requirements extracted from a non-LOD dataset, etc. Then, as a separate and following step, knowledge graphs populate those pre-defined and formally specified ontologies. However, this is not always the case. Some knowledge graphs include triples that make use of properties and instantiate classes, but those classes and properties are not formally defined in an ontology, or their definition is quite incomplete and shallow. Therefore, the use of such classes and properties is subject to the interpretation of the knowledge graph developers that generate the data, possibly aided by some informal definition/documentation/guidelines. When an explicitly and well defined ontology is missing, it becomes even more difficult for an external user to understand a knowledge graph, to reuse its data, or to reuse its underlying semantics (classes, properties) for another use case. Similarly to the case of ontology understanding and reuse, the progressive analysis of the content of a KG could have as goals (i) understanding its structure and nature, (ii) identifying whether the KG can satisfy some user needs/questions, and (iii) finding the exact set of triples pertinent to the use case [61]. In this context, it becomes crucial to exploit the data level, and all the pieces of information we can derive from it, in order to *infer* some sets of axioms or constraints, i.e. building blocks of a *background* ontology empirically extracted from the data. Contrary to most cases, this ontology would be (partially) built bottom-up, and would reflect the usage of classes and properties in actual KGs, representing an access point to the content of the knowledge graph, and possibly supporting its reuse.

In this Chapter, we present a method for extracting empirical ontology design patterns that *emerge* from a knowledge graph. These patterns are initially built in the form of domain-property-range triplets, accompanied by their usage statistics, and are then translated into (i) owl-compliant ontology design patterns, as sets of axioms to which a probability is assigned based on the usage in the KG, and (ii) shapes, as sets of constraints accompanied by their probability. A shape is a set of conditions, also called constraints, usually used for validating a knowledge graph, but also providing a *description* of a certain graph complying with those conditions.

Moreover, using the Wikidata knowledge graph as a use case, we show how these empirical patterns can provide additional information about the content of a domain-specific portion of the knowledge graph, information that is not available from existing guidelines on the usage of the semi-formally defined ontology the Wikidata KG is built upon.

5.1 Method for extracting empirical patterns from a KG

5.1.1 Related work

There exist many approaches to generate constraints for concepts, in the form of shapes or patterns, in order to mainly support knowledge graph validation, but possibly useful also for its exploration.

Some of them (like Astrea [26]) are only based on ontologies. Astrea[1] [26] is a tool that automatically generates SHACL[2] shapes from a collection of ontologies by executing a set of mappings between ontology constraint patterns and SHACL constraint patterns, encoded in the Astrea-KG. However, Astrea does not take into

[1]https://github.com/oeg-upm/astrea

[2]https://www.w3.org/TR/shacl/

account the data level. [57] compares various aspects of OWL and SHACL, and useful mappings between the two languages are provided. However, most methods focus on generating shapes from a knowledge graph. Shape Designer [15] is a graphical tool for the automatic construction of valid SHACL or ShEx constraints that are satisfied by an RDF dataset. The cardinality of the triple constraints (*exactly one*, *optional*, *at least one*, *any number*) is inferred from the data based on some rules. However, a limit to the number of the SPARQL query results needs to be specified, when working with large KGs such as Wikidata. [76] performs some experiments that show that the existing methods cannot handle the scale of large knowledge graphs like Wikidata, indeed they crash even with KGs with a few millions triples[3]. sheXer [31] is an automatic shape extractor that extracts shapes, which are serialised in both ShEx and SHACL, by mining the graph structure and exploring the neighborhood of specific target nodes. All constraints are associated with a trustworthiness score, which allows to filter constraints based on their frequency and to sort and merge the constraints that are inferred in order to build the final shapes. Some methods exploit knowledge graph profiling techniques: the concise and meaningful summaries generated from RDF knowledge graphs are used as input for building shapes. [65] describes a data-driven approach that uses machine learning techniques for automatically generating RDF shapes, as collections of validation rules. Profiled RDF data is used as features, and the method relies on the Loupe tool[4] [64], which provides information about the frequency of triple patterns (in the form $\langle subjectType, predicate, objectType \rangle$) that appear in a dataset. Even if the approach can be extended to other types of constraints, in this work shapes are generated at the class level (e.g. a shape for validating instances of the class `dbo:Person`), and can combine only two constraint types, i.e. cardinality and range constraints, by analyzing data patterns and statistics. In [82], the profiles generated by ABSTAT are converted into SHACL shapes around a specified target

[3]The subKG of Wikidata on the music domain, to which we applied our method, contains more than 5 millions triples.

[4]http://loupe.linkeddata.es/

class; then, a human user can update and correct automatically generated shapes. ABSTAT [83, 1] is a profiling tool that generates a *semantic profile*, starting from a KG and, if any, an ontology that is used in the KG. The resulting profile is composed of what they name *Abstract Knowledge Patterns* (AKPs), that are associated with their occurrences, where *subjectType* is the most specific type of the subject and *objectType* is the most specific type of the object, excluding more generic and redundant patterns by using the possible ontology.

As demonstrated by [76], all approaches that support the automatic generation of shapes build a high number of shape constraints such that it is non-trivial for a human user to check their validity. Moreover, in most cases no constraint is generated for non-literal objects, i.e. most constraints do not indicate that the objects of a property should be of a specific type.

The closest work to ours is described in [93, 14]: Statistical Knowledge Patterns (SKPs) are extracted from knowledge graphs through a method based on statistical measures similar to the ones usually used for generating data-driven shapes, that is related to the frequency in the data. An SKP is expressed in OWL and is constructed around one of the *main* (i.e. most populated) classes from an ontology: it summarises the usage of that class by enriching the properties and axioms involving the class from the ontology with properties and axioms that can be induced from statistical measures on the data. The most frequent (based on a threshold) properties in the data are selected, and the appropriate range(s) is/are specified, unless they are already explicitly asserted in the ontology. The catalogue with 34 SKPs extracted from a version of DBpedia is online[5]. However, the code of the method described in the paper is not publicly available, so it is not possible to reproduce their results, and the SKPs do not include any metadata about the actual usage of the selected properties in the KGs, such as the number of occurrences in the data.

[5]http://www.ontologydesignpatterns.org/skp/

5.1.2 Method

As mentioned above, our method extracts the empirical patterns from a knowledge graph, without relying on a formally defined ontology. For this reason, any axiom/constraint that is generated from a KG and is part of an empirical pattern is associated with a probability, due to the fact that it is strictly related to the specific KG being processed. The interpretation of probability we adopt is the *frequentist* one[6]: it defines the probability of an event as the limit of its relative frequency in many trials. In our case, for *trials* we mean the occurrences in the data. For instance, if we have a knowledge graph on the fishing domain with 100 instances, 100 will be our *trials*; if 55 of them are instances of the class *Fishing Rod*, the probability that an instance in the fishing domain is a fishing rod, based on that particular KG, is equal to 55/100, i.e. 55%. Moreover, each axiom/constraint is also annotated with the KG which it is derived from.

In the following paragraphs, we describe in detail our method for extracting empirical patterns from a knowledge graph. An overview of the method is shown in Figure 5.1.

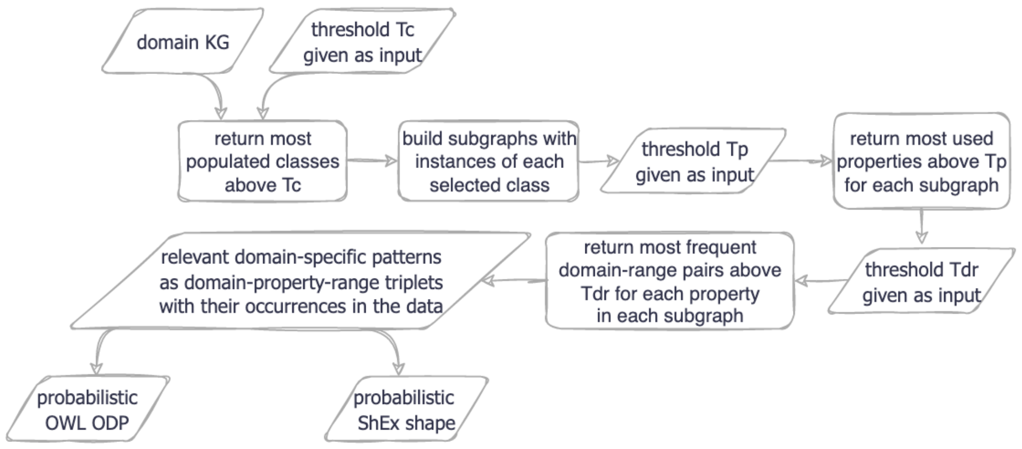

domain KG
threshold Tc given as input
return most populated classes above Tc
build subgraphs with instances of each selected class
threshold Tp given as input
return most used properties above Tp for each subgraph
threshold Tdr given as input
return most frequent domain-range pairs above Tdr for each property in each subgraph
relevant domain-specific patterns as domain-property-range triplets with their occurrences in the data
probabilistic OWL ODP
probabilistic ShEx shape

Figure 5.1: Method for extracting empirical patterns from a KG.

Select relevant classes from the domain subgraph. The first step of the method takes as input the domain subgraph and counts the number of instances

[6]https://en.wikipedia.org/wiki/Frequentist_probability

for each instantiated class of the graph. A percentage of coverage for each class is also computed, indicating its frequentist probability, as a simple ratio between the instances of a class and the total number of distinct instances in the subgraph. Then, a threshold is required to be given as input, and is used to filter out all the classes whose instances fall below it. Only the classes that have been selected based on this threshold will be used to generate the empirical patterns, i.e. at the end of the process there will be one pattern for each filtered class.

The threshold is based on the absolute distance between the number of instances of a specific class and the number of instances of the class that is most instantiated, that is the maximum number in the distribution. This distance undergoes a normalization, by dividing the result by the maximum value, so that the threshold falls within the range [0, 1]. If the threshold T_c is equal to 0, only the most instantiated class will be selected, since the distance between the count of the class in question and the maximum count must be smaller or equal to 0; on the contrary, if T_c is equal to 1, all classes with at least one instance in the KG will be considered: indeed, the distance between the count of a given class and the maximum count must be smaller or equal to the maximum count. Notice that our method does not define the best thresholds to be used: the value of the generated patterns is the incorporation of information about the frequentist probability; however, it is the user that, based on her requirements and use case, chooses the most appropriate thresholds.

Extract a subgraph for each selected class. Once we have the list of classes, we build a subgraph for each class, by picking from the domain subgraph only the triples with an instance of the given class as subject. For example, for the class *album*, we will build a subgraph including all the triples where an instance of album is in the subject position.

Most frequent properties for each class. At this point, we iterate over the subgraphs, and compute the number of occurrences of all the properties instantiated in each subgraph, i.e. we count the number of distinct instances that have at least one triple involving that property. Again, the frequentist probability, expressed as a percentage, indicates the ratio between the number of instances that have that property, and the total number of instances of the specific subgraph. Then, for each subgraph, we consider only the properties that are above a new threshold T_p given as input. Notice that we discard from this computation both the property expressing the type of an instance (e.g. `rdf:type` for RDF KGs, `wdt:P31`[7] for Wikidata KG) and the property used for expressing the hierarchy of classes (e.g. `rdfs:subClassOf` for RDF KGs, `wdt:P279` for Wikidata KG).

Most frequent ranges for each frequent property. For each subgraph, we compute all the domain-property-range triplets, where *domain* is the type of the subject and *range* is either the type of the object (when the object is a resource) or the data type (e.g. *rdfs:Literal* for RDF KGs). The number of occurrences of each triplet is then computed to find the most common domain-range pairs for each property in the graph. Again, a threshold T_{dr}, given as input, allows to filter only the most common domain-range pairs for any of the most common properties that have been selected in the previous step.

Empirical patterns: from triplets to probabilistic ODPs. At this point, we have computed the patterns in the form of sets of domain-property-range triplets associated with their number of occurrences and a probability value. In order to translate these *informal* patterns in OWL-compliant ontology design patterns, we transform each domain-property-range triplet into an OWL existential axiom. Each axiom is annotated with its frequentist probability with respect to the specific pattern. For this reason, in our implementation of the method, we express the patterns using rdf-star[8], which extends RDF with a convenient way to make statements about other statements, and we rely on the owl-star vocabulary[9] for encoding ontological assertions (axioms) as statements that can be annotated. Moreover, we also extend (and reuse) for this particular use case the OPLaX ontology (see Chapter 4), by introducing the class `oplax:FrequentistProbabilisticPattern`, as a subclass of `oplax:Pattern`. As a result, to each pattern will correspond a file containing all probabilistic axioms.

[7]`wdt:` http://www.wikidata.org/prop/direct/

[8]https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

[9]https://github.com/cmungall/owlstar/blob/master/owlstar.ttl

Empirical patterns: from triplets to shapes. Additionally, each set of triplets is transformed into a shape, associating each constraint with its probability value through comments. In our implementation of the method, we rely on the ShEx[10] schema language for expressing the shape.

5.2 Empirical patterns from Wikidata

Wikidata[11] is a huge knowledge graph that has been built collaboratively, and stores structured data for its Wikimedia sister projects, including Wikipedia and Wiktionary [91]. The collaborative editing activities on the Wikidata KG are performed on a daily basis; indeed, Wikidata contains a wide and rich set of factual statements about a huge number of different entities and events in the real world. Due to its frequent updates by its contributors and the possible changes in the way they model the data, the ontology underlying the knowledge graph is constantly subject to change.

Considering that the definition of the Wikidata ontology is bottom-up, often implicit, and undergoes a constant evolution, it can sometimes be challenging to effectively reuse the Wikidata ontology [72, 16]. While the Wikidata project does provide some flexible guidelines around use (see Section 5.2.1), there still remains room to provide additional and more detailed guidance on how to use the ontology based on its *actual usage*. For these reasons, the Wikidata KG worked as a perfect use case for our method.

5.2.1 Motivation

In this section, we provide some details about the various resources and approaches adopted by Wikidata for recommending how to use its underlying ontology.

[10]https://shex.io/

[11]https://www.wikidata.org/

Property constraints. Property constraints[12], as defined by the Wikidata community, are rules on a property that specify how the property should be used in the knowledge graph, with possible exceptions. Indeed, these rules have a degree of flexibility: they aim at guiding the Wikidata editor in injecting or editing (new) statements in the KG, providing possible useful suggestions. Their definition is informal, without an explicit logical specification, thus they can still be violated or ignored. There are several types of property constraints. Two popular property constraint types are the *subject type constraint* and the *value-type constraint*, which indicate that the domain or range of a property, respectively, should be one in a list of classes. However, unlike OWL property restrictions on classes, they do not limit the applicable classes. For example, a triple with an instance of *recurrent event edition* (`wd:Q27968055`) as subject, *part of the series* (`wdt:P179`) as predicate, and an instance of *collection of articles* (`wd:Q17518557`) as object would comply with the property constraints of the property `wdt:P179`, even if a more appropriate range in this case would be the class *recurring event* (`wd:Q15275719`).

Properties for this type. The property *properties for this type* (`wdt:P1963`) allows to list the properties recommended to be used for instances of a certain type. For example, *part of the series* is one of the recommended properties for instances of the type *recurrent event edition*, however the appropriate range(s) to be paired with that specific type (e.g. *recurring event*) cannot be specified.

Type of Wikidata property. The class *Type of Wikidata property* (`wd:Q107649491`) is a Wikidata metaclass, i.e. the instances of this class are other classes that are related to a specific set of items, domain or topic; the property *facet of* (`wdt:P1269`) expresses the relation between the metaclass and its topic. These classes are placed into a hierarchy, and their instances are properties. For instance, the property *Chessgames.com player ID* (`wdt:P1665`) is an instance of the class *Wikidata property related to chess* (`wd:Q27698571`) that is a subclass of *Wikidata property related to sport* (`wd:Q21818626`) , which includes e.g. the property *number of medals* (`P10659`) between its instances. However, this classification of properties is an activity in pro-

[12]https://wikidata.org/wiki/Help:Property_constraints_portal

gress, thus it is not close to be complete for some domains; moreover, properties that are actually relevant to be used for instances of a certain type may be excluded from the metaclass specific to that set of items because they are declared as relevant only for more general domains.

Wikidata schemas. The Schemas Wikidata project[13] aims to define schemas, expressed in the Shape Expression language (ShEx) that can be used for the validation of subsets of items inside the Wikidata KG, in order to check whether they conform to a specific and *recommended* structure and possibly improve the quality of the data. Currently, more than 300 schemas have been manually defined by the Wikidata community[14], and these schemas vary considerably with respect to their size and granularity. For instance, the shape *E25* for actors[15] includes 4 constraints, and the only domain-specific constraint, i.e. a constraint that can be considered valid only for humans that are authors, not any human, specifies their occupation (*actor*). Instead, the shape *E42* for authors[16] is more detailed, and includes both constraints that are valid for all humans (shape *E10*) and more *author-specific* constraints, such as *copyright status*. In any case, it is rare that these constraints express the suggested range; for example, the property *notable work* in the author shape has generically an IRI as recommended range, instead of possible specific classes.

Properties list in a WikiProject. In the context of domain-specific projects (e.g. music, astronomy, books, geology), the members of the community that are expert in that domain may define a list of properties that are recommended to be used for describing relevant entities of that domain. Each property, listed in a table, is usually associated with the data type of its range (that is, item, string, etc.), and a description of the usage of that property. This description, in some cases, also includes in plain text possible types for the range. For example, in the context of the WikiProject Books[17], the textual description of the property *inspired*

[13]https://wikidata.org/wiki/Wikidata:WikiProject_Schemas
[14]https://wikidata.org/wiki/Special:AllPages?from=&to=&namespace=640
[15]https://www.wikidata.org/wiki/EntitySchema:E25
[16]https://www.wikidata.org/wiki/EntitySchema:E42
[17]https://wikidata.org/wiki/Wikidata:WikiProject_Books

by, recommended for instances of *written work*, recommends *artistic inspiration* as range. The whole process of defining relevant properties for a domain is performed manually, and possible ranges for properties are not always specified, and, in any case, not formally.

Some tools proposed by/to the Wikidata community for inserting new data suffer from a problem similar to the shapes generated with current automatic methods: for instance, Recoin[18] recommends properties for a class based on their frequency in the KG, and lists frequent properties that are missing for instances of a specific type, but does not provide information on the appropriate ranges.

5.2.2 Input

In order to deal with the huge size of Wikidata, we used the Knowledge Graph Toolkit (KGTK)[19] [55]. KGTK is a recent framework, developed as a Python library that aims at supporting an easy manipulation of knowledge graphs, thanks to its scalability and speed. We downloaded a json dump of two versions of Wikidata[20]: on 04-04-2022 and 6 months later, on 10-10-2022. The experiments that we present in Sections 5.2.3 and 5.2.4, and that we evaluate in Section 5.2.5, have been run on the former version (04-04-2022); however, in Section 5.2.5.4 we compare the patterns obtained from the two versions in order to show their evolution in 6 months.

In order to extract domain-specific patterns, and to handle a sub-graph of Wikidata with a more manageable size, we focused on specific domains represented in the Wikidata KG. While we choose to work on the ‘music’ and ‘art, architecture, and archaeology’ domains (AAA), the method can be applied to any domain. The extraction of instances related to both domains is based on a list of WordNet and BabelNet synsets identified as belonging to the respective domains, according to BabelDomains [89]. Then, the two Wikidata subgraphs are extracted by selecting each triple where the Wikidata domain-specific instance is in the subject position. We

[18]https://www.wikidata.org/wiki/Wikidata:Recoin
[19]https://kgtk.readthedocs.io/en/latest/
[20]https://dumps.wikimedia.org/wikidatawiki/entities/

remark that our method does not focus on the extraction of domain-specific knowledge graphs from Wikidata (or any KG): by relying on BabelDomains we were able to easily obtain the two Wikidata subgraphs, however, a user that wants to use our method for extracting empirical patterns from a KG, could either run the method on the whole Wikidata KG, or give as input a Wikidata subgraph customized with a method of her choice.

As explained in Section 5.1, each main step of our method takes as input a threshold, for filtering the list of classes (which defines the set of empirical patterns that will be extracted), the list of properties to be considered for each pattern, and the number of ranges that will define the final number of triplets/probabilistic axioms. These thresholds are to be chosen by the user, based on her requirements. In this work, we do not define a method for finding the *best* thresholds to be used. However, for the purpose of presenting our results in this thesis, comparing them with the current support offered in Wikidata, we have chosen the thresholds we considered reasonable. For both the music and AAA patterns, the threshold T_c is equal to 0.95, the threshold T_p is equal to 0.85, and the threshold T_{dr} is 0.5[21].

5.2.3 Wikidata empirical patterns on music

The Wikidata subgraph on the music domain contains 5,083,818 triples, and 226,989 distinct instances.

Most populated classes: music patterns. Having T_c equal to 0.95, we filter out all classes that have a number of instances that is lower than the 5% of the number of instances of the most populated class (from a total of 6,043 classes, ∼6,000 of which have less than 200 instances). Consider that the same entity can be an instance of more than one class.

In Table 5.1, the 7 classes around which we build our music patterns are presented, with the respective number of instances and of triples that have an instance

[21]Both code and results are available on GitHub: https://github.com/valecarriero/wikidata-emerging-patterns

Table 5.1: Most populated classes in the Wikidata music subKG.

Class	Instances	Triples
Q5 human	63,594	2,348,331
Q482994 album	63,213	723,722
Q215380 musical group	25,016	527,537
Q134556 single	20,977	253,201
Q105543609 musical work/composition	14,600	198,841
Q169930 extended play	3,816	33,725
Q18127 record label	3,640	35,118

of the class as subject. The most relevant entities in this Wikidata music KG include both agents (such as human and musical group) and objects (namely, single, album, musical work, extended play, record label). Notice that *single* (`wd:Q134556`) and *extended play* (`wd:Q169930`) are not subclasses of *musical work/composition* (`wd:Q105543609`) in the general Wikidata hierarchy (`wdt:P279*`).

If we have a look at the ratio between the number of instances and the number of triples for each class, we can notice at first sight that humans are more well *described* by facts than albums, if we consider that the number of respective instances is roughly equal.

Recommended properties for each pattern. The average number of the most frequent properties selected for each pattern based on the T_p threshold is ∼21. In Table 5.2 we report the actual number of properties for each class, and the respective maximum and minimum number of their *occurrences*, intended as the number of instances that are subject of at least one triple including a specific property. As it can be noticed from the table, the number of selected properties is not directly proportional to the number of triples in the subgraph: for instance, for musical groups we recommend more properties that are frequently used (selected out of a total of 891 properties) than albums (369 properties in total). The most common properties across all patterns (without considering ID properties) are: `wdt:P136` *genre*, which is used with all 7 classes, and `wdt:P264` *record label*, present in all patterns except for record labels.

Table 5.2: Statistics of selected properties and triplets for each music pattern.

Class	Properties	Occurrences		Triplets
		max	min	
Q5 human	48	63,583	9,543	63
Q482994 album	14	61,772	11,735	18
Q215380 musical group	33	22,423	3,474	38
Q134556 single	15	20,860	5,076	22
Q105543609 musical work/composition	17	13,916	2,204	29
Q169930 extended play	10	3,793	650	12
Q18127 record label	11	3,577	625	20

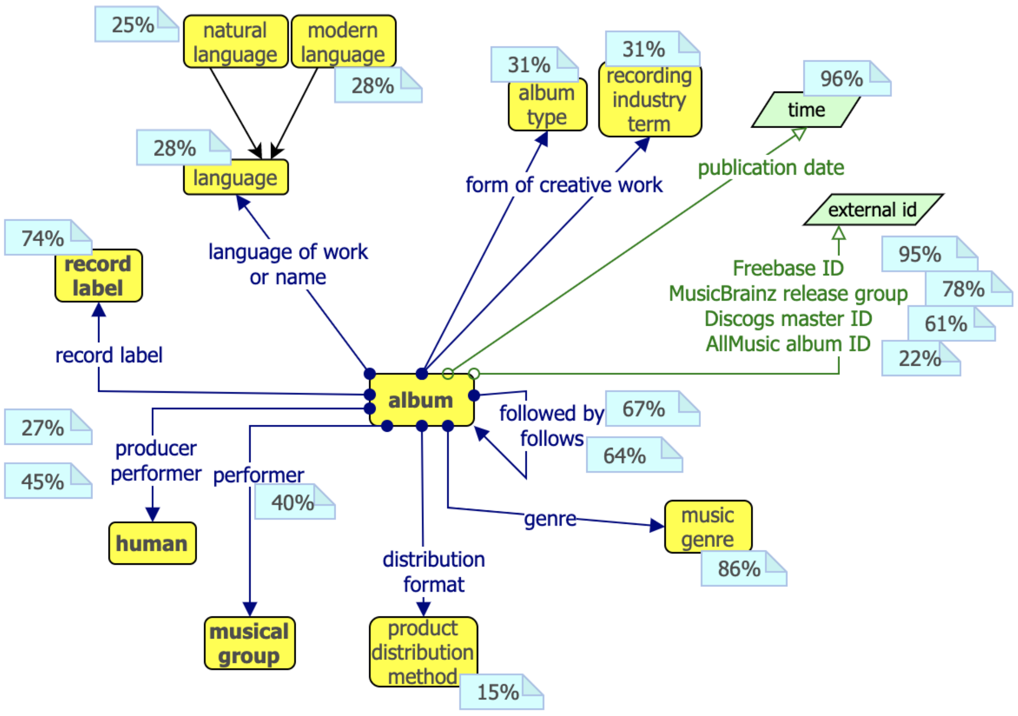


Figure 5.2: The album pattern.

Recommended ranges for each property. Table 5.2 lists the number of triplets $\langle d,\ p,\ r\rangle$ – that is, the domain d and range r pairs for each recommended property p – generated for each pattern. Datatype properties will have only one range in any case, while for other properties the number of ranges depends on the T_{dr} threshold. The average number of triplets across all patterns is $\sim$29. Since the same property can be involved in more than one pattern, for each pattern we may recommend different ranges for that property, except for datatype properties. That is, ranges recommendations are local to the individual pattern. For example, both the patterns for albums and singles include the property `wdt:P155` *follows*, with different ranges: *album* for albums, and *single* for singles, as expected.

Example: the album pattern. In Figure 5.2 we graphically represent the pattern for albums. Each domain-property-range triplet is accompanied by the number of instances in the Wikidata music subgraph that comply with that triplet (light blue rectangles). Based on the threshold we used, for most properties we recommend only one range. However, the *performer* property, when used with albums, can have either a human or a musical group as range within the pattern, and the 3 recommended ranges for the property *language of work or name* have a subclass-of relation. It is interesting to notice that 4 recommended properties link to other empirical patterns as recommended ranges (*record label*, *human*, *musical group*).

Listing 5.1 contains a snapshot of the rdf-star – using the ttl-star syntax – probabilistic pattern about albums, derived from the sets of triplets selected from our method. The first 9 triples that can be found in 5.1, after the list of prefixes, aim at annotating the pattern itself and its related resources: the pattern (`weps:music_Q482994_09508505` is defined as an `oplax:FrequentistProbabilisticPattern`, is related to its source, i.e. the music Wikidata subgraph, and to the actual Wikidata class it is built around (`dcterms:references wd:Q482994`). Then, the music Wikidata subgraph, defined as a dataset, is linked to the whole Wikidata KG version it was derived from, and is part of (`weps:wikidata_20220404`). Finally, the whole version of Wikidata is annotated with the date on which the KG dump was downloaded ("2022-04-04"). The main properties and classes used for these annotations are also reported in Figure 5.3. The next statements are a subset of the probabilistic axioms we generate starting from the triplets of the pattern. `os:interpretation os:AllSomeInterpretation` indicates that these axioms are existential. The first one, `wd:Q482994 wdt:P175 wd:Q35120` (*album, performer, entity*) is the general probabilistic axiom about an album linked to some entity with the property *performer*. The frequentist probability of this axiom, annotated with `os:frequentistProbability`, is equal to 97.72%. However, as already explained, we also suggest more specific ranges for properties. In this case, two additional probabilistic axioms are generated: *album, performer, human* (probability 44.62%) and *album, performer, musical group* (probability 40.39%). The sum of the probabilities of such axioms is 85.01%: the remaining 12.71% corresponds to the sum of the probabilities of all the other possible, less frequent, ranges for this property when used with albums, that have been discarded based on the selected T_{dr} threshold.

Frame 5.1: Snapshot of the album empirical ODP, expressed using rdf-star and owl-star.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .
@prefix wikibase: <http://wikiba.se/ontology#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix os: <http://w3id.org/owlstar/> .
@prefix oplax: <https://w3id.org/OPLaX/> .
@prefix weps: <https://w3id.org/wikidata-eps/> .

weps:music_Q482994_09508505 rdf:type
 oplax:FrequentistProbabilisticPattern ;
 dcterms:source weps:wikidata_music_subkg_20220404 ;
 dcterms:references wd:Q482994 .

weps:wikidata_music_subkg_20220404 rdf:type dcterms:Dataset ;
  prov:derivedFrom weps:wikidata_20220404 ;
  dcterms:isPartOf weps:wikidata_20220404 .

weps:wikidata_20220404 rdf:type dcterms:Dataset ;
 dcterms:hasPart weps:wikidata_music_subkg_20220404 ;
 dcterms:date "2022-04-04"^^xsd:date .
```

```
# 'album' 'performer' 'entity'
<< << wd:Q482994 wdt:P175 wd:Q35120 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "97.72%" ;
oplax:isNativeTo weps:music_Q482994_09508505 .

# 'album' 'performer' 'human'
<< << wd:Q482994 wdt:P175 wd:Q5 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "44.62%" ;
oplax:isNativeTo weps:music_Q482994_09508505 .

# 'album' 'performer' 'musical group'
<< << wd:Q482994 wdt:P175 wd:Q215380 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "40.39%" ;
oplax:isNativeTo weps:music_Q482994_09508505 .
```

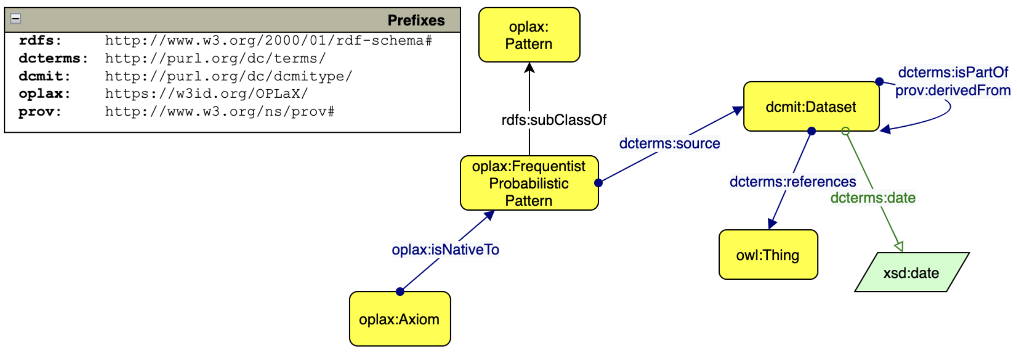

Prefixes
rdfs: http://www.w3.org/2000/01/rdf-schema#
dcterms: http://purl.org/dc/terms/
dcmit: http://purl.org/dc/dcmitype/
oplax: https://w3id.org/OPLaX/
prov: http://www.w3.org/ns/prov#
oplax:Pattern
rdfs:subClassOf
oplax:Frequentist Probabilistic Pattern
dcterms:source
dcmit:Dataset
dcterms:isPartOf
prov:derivedFrom
dcterms:references
dcterms:date
owl:Thing
xsd:date
oplax:isNativeTo
oplax:Axiom

Figure 5.3: Classes and properties used for annotating the probabilistic pattern.

The axioms in 5.1 correspond to the subset of the ShEx shape we generate for this pattern, that can be found in Listing 5.2. The frequentist probabilities are reported via comments. Additionally, the shape also includes the constraint about the type (`wdt:P31`) of the instances that need to comply with the shape, i.e. `wd:Q482994`.

Frame 5.2: Snapshot of the album empirical ODP, expressed as a shape using ShEx.

```
# shape extracted from dataset
wikidata_music_subkg_20220404
which is derived from wikidata_20220404,
dated 2022-04-04

start = @<album>

<album> { wdt:P31 [wd:Q482994] } ;

# 'album' 'performer' 'entity'
wdt:P175 { wdt:P31 [wd:Q35120] } ;   # probability: 97.72%

# 'album' 'performer' 'human'
wdt:P175 { wdt:P31 [wd:Q5] } ;   # probability: 44.62%

# 'album' 'performer' 'musical group'
wdt:P175 { wdt:P31 [wd:Q215380] } ;   # probability: 40.39%

}
```

5.2.4 Wikidata empirical patterns on art, architecture, and archaeology (AAA)

The Wikidata subgraph on the art, architecture, and archaeology (AAA) domain contains 493,999 triples, and 26,380 distinct instances.

Most populated classes: AAA patterns. From the Wikidata art, architecture, and archaeology (AAA) subgraph, using the 0.95 threshold, we select 11 classes from a total of 2184 classes, ~2100 of which have less than 50 instances.

Table 5.3 presents the list of 11 AAA patterns, along with their number of instances and the number of triples with an instance of the class as subject. The only agent included in the most relevant entities of the Wikidata AAA domain is

Table 5.3: Most populated classes in the Wikidata art, architecture, and archaeology subKG.

Class	**Instances**	**Triples**
Q5 human	7,622	214,881
Q3947 house	2,333	21,433
Q41176 building	1,617	15,382
Q23413 castle	1,043	13,556
Q33506 museum	636	9,129
Q11303 skyscraper	601	8,011
Q1343246 English country house	596	7,240
Q3305213 painting	564	12,814
Q1307276 single-family detached home	459	4,922
Q27686 hotel	427	4,577
Q207694 art museum	413	7,994

human, while all other patterns are built around objects (house, skyscraper, painting, museum, etc.). As for hierarchical relations (`wdt:P279*`) between the classes of the AAA patterns, *building* has 6 (indirect) subclasses (*house*, *castle*, *skyscraper*, *English country house*, *single-family detached home*, *art museum*); *English country house* and *single-family detached home* are subclasses of *house* (indirect and direct, respectively); *art museum* is an indirect subclass of both *museum* and *building.* Surprisingly, *museum*, unlike *art museum*, is not a subclass of *building*, as well as *hotel.* Finally, *human* and *painting* are not subclasses of any other empirical pattern. Humans, paintings and art museums are more well *described* with facts than the other selected classes.

Recommended properties for each pattern. The average number of selected properties for each pattern on the AAA domain is $\sim$16. In Table 5.4 you can find reported the number of selected properties for each pattern, and the maximum and minimum number of occurrences from this set of properties.

Recommended ranges for each property. Table 5.4 also reports the number of triplets $\langle d, p, r \rangle$ recommended for each pattern, based on the 0.5 threshold. The

Table 5.4: Statistics of selected properties and triplets for each AAA pattern.

Class	**Properties**	**Occurrences**		**Triplets**
		max	min	
Q5 human	33	7,614	1,151	46
Q3947 house	9	2,331	494	15
Q41176 building	10	1,613	278	16
Q23413 castle	12	1,043	161	22
Q33506 museum	15	632	99	19
Q11303 skyscraper	19	600	106	28
Q1343246 English country house	11	596	110	17
Q3305213 painting	23	562	91	32
Q1307276 single-family detached home	11	459	108	17
Q27686 hotel	13	427	79	19
Q207694 art museum	21	413	65	26

average number of triplets across all AAA patterns is ∼23.

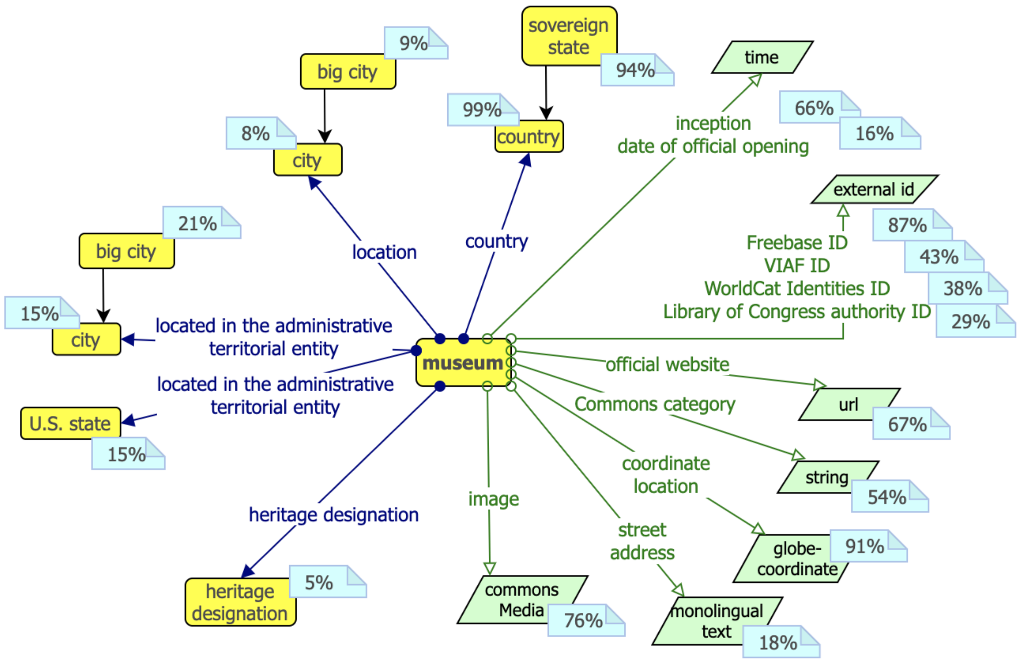


Figure 5.4: The museum pattern.

Example: the museum pattern. In Figure 5.4 we provide a graphical representation of the *museum* pattern. Based on the 0.5 threshold, 15 properties have been selected. 11 out of the total number of properties are datatype, having only one recommended range (monolingual text, url, etc.). 1 of the remaning 4 object properties has one recommended range, i.e. *heritage designation* for the property *heritage designation*, while the other 3 properties, which are all related to locations, have more than one range: *country* and its subclass *sovereign state* for the property *country*, with a very close percentage of coverage; *city* (8%) and its subclass *big city* (9%) for the property *location*; and *city* (15%) and *big city* (21%) for the property *located in the administrative territorial entity*, in addition to *U.S. state* (15%). As it can be noticed from the percentages, *country* is the most frequent place-related properties, being used by almost all museums (99%). Listing 5.3 contains a snapshot of the *museum* probabilistic pattern. After the triples for annotating the pattern and its related sources, we include part of the probabilistic axioms automatically generated. `wd:Q33506 wdt:P571 wikibase:Time` (*museum, inception, time*) is a probabilistic axiom including a datatype (*time*), which is, as expected, the only range for the property *inception*, with a frequentist probability equal to 66.35%. Then, the following statements involve the property *country*; it is clear from the probabilities that *country* and *sovereign state* are very frequently used as ranges for this property when the subject is a museum, and the majority of the instances of the ranges for this property are both of type *country* and *sovereign state*.

Frame 5.3: Snapshot of the museum empirical ODP, expressed using rdf-star and owl-star.

```
weps:AAA_Q33506_09508505 rdf:type
 oplax:FrequentistProbabilisticPattern ;
 dcterms:source weps:wikidata_AAA_subkg_20220404 ;
 dcterms:references wd:Q33506 .

weps:wikidata_AAA_subkg_20220404 rdf:type dcterms:Dataset ;
 prov:wasDerivedFrom weps:wikidata_20220404 ;
 dcterms:isPartOf weps:wikidata_20220404 .

weps:wikidata_20220404 rdf:type dcterms:Dataset ;
```

```
 dcterms:hasPart weps:wikidata_AAA_subkg_20220404 ;
 dcterms:date "2022-04-04"^^xsd:date .

# 'museum' 'inception' time
<< << wd:Q33506 wdt:P571 wikibase:Time >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "66.35%" ;
oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'entity'
<< << wd:Q33506 wdt:P17 wd:Q35120 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "99.37%" ;
oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'country'
<< << wd:Q33506 wdt:P17 wd:Q6256 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "99.21%" ;
oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'sovereign state'
<< << wd:Q33506 wdt:P17 wd:Q3624078 >>
os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "94.5%" ;
oplax:isNativeTo weps:architecture_Q33506_09508505 .
```

In 5.4 you can find the constraints that correspond to the axioms in 5.3.

Frame 5.4: Snapshot of the museum empirical ODP, expressed as a shape using ShEx.

```
# shape extracted from dataset
wikidata_AAA_subkg_20220404
which is derived from wikidata_20220404,
dated 2022-04-04
```

```
start = @<museum>

<museum> { wdt:P31 [wd:Q33506] } ;

# 'museum' 'inception' time
wdt:P571 xsd:dateTime ;   # probability: 66.35%

# 'museum' 'country' 'entity'
wdt:P17 { wdt:P31 [wd:Q35120] } ;   # probability: 99.37%

# 'museum' 'country' 'country'
wdt:P17 { wdt:P31 [wd:Q6256] };   # probability: 99.21%

# 'museum' 'country' 'sovereign state'
wdt:P17 { wdt:P31 [wd:Q3624078] } ;   # probability: 94.5%

}
```

5.2.5 Discussion and evaluation

In this section, we discuss the results obtained from both the music and AAA domains, and we perform an evaluation of the resulting patterns by comparing them with the current support provided by Wikidata.

5.2.5.1 Music

Patterns coverage. In order to analyse how the extracted patterns are populated in the Wikidata music subKG, we report in Table 5.5[22] the percentage of the total number of instances that cover different and increasing subsets of recommended

[22]Columns indicate the number/fraction of properties considered. The actual number of properties corresponding to the fraction is reported in square brackets. The number of instances covering the whole pattern is in round brackets. Example instances populating the whole patterns can be found here: https://github.com/valecarriero/wikidata-emerging-patterns/tree/main/results/music/supplementary_materials/example_instances

properties. No pattern has a 100% coverage even if we only consider the most frequent property; in some cases, the percentage of coverage of the set including the two most common properties is very close to the one of the first property (e.g. in the *human* pattern), while in other cases (e.g. the pattern *musical work*) it already decreases significantly. In 4 out of 7 patterns, the instances that comply with the first half (1/2) of the recommended properties are between the 35 and ∼58% of the total number of instances; instead, musical works, humans, and musical groups have already a very low percentage of coverage. The most populated pattern (including all properties), with respect to the total number of its instances, is *extended play* (112/3,816), followed by *album* (845/63,213) and *musical group* (327/25,016). Instead, the pattern that has the lower percentage of coverage is *musical work*: only 1 out of a total of 14,600 musical works. Anyway, all patterns have at least 1 representative instance. While the coverage percentages might seem very low, this is not *bad*, neither surprising: by using the 0.85 threshold, as in our experiments, we include all properties that are used by at least 15% of the total number of instances. So, taking the *record label* pattern as an example, if the least common property is used for 625/3,577 instances, it is not surprising that the intersection of instances with all the 11 recommended properties is equal to 28 instances.

Comparison with property constraints. If we consider the most common properties across all patterns, that is *genre* (7/7 patterns) *record label* (6/7), we can observe that the domains and ranges we suggest are all included in the *subject type* and *value-type* constraints of the two properties. In some cases, the Wikidata constraints include as ranges classes that are more general in the hierarchy with respect to the classes we suggest: for instance, they include work in place of the more specific musical work. However, as explained in Section 5.2.1, the correct pairs of domain and range cannot be specified inside Wikidata, thus our results integrate these constraints by suggesting that e.g. *music genre* is *more appropriate* as range of the property *genre* with *record label* as domain, than e.g. *criticism* – which is included in the value-type constraint of *genre*, and which never occurs in the data. Moreover, the *subject type* and *value-type* constraints are not available for all properties; for instance, *follows* (used in 4 out of 7 patterns) has no such constraints.

Table 5.5: Percentages of coverage of the music patterns properties in the KG.

Class	1 prop	2 props	1/8	1/4	1/2	all
Q5 human	99.98	98.99	[8] 50.34	[12] 32.97	[24] 3.65	[48] 0.007 (5 instances)
Q482994 album	97.72	94.19	[2] 94.19	[4] 78.30	[7] 40.48	[14] 1.33 (845 instances)
Q215380 musical group	89.63	78.36	[4] 60.99	[8] 34.22	[16] 9.82	[33] 1.31 (327 instances)
Q134556 single	99.44	98.80	[2] 98.80	[4] 87.87	[7] 57.67	[15] 0.71 (151 instances)
Q105543609 musical work	95.31	76.36	[2] 76.36	[4] 39.69	[8] 6.34	[17] 0.006 (1 instance)
Q169930 extended play	99.39	97.95	[1] 99.39	[3] 92.29	[5] 56.70	[10] 2.93 (112 instances)
Q18127 record label	98.26	84.25	[1] 98.26	[3] 69.06	[5] 35.0	[11] 0.76 (28 instances)

Comparison with properties for this type. Taking into account our selected music patterns, we compared the properties we include and those included as value of the property *properties for this type* (`wdt:P1963`) for those classes. Based on a manual observation, we can report that some properties that are highly populated in the data are not suggested as properties for this type, while all the properties actually listed as properties for the type, but not included in our patterns, are significantly less frequent, and in some cases with a very low number of occurrences. Let us take musical group[23] (`wd:Q215380`) as an example. IDs properties such as *Freebase ID*, *MusicBrainz artist ID* and *Discogs artist ID* are widely used (about 81, 75 and 74 % respectively), but not listed as properties for this type. On the contrary, IDs that are less frequent in the data (e.g. *Apple Music artist ID (U.S. version)*, with ∼6.5% coverage for musical groups), thus filtered out from our pattern, are recommended as properties for this type. As another example, the properties *influenced by* and *award received* are recommended for musical groups inside Wikidata, while they are discarded in our pattern because they have a very low frequency (less than 0.5 and 2 %, respectively). Anyway, 10 out of the 18 properties recommended as properties

[23]https://github.com/valecarriero/wikidata-emerging-patterns/blob/main/results/music/supplementary_materials/properties_forthis_type/Q215380_properties_comparison.tsv

for this type are also included in our pattern.

Comparison with type of wikidata property. `wd:Q27525351`, a subclass of *Type of Wikidata property*, includes as instances *properties related to music*, such as music-related IDs (such as *YouTube playlist ID*) and other relations (e.g. *composer*, *performed at*, *discography*). However, it is not indicated, for each property, their possible domain(s), so, if a user needed to model a specific musical entity, e.g. a musical group, would have no support for understanding which of those properties to use for musical groups. *Wikidata property related to music* has 24 subclasses that are specific to some musical entities (e.g. music genres, songs, instruments). However, 14/24 include only identifiers, e.g. IDs for musical works, songs and bands. Even considering just the identifiers, our patterns are more complete and representative. For example, the class *Wikidata property to identify bands*, which is *facet of musical group*, has as instance only the property *Encyclopaedia Metallum band ID* (`wdt:P1952`). The pattern we extracted for the class *musical group* contains 33 properties, including the most common IDs, – like *Freebase ID*, *MusicBrainz artist ID* and *Discogs artist ID* – while excluding `wdt:P1952`, which is used with only 8% of musical groups. Moreover, some relevant properties that we are able to include in the patterns cannot be identified based on the *Wikidata property* classes: for instance, *genre*, which is widely used in the data for describing instances of musicians (about 50%) and musical works (about 60%), is recommended for both humans and musical works/compositions, while it can only be found as instance of the more general classes *Wikidata property for items about people* and *for items about works.*

Comparison with properties listed in the WikiProject Music. The WikiProject Music[24] (WPMusic hereinafter) identifies 6 relevant entities in the domain, and defines a set of recommended properties for each of them: human, musical ensemble, musical work, track, release, record label. Apart from *human* and *record label*, our patterns do not perfectly overlap with these 6 classes: musical ensemble *vs* musical group (the latter is the most populated subclass of the former); musical work *vs* musical work/composition (musical work has very few direct instances, while many

[24]https://wikidata.org/wiki/Wikidata:WikiProject_Music

of its subclasses are widely populated e.g. song); release, which groups together its subclasses album, single and extended play. However, it is still possible and useful to make a comparison between them. Let us consider the class *record label*: the WPMusic recommends 4 properties[25] in addition to 13 IDs. Our pattern contains 6 properties, in addition to 5 identifiers. Apart from *instance of* `wdt:P31`, which we always exclude from our patterns, we report in Table 5.6 a comparison between the properties recommended by WPMusic and those selected by our method (EP), except for IDs and `wdt:P31`[26]. It can be observed that our pattern is more inclusive (6 *vs* 3 properties) with the threshold we have chosen. More importantly, we include all properties recommended by WPMusic, while WPMusic does not recommend the property *inception*, even if it is the second most frequent property. In our pattern, *country* (that is the range of the property *country* recommended by WPMusic) is indeed the most frequent range, but we also suggest 6 additional and more specific classes, such as *sovereign state*.

Table 5.6: Comparison between properties recommended by WikiProject Music and properties included in our pattern for record labels.

Property	Occurrences	WPM	EP
P17 country	3,123	Y	Y
P571 inception	2,905	N	Y
P856 official website	1,833	Y	Y
P159 headquarters location	1,023	Y	Y
P136 genre	972	N	Y
P112 founded by	714	N	Y

Now, let us compare the properties recommended for instances of subclasses of Release by WPMusic and our relevant patterns album (A), single (S) and extended play (P) (Table 5.7). 6 properties recommended by WPMusic are included in all our 3 patterns, 3 properties are included in some of our patterns, while 9 properties

[25]https://wikidata.org/wiki/Wikidata:WikiProject_Music#Record_label_properties

[26]Y stands for yes, N stands for no.

Table 5.7: Comparison between properties recommended by WikiProject Music and properties included in our patterns for releases.

WPM Property	**EP**	**WPM Property**	**EP**	**WPM Property**	**EP**
P577 publication date	A, S, P	P136 genre	A, S, P	P156 followed by	A, S, P
P155 follows	A, S, P	P264 record label	A, S, P	P175 performer	A, S, P
P162 producer	A, S	P407 language of work	P	P361 part of	S
P1303 instrument	none	P483 recorded at studio	none	P676 lyrics by	none
P86 composer	none	P658 tracklist	none	P736 cover art by	none
P2291 charted in	none	P9237 reissue of	none	P1638 working title	none

are not included. However, for instance, the property *composer* is used only 6, 186 and 1602 times for instances of extended play, album and single, respectively; *instrument* is never used for any of these entities (instead, it is frequently used, and is included, in the *human* pattern); *working title* is used only twice for albums, while the property *title* (`wdt:P1476`) is much more frquent (9,007 occurrences).

Comparison with music-related shapes. We manually identified from the complete list of Wikidata entity schemas[27] only two music-related shapes: *music composition by W.A.Mozart* (E66), and *album* (E248), so it is strongly relevant to try to increase the coverage at least of the music domain. The shape for albums[28] (E248) recommends 18 properties as obligatory (with *exactly one/at least one* constraints): 7 of these properties are also included also in our pattern; while some recommended properties can actually be considered statistically relevant (such as the property *title*: 9,007/63,213 occurrences) and would have been included in our pattern with a little higher threshold, other properties included in the shape have very few occurrences that, at least, do not seem to justify their mandatory use (e.g. *review score*, with 722 occurrences, and *distributed by*, with 299 occurrences). Instead, for instance, the *producer* property (with *any number* as cardinality constraint in the

[27]https://wikidata.org/wiki/User:HakanIST/EntitySchemaList
[28]https://www.wikidata.org/wiki/EntitySchema:E248

E248 shape) is much more used (18,362), and is indeed recommended in our pattern.

5.2.5.2 Art, architecture, and archaeology (AAA)

Patterns coverage. In Table 5.8 we show the percentages, out of the total number of instances, that cover increasing subsets of the properties recommended for each AAA class. 3 out of 11 AAA patterns (i.e. *castle*, *English country house*, and *art museum*) have 100% coverage considering only the most frequent property, that is, all instances of the 3 classes are subject of at least one triple involving the most frequent property of the respective pattern. However, in the case of the *castle* and *English country house* patterns also the percentage of coverage considering the whole set of properties of the patterns is high (wrt the other patterns), i.e. 2.78% and 9.89%, respectively, being *English country house* the most populated pattern out of all patterns. Instead, *art museum* has the 0.24% of its instances complying with the whole pattern (i.e. 1/413). Other patterns widely populated taking into account all properties, wrt the total number of instances, are *house* (4.88%) and *single-family detached home* (5.88%). *painting* is the only pattern that does not have at least one representative entity for all properties. This pattern, which instead has a coverage higher than other patterns when considering the first sets of properties (e.g. 97.34% with the first three properties), has a 0.70 percentage of coverage with 20/23 selected properties, while has no instances covering all the most frequent 21, 22, and 23 (i.e. all) properties. Other patterns with a relatively low percentage of coverage considering the whole set of properties are: *human* (8 individuals complying with the whole pattern out of 7,622), *museum* (1 out of 636), and *hotel* (1 out of 427).

Comparison with property constraints. The two properties more used across all patterns are `wdt:P17` *country* (that is recommended for 9 out of 11 patterns) and `wdt:P131` *located in the administrative territorial entity* (9/11 patterns). Neither properties have the *subject type* constraint, thus we could contribute by integrating the constraints and suggesting possible classes the subject can be type of (like *building*) in the AAA domain. All ranges we recommend for the *located in the administrative territorial entity* property have a `wdt:P279*` hierarchical relation with

Table 5.8: Percentages of coverage of the AAA patterns properties in the KG.

Class	1 prop	2 props	1/8	1/4	1/2	all
Q5 human	99.89	99.39	[4] 90.12	[8] 39.71	[16] 2.91	[33] 0.10 (8 instances)
Q3947 house	99.91	98.92	[1] 99.91	[2] 98.92	[4] 96.39	[9] 4.88 (114 instances)
Q41176 building	99.75	97.71	[1] 99.75	[3] 94.61	[5] 77.79	[10] 0.37 (6 instances)
Q23413 castle	100	99.04	[2] 99.04	[3] 93.76	[6] 60.88	[12] 2.78 (29 instances)
Q33506 museum	99.37	90.72	[2] 90.72	[4] 70.12	[7] 33.96	[15] 0.15 (1 instance)
Q11303 skyscraper	99.83	97.17	[2] 97.17	[5] 71.38	[9] 33.44	[19] 0.33 (2 instances)
Q1343246 English country house	100	93.95	[1] 100	[3] 92.44	[5] 73.15	[11] 9.89 (59 instances)
Q3305213 painting	99.64	98.93	[3] 97.34	[6] 72.51	[11] 51.59	[23] 0 (0 instances)
Q1307276 single-family detached home	99.78	99.78	[1] 100	[3] 99.12	[5] 97.16	[11] 5.88 (27 instances)
Q27686 hotel	100	94.14	[2] 94.14	[3] 81.49	[6] 20.84	[13] 0.23 (1 instance)
Q207694 art museum	100	94.67	[3] 86.44	[5] 75.54	[10] 41.64	[21] 0.24 (1 instance)

one of the classes included in the *value-type* constraint (namely `wd:Q56061` *administrative territorial entity*), excluding *village* (`wd:Q532`). 3 out of 11 ranges suggested for the property *country* are included in the *value-type* constraint, 5/11 are instead subclasses of classes listed in the constraint, while 3/11 (namely *democratic republic*, *constitutional republic* and *superpower*) are not included in the *value-type* constraint list; however, for example, more than 80% buildings have a *constitutional republic* as type of the object of the property *country*, so this range, even if not explicitly recommended, is very common in the data.

Comparison with properties for this type. 8 out of the 11 AAA classes have some values for the property *properties for this type* (`wdt:P1963`). Many frequently used properties that we include are left out from the *properties for this type*. For example, Wikidata recommends 30 properties for the type *painting*. 19 of these properties (like *creator*, *collection*, *height*, *width*, etc.) are also included in our pattern for *paintings*, while the remaining 11 properties have a percentage of occurrences that ranges between 11% of all paintings and 0%. Our pattern recommends also other 4 properties that are absent from the *properties for this type*: *Freebase ID* (99.65%), *Commons category* (40.78%), *BabelNet ID* (40.07%) and *Google Arts & Culture asset ID* (17.3%). Let us take *skyscraper* as another example: out of the 16 *properties for this type*, 7 are also present in our pattern, and the remaining 9 have been excluded due to their low frequency (from 8% to 0.17%), while very frequent properties of our pattern, such as *coordinate location* (97.17%) and *image* (83.86%), are not listed as *properties for this type.*

Comparison with type of wikidata property. Instances of *Type of Wikidata property* relevant to the AAA domain are: *Wikidata property related to artworks* (`wd:Q44847669`), and *Wikidata property related to architecture* (`wd:Q43831109`). *Wikidata property related to artworks* is only linked to very specific identifier properties, e.g. IDs related to individual museums, while the only IDs we selected for *paintings*, based on their usage, are quite general: *Freebase ID* (which is the most common property for paintings), *inventory number*, *BabelNet ID*, *catalog code*, and *Google Arts & Culture asset ID* – which is actually included in the Wikidata properties related to artworks too. *Wikidata property related to architecture*, apart from IDs, lists the properties *architect*[29], which is recommended by our *skyscraper* and *single-family detached home* patterns, and *architectural style*, which is included in the patterns for *skyscrapers*, *single-family detached homes* and *buildings. Wikidata property for items about buildings* only includes the property *has certification*, which is never used for buildings.

Comparison with properties listed in the WikiProjects. There is no WikiProject that addresses the whole Art, architecture, and archaeology (AAA) domain. The WikiProject *Archaeology*[30] is related to archeology in general, however the list of properties included, which does not specify the different classes of the subject, contains only identifiers, apart from the property *director of archaeological fieldwork.* The WikiProject *sum of all paintings* lists 6 important properties for paintings[31],

[29]And the more specific *landscape architect.*

[30]https://www.wikidata.org/wiki/Wikidata:WikiProject_Archaeology

[31]https://www.wikidata.org/wiki/Wikidata:WikiProject_sum_of_all_paintings#Item_structure_to_describe_paintings_on_Wikidata

which we all include in our *painting* pattern. There is also a list of properties related to *structures*[32]; however, it is not completely clear the kind of structures it refers to. By looking at the triples mentioned as examples, we can find instances of hotels, bridges, skyscrapers, airports, so we could probably associate these properties with instances of the class *building* and subclasses. Except for IDs, the properties that we can relate to buildings are quite specific (e.g. *structure replaced by*, which is never used neither for buildings nor hotels in our AAA subKG), so most of them are absent from our *structure-related* patterns, apart from *floors above ground*, that is included in the *skyscraper* and *hotel* patterns, and *number of elevators*, recommended by the *skyscraper* pattern.
The WikiProject *Museums*[33] (WPMuseum hereinafter) identifies a set of properties recommended for museums, splitting them based on different topics (location, communications, practical information, and so on). We report in Table 5.9 a comparison between the properties recommended by WPMuseum and our method (i.e. the EP *museum*), except for identifiers. Our pattern includes 7 properties that are also recommended by WPMuseum, whose percentages of occurrence range from 99% to 18%, while WPMuseum does not recommend 3 properties we include, one of which is very frequently used: *image* is used for 75.79% museums, while WPMuseum recommends a more specific property, *logo image*, which is only used by 1.57% museums. In addition to *logo image*, WPMuseum also recommends 17 other properties that are instead absent from our pattern; however, even if we used a higher threshold, thus including more properties, many of them would be discarded in any case: *closed on*, *open period from* and *open period to* have 0 occurrences in the data, and other 9 properties have a percentage of occurrence between 0% and 3%.

Comparison with AAA-related shapes. There are 10 shapes related to architecture, archaeology and art that we were able to identify from the list of Wikidata entity schemas[34], namely: *statue* (E99), *museum* (E125), *painting* (E130), *UNESCO*

[32]https://www.wikidata.org/wiki/Wikidata:List_of_properties/work#Wikidata_property_related_to_structures

[33]https://www.wikidata.org/wiki/Wikidata:WikiProject_Museums

[34]https://wikidata.org/wiki/User:HakanIST/EntitySchemaList

world heritage site (E142), *hospital* (E187), *public artwork* (E216), *building* (E270), *runic inscription* (E276), and *street* (E317). Let us take a closer look at the two shapes that correspond to two patterns we extract, i.e. *museum* and *painting*. The shape for museums[35] recommends 33 properties: 30/33 are included in optional constraints (*zero or more*, *zero or one*), while only 3 properties are mandatory: *country* (`wdt:P17`), *located in the administrative territorial entity* (`wdt:P131`), *coordinate location* (`wdt:P625`). All 3 mandatory properties are also included in our *museum* pattern. As for the optional properties, we include 2 identifiers and 5 other properties that are also in the shape. Some of the IDs recommended as optional in the shape – such as *ISNI*, *GND ID*, *Commons Institution page*, and *Twitter username* – have a number of occurrences not that lower than the last property we include using the 0.85 threshold, so they would have been selected with a more inclusive threshold. Instead, other IDs are very uncommon, including *Museofile* (0.47%) and *GitHub username* (0.31%), or not used in the data, such as *SUDOC authorities ID*. On the contrary, the shape does not include in its constraints some quite frequent properties of our pattern, like *Freebase ID* (87.74%) and *location* (34.12%).

The shape for paintings[36] is much less rich in number of properties, recommending 4 mandatory (*one or more*) properties – i.e. *title*, *location*, *collection*, *creator* – and one optional (*zero or one*) property, i.e. *inception*. The list of 23 properties of our *painting* pattern also includes all 5 properties of the *painting* Wikidata shape.

5.2.5.3 Music and AAA

The only class, around which we build our patterns, that is common to both domains is `wd:Q5` *human*. The music *human* pattern includes 48 properties, while the AAA *human* pattern includes 33 properties. The two patterns share 27 common properties, that can be indeed mostly considered general, such as *sex or gender*, *occupation*, *country of citizenship*, *given name*. The music *human* recommends 21 additional properties, 11 of which are clearly music-specific, including properties

[35]https://www.wikidata.org/wiki/EntitySchema:E125
[36]https://www.wikidata.org/wiki/EntitySchema:E130

Table 5.9: Comparison between properties recommended by WikiProject Museums and properties included in our pattern for museums.

Property	**Percentage of occurrence**	**WPM**	**EP**
P17 country	99.37%	Y	Y
P625 coordinate location	91.35%	Y	Y
P131 located in the administrative territorial entity	87.11%	Y	Y
P856 official website	66.67%	Y	Y
P571 inception	66.35%	Y	Y
P276 location	34.12%	Y	Y
P6375 street address	18.24%	Y	Y
P18 image	75.79%	N	Y
P1435 heritage designation	16.51%	N	Y
P1619 date of official opening	15.57%	N	Y
P1612 Commons Institution page	8.33%	Y	N
P159 headquarters location	7.86%	Y	N
P1329 phone number	7.86%	Y	N
P968 email address	5.03%	Y	N
P1174 visitors per year	5.03%	Y	N
P669 located on street	4.87%	Y	N
P2851 payment types accepted	2.83%	Y	N
P576 dissolved, abolished or demolished date	2.2%	Y	N
P1037 director/manager	2.04%	Y	N
P2900 fax number	2.04%	Y	N
P154 logo image	1.57%	Y	N
P3025 open days	0.63%	Y	N
P1436 collection or exhibition size	0.31%	Y	N
P2555 fee	0.16%	Y	N
P2846 wheelchair accessibility	0.16%	Y	N
P3026 closed on	0%	Y	N
P3027 open period from	0%	Y	N
P3028 open period to	0%	Y	N

such as *genre*, *instrument*, and *record label*, along with many domain-specific identifiers, like *Spotify artist ID* and *MusicBrainz artist ID*. The AAA *human* pattern includes 7 properties in addition to those shared by humans from both domains, 4 of which are specific to the AAA domain: *RKDartists ID*, *Artnet artist ID*, *Invaluable.com person ID*, and *has works in the collection*. Instead, other properties included in either the music or the AAA *human* patterns are applicable to both domains, such as *Facebook ID* and *official website* (music), *Union List of Artist Names ID* and *award received* (AAA), but they have a higher frequency in either domains.

Let us now have a look at the ranges recommended for some common properties. The property *occupation* has, as common ranges, the classes *occupation* and *profession* in both patterns; however, the more specific range *artistic profession* is recommended for AAA *humans* (45.76%), while the domain-specific range *musical profession* is recommended for music *humans* (83.78%). The property *educated at* has as recommended ranges, along with other general classes, *conservatory* in the music *human* pattern, and *art school* in the AAA *human* pattern. It is evident from these examples that, even in the case of general properties common to patterns from different domains, it is possible to have domain-specific recommendations of ranges.

5.2.5.4 Evolution of music and AAA patterns across two versions of Wikidata

In this section, we report our observations after analysing the patterns extracted on both the music and the art, architecture, and archaeology domains, from two different versions of Wikidata, i.e. the one on which the experiments presented above have been run (which we will call *april version* from now on), and a release dated 6 months later (*october version* from now on). For both domains, the classes selected for building the patterns are the same across the two versions, and the number of instances for each class is almost equal, or slightly different.

Music. 3 out of the 7 patterns extracted from both versions of the Wikidata sub-KG on the music domain have exactly the same set of properties in the two versions, with the same – or slightly different – order of frequency. As for the remaining 4 patterns:

- the *human* pattern includes 2 additional properties in the *october version*, that is two IDs: *National Library of Israel J9U ID* (with an increase of 14.26% of coverage than the *april version*) and *Grove Music Online ID* (+15.47%);
- *musical group* has 2 additional identifiers in the october version: *Musik-Sammler.de artist ID* (+6.58%) and *Bibliothèque nationale de France ID* (+0.18%);
- *album* also recommends 2 more properties (not IDs) than the *april version*: *place of publication* (+10.37%) and *title* (+1.2%);
- the *musical work/composition* pattern is the only one loosing one property from the april to the october version, that is *followed by* (-0.31%), however this is the first most frequent property between the properties excluded based on the threshold.

The properties that have undergone a wider increase in their usage, due to the edits along 6 months, are *National Library of Israel J9U ID* (*human*), *Grove Music Online ID* (*human*), and *place of publication* (*album*).
AAA. Out of the 11 patterns extracted on the art, architecture, and archaeology Wikidata sub-KG, 7 patterns have the same set of properties across the two versions. We list here the remaining 4 patterns with some changes in the recommended properties:

- the *human* pattern has 2 additional identifiers in the october version: *described by source* (+5.08%) and *National Library of Israel J9U ID* (+12.42%);
- *castle* includes 4 additional properties in the later version, that is: *TripAdvisor ID* (+16.36%), *described by source* (+15.67%), *official website* (+3.11%), and *Historic England research records ID* (it was never used for castles in the earlier version);
- *hotel* recommends 4 more properties: *hotel rating* (+4.73%), *Skyscanner hotel ID* (+4.96%), *Agoda hotel numeric ID* (+3.78%), and *Booking.com numeric ID* (+4.72%);

- the *art museum* pattern has one additional property, that is *National Library of Israel J9U ID* (+15.23%).

It is interesting to notice that, for both humans related to the AAA domain and to the music domain, one of the properties that most increased in their usage is *National Library of Israel J9U ID*; moreover, in the AAA domain, this property is more used also for *art museums.* Other properties that have undergone a significant increase are *TripAdvisor ID* and *described by source* (*castle*).

Chapter 6

Conclusion and future work

In this thesis we investigated how to empirically extract ontology design patterns from ontologies and knowledge graphs. The main assumption behind this was that the identified ontology design patterns contribute to support the exploration and understanding of semantic web resources, through a pattern-based view on such resources. Ontology and KG understanding is a key prerequisite for performing multiple ontology and knowledge graph engineering tasks, such as ontology reuse, and for exploiting and reusing such shared knowledge.

The thesis contributes to this goal by proposing two methods for extracting what we term *empirical* ontology design patterns (EODPs) from ontologies and knowledge graphs, respectively.

Extracting EODPs from ontologies, and grouping them in what we name *conceptual components* based on the modelling problems they are addressing, allow us to empirically *observe* the common conceptual issues that an ontology designer may face, and the actual modelling solutions adopted in existing ontologies, regardless their correctness or quality level, or whether they reuse state-of-the-art ontology design patterns. The output of the method, that builds a catalogue of conceptual components from a corpus of ontologies, may be used as a resource for supporting multiple ontology engineering tasks, such as a pattern-based ontology selection and reuse, and ontology interoperability, by proposing possible ontology fragments to align. A better understanding of ontologies is also an access point to the understanding, and possible reuse, of a knowledge graph relying on such ontologies.

However, a knowledge graph does not always use an explicitly and formally modelled ontology, but can be based on an underlying model, defined in a shallow way. This is the reason for the second method that we propose in this thesis. This method takes as input a knowledge graph, which may be related to a specific domain, and extracts the empirical ontology design patterns around the classes that are populated in the KG, generating axioms/constraints involving those classes. These EODPs include data about the probability of such axioms/constraints to *happen*, based on the number of occurrences in the KG in question. Information about the probabilities can be used as a filter to decide how frequent a class or axiom should be to be considered relevant e.g. for reuse purposes. This method allows us to empirically *observe* the main (in term of frequency) EODPs in a (domain-specific) knowledge graph, and how the core classes of such EODPs are *described* with properties, and to which other types of entities are linked, in the actual usage. Besides providing a method for extracting patterns without relying on an ontology, the extracted patterns can be used as an access point to understand the content of the knowledge graph, and to possibly compare multiple knowledge graphs. Both methods have been applied to real-world ontologies and knowledge graphs, and the results of the experiments are presented in the thesis.

To lay the foundations for the proposed methods, and thanks to two important projects that also worked as use cases, we analysed the role of ontology design patterns in the ontology engineering activities, with a specific focus on ontology reuse, that is particularly crucial for guaranteeing the interoperability needed for the semantic web to function how it was conceived. We placed this analysis in the general context of currently implemented ontology reuse approaches.

From our analysis of the state of the art with respect to current approaches for selecting and reusing existing ontologies, presented in Section 2.1, we conclude that there exist several approaches to address these tasks, and that such approaches differ for their motivations (reuse of popular ontologies, reuse of standard ontologies, reuse guided by a cognitive analysis of the requirements), and implementations (direct, indirect, hybrid reuse). All approaches and implementations have both benefits and limits. While reusing popular or standard ontologies can foster interoperability between knowledge graphs relying on those ontologies, and increase the chances that such data will be reused by external applications, popularity-based metrics should not be the only ones to be considered, as the selection of ontologies to be reused should be primarily driven by clear ontological requirements, which not necessarily are compatible with popular/standard ontologies. Directly reusing third parties ontologies can be convenient since it makes it possible to delegate the issue of dealing with ontology preservation, versioning, storage and evolution. However, this creates a strong dependency on the original vocabulary and the semantics adopted there, such that any change in the reused ontologies could introduce inconsistencies contrary to the original requirements. In any case, the most appropriate approach and implementation strategy should be chosen based on the project requirements; however, there is still a lack of clear guidelines for supporting ontology designers in their decision-making when it comes to ontology selection and reuse.

In Section 2.2, we focused on the pattern-based ontology engineering methodology named eXtreme Design. Several experiments have already proven that developing ontologies as compositions of small, reusable building blocks, that is ontology design patterns, makes such ontologies more reusable in different contexts. We contributed to two projects adopting this methodology, and thanks to these use cases we show in this thesis how eXtreme Design supports ontology reuse in actual projects, and how real-world use cases can continuously improve the methodology. Indeed, eXtreme Design provides clear guidelines about the approach to ontology selection and reuse, which should be pattern-based, and always subsequent to the process of collecting requirements as competency questions, and matching such CQs with existing, or new, ontology design patterns. This produces ontologies that are inherently modular, thus easier to understand and reuse. Based on the ontology in question (e.g. whether it is developed by the same ontology design team, or not), direct or indirect reuse is implemented. The two use cases analysed showcase the pattern-based ontology reuse process, and we advance the methodology by (i) refining methods and tools for requirements collection, (ii) proposing an architectural pattern for organising the modules of a big ontology network into a clear structure, and (iii) integrating existing guidelines for ontology and knowledge graph publication and documentation. All these activities are important to perform in order to make the ontology more reusable, and to better reuse external ontologies.

Moreover, the thesis proposes an ontology for annotating ontology design patterns in ontologies, and their instances in knowledge graphs, in order to ease the process of exploring and understanding an ontology and a knowledge graph, and possibly reusing their patterns, both explicitly defined by the ontology designer, or empirically extracted through methods like the ones we present in this thesis.

We perform an evaluation of each component developed, thanks to multiple suitable experiments and use cases that underwent both a manual analysis, performed following specific criteria and measures, and a quantitative analysis: the methods have been evaluated against specific ontology engineering tasks, relevant to our objectives, such as ontology reuse and ontology matching, and have been compared with currently existing resources/guidelines for supporting such tasks.

6.1 How we answered our research questions

In this section we review the research questions presented in 1.2, discussing the answers we formulate to those questions.

RQ1: Are there different modelling solutions implemented in multiple ontologies addressing the same modelling problem? Is it possible to automatically identify and classify them?

This thesis answers to this RQ by developing a method that, after the extraction of what we term empirical ontology design patterns from multiple ontologies, is able to cluster them based on the complex (cognitive) relational structures they implement, in a (more or less) similar way (see Chapter 3).

Splitting an ontology into a set of ontology fragments that are proposing modellig solutions to specific modelling problems makes such ontology more manageable, and easier to understand and reuse, as opposed to a monolithic view on the ontology. Moreover, grouping these fragments (EODPs) occurring in multiple ontologies into clusters (that we call conceptual components) that represent the abstract modelling problems they are all addressing in their specific way, provides a means of accessing one or more ontologies based on the modelling issues they provide solutions for, and of exploring and comparing multiple ontologies at once. This method is evaluated with both manual observations of the results and through experiments against specific ontology engineering tasks that can be supported, starting from two corpora of domain-specific ontologies.

RQ2: What are the relevant concepts and relations to describe the ontology design patterns used in ontologies and knowledge graphs?

The answer to this research question is included in the ontology that we develop, by partially reusing existing ontologies addressing a subset of our requirements, presented in Chapter 4. Indeed, building an ontology by reusing and implementing ontology design patterns, or empirically extracting them from existing ontologies and knowledge graphs, is not sufficient to guarantee that a user that wants to reuse the ontology and/or the knowledge graph actually benefits from the presence of patterns: such resources should be explicitly annotated with the patterns they (un)intentionally implement, and these annotations would have the function of documenting the ontology or knowledge graph, and easing a pattern-based reuse or alignment. Our ontology addresses a number of new requirements not satisfied by previous ontology design patterns' annotation languages: it allows to annotate the ontology with the conceptual components it implements through its ontology design patterns, and to annotate a knowledge graph with the sets of triples complying with a pattern. The usefulness of our ontology is also showcased by three real-world use cases that we report at the end of the chapter.

RQ3: Is it possible to automatically identify the patterns that emerge from a knowledge graph without relying on an explicit ontology?

The question about how to extract patterns from a knowledge graph that does not rely on an explicit ontology, or, even so, without considering the ontology behind it, is answered by the method we develop and present in Chapter 5. Information about the number of instances of the classes in a knowledge graph, and about the occurrences of combinations of subject type (domain), property, and object type (range), is exploited for empirically extracting ontology design patterns that are expressed as sets of axioms associated with a probability based on their frequency. These patterns are also expressed in the form of sets of constraints against which the KG can be evaluated (shapes). Our evaluation, based on the Wikidata KG use case, demonstrates how these empirical ODPs extracted from a knowledge graph can offer a modularised view of such KG, and give insights into its content. Moreover, they provide a user with a tool to choose the level of probability above which a pattern, or an axiom inside it, will be included in the output, generating patterns that are *informed* about the fact that they are *inferred* from a KG based on the actual usage of ontology entities.

6.2 Future work

We conclude this thesis with a discussion on possible lines of future work, some of which are underway at the time of writing.

Comparison and integration of patterns extracted from ontologies and knowledge graphs. In this thesis, we proposed two methods for extracting empirical ontology design patterns. The first works only with ontologies, the second takes into account the knowledge graph exclusively. In the future, we would like to compare the patterns that we extract from ontologies and those that emerge from knowledge graphs, in order to draw conclusions about how much the way modelling issues are addressed in formally modelled ontologies matches with the way knowledge is expressed in the data, i.e. how different are ontology-driven and KG-driven empirical ODPs, in terms of structure and semantics. Moreover, the EODPs extracted from a knowledge graph that is based on an ontology can be compared with the EODPs extracted from such ontology, in order to analyse how an ontology is used in practice. An important future development would be combining the two methods in an overall method that, starting from a knowledge graph based on one ontology, or multiple ontologies, extracts EODPs that take into account both the formally defined ontology, and the statistical information that can be found at the data level.

Comparison and alignment of empirical ODPs and sota ODPs. The two methods to empirically extract ODPs from ontologies and knowledge graphs show their potential to provide a means of observing how an ontology is being modelled, addressing which modelling issues, and how a knowledge graph can shed light on modelling practices put into use at the data level. However, it is to be considered that there are many ontology design patterns explicitly defined, both as independent small ontologies, e.g. published in online catalogues, and included in larger ontologies. A line of research we want to follow in the future is the design of a method for automatically linking empirical patterns to state-of-the-art catalogues' ODPs, and managing a possible synchronous evolution of both resources.

User-based evaluation. As presented in this thesis, and mentioned above, we evaluated our methods for extracting empirical ontology design patterns from ontologies and knowledge graphs with multiple experiments and manual analysis. However, a task-based user study would allow us to gather more evidences to assess the validity of the proposed solutions, and, more importantly, to have external users testing our tools by performing tasks related to the ontology engineering activities that we want to support, like ontology understanding, selection and reuse. In this way, we could prove whether we actually support those tasks, in terms of ease of performing them, saving of time, etc. Moreover, users would provide useful feedback to improve our methods, along with clear improvements we are already working on, like refining the conceptual components, by split or merge, and polishing their naming.

Facing other domains. In our experiments, we focused on some domains of knowledge, for proving the generalisability of our methods. Part of future activities will be dedicated to widening the set of domains, in order to make additional analysis, also expecting that new requirements may emerge from such new experiments. Moreover, we want to test our method for extracting EODPs from knowledge graphs on KGs other than Wikidata.

Bibliography

[1] Renzo Arturo Alva Principe, Andrea Maurino, Matteo Palmonari, Michele Ciavotta and Blerina Spahiu. ‘ABSTAT-HD: a scalable tool for profiling very large knowledge graphs’. In: *VLDB Journal* (2021), pages 1–26. DOI: 10.1007/s00778-021-00704-2.

[2] Flora Amato, Aniello De Santo, Vincenzo Moscato, Fabio Persia, Antonio Picariello and Silvestro Roberto Poccia. ‘Partitioning of ontologies driven by a structure-based approach’. In: *Proceedings of the 9th IEEE International Conference on Semantic Computing (ICSC 2015).* Edited by Mohan S. Kankanhalli, Tao Li and Wei Wang. IEEE Computer Society, 2015, pages 320–323. DOI: 10.1109/ICOSC.2015.7050827.

[3] Luigi Asprino, Valentina Anita Carriero, Christian Colonna and Valentina Presutti. ‘OPLaX: annotating ontology design patterns at conceptual and instance level’. In: *Proceedings of the 12th Workshop on Ontology Design and Patterns (WOP 2021) co-located with 20th International Semantic Web Conference (ISWC 2021).* Edited by Karl Hammar, Cogan Shimizu, Hande Küçük-McGinty, Luigi Asprino and Valentina Anita Carriero. Volume 3011. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pages 1–13.

[4] Luigi Asprino, Valentina Anita Carriero and Valentina Presutti. ‘Extraction of Common Conceptual Components from Multiple Ontologies’. In: *Proceedings of K-CAP ’21: Knowledge Capture Conference 2021.* Edited by Anna Lisa Gentile and Rafael Gonçalves. ACM, 2021, pages 185–192. DOI: 10.1145/3460210.3493542.

[5] Luigi Asprino, Christian Colonna, Misael Mongiovì, Margherita Porena and Valentina Presutti. *Pattern-based Visualization of Knowledge Graphs.* 2021. arXiv: 2106.12857 `[cs.HC]`.

[6] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer and Nathan Schneider. 'Abstract Meaning Representation for Sembanking'. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013.* Edited by Stefanie Dipper, Maria Liakata and Antonio Pareja-Lora. Association for Computational Linguistics, 2013, pages 178–186.

[7] Tim Berners-Lee, James Hendler and Ora Lassila. 'The Semantic Web'. In: *Scientific American 284.5* (2001), pages 34–43.

[8] Diego Berrueta, Jon Phipps, Alistair Miles, Thomas Baker and Ralph Swick. 'Best practice recipes for publishing RDF vocabularies'. In: *Working draft, W3C* 7 (2008).

[9] Eva Blomqvist, Aldo Gangemi and Valentina Presutti. 'Experiments on pattern-based ontology design'. In: *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009).* Edited by Yolanda Gil and Natasha Fridman Noy. ACM, 2009, pages 41–48. DOI: 10.1145/1597735.1597743.

[10] Eva Blomqvist, Karl Hammar and Valentina Presutti. 'Engineering Ontologies with Patterns - The eXtreme Design Methodology'. In: *Ontology Engineering with Ontology Design Patterns - Foundations and Applications.* Volume 25. Studies on the Semantic Web. Amsterdam: IOS Press, 2016. DOI: 10.3233/978-1-61499-676-7-23.

[11] Eva Blomqvist, Pascal Hitzler, Krzysztof Janowicz, Adila Krisnadhi, Tom Narock and Monika Solanki. 'Considerations regarding Ontology Design Patterns'. In: *Semantic Web* 7.1 (2016), pages 1–7. DOI: 10.3233/SW-150202.

[12] Eva Blomqvist, Valentina Presutti, Enrico Daga and Aldo Gangemi. 'Experimenting with eXtreme Design'. In: *Proceedings of the 17th International Conference on Knowledge Engineering and Management by the Masses (EKAW 2010)*. Edited by Philipp Cimiano and Helena Sofia Pinto. Volume 6317. Lecture Notes in Computer Science. Springer, 2010, pages 120–134. DOI: 10.1007/978-3-642-16438-5_9.

[13] Eva Blomqvist, Azam Seil Sepour and Valentina Presutti. 'Ontology Testing - Methodology and Tool'. In: *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*. Edited by Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles and Nathalie Hernandez. Volume 7603. Lecture Notes in Computer Science. Springer, 2012, pages 216–226. DOI: 10.1007/978-3-642-33876-2.

[14] Eva Blomqvist, Ziqi Zhang, Anna Lisa Gentile, Isabelle Augenstein and Fabio Ciravegna. 'Statistical Knowledge Patterns for Characterising Linked Data'. In: *WOP co-located with ISWC*. Edited by Aldo Gangemi, Michael Gruninger, Karl Hammar, Laurent Lefort, Valentina Presutti and Ansgar Scherp. Volume 1188. CEUR Workshop Proceedings. CEUR-WS.org, 2013.

[15] Iovka Boneva, Jérémie Dusart, Daniel Fernández-Álvarez and José Emilio Labra Gayo. 'Shape Designer for ShEx and SHACL constraints'. In: *Proceedings of Posters & Demonstrations, Industry, and Outrageous Ideas - ISWC 2019*. Edited by Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet and Abraham Bernstein. Volume 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pages 269–272.

[16] Freddy Brasileiro, João Paulo A. Almeida, Victorio A. Carvalho and Giancarlo Guizzardi. 'Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata'. In: *WWW '16 Companion*. Edited by Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks and Ben Y. Zhao. 2016, pages 975–980. DOI: 10.1145/2872518.2891117.

[17] Valentina Anita Carriero, Fiorela Ciroku, Jacopo de Berardinis, Delfina Sol Martinez Pandiani, Albert Meroño-Peñuela, Andrea Poltronieri and Valentina Presutti. 'Semantic integration of MIR datasets with the Polifonia ontology network'. In: *Proceedings of Late Breaking Demo Session of the 22nd Int. Society for Music Information Retrieval Conference (ISMIR 2021).* 2021.

[18] Valentina Anita Carriero, Marilena Daquino, Aldo Gangemi, Andrea Giovanni Nuzzolese, Silvio Peroni, Valentina Presutti and Francesca Tomasi. 'The Landscape of Ontology Reuse Approaches'. In: *Applications and Practices in Ontology Design, Extraction, and Reasoning.* Edited by Giuseppe Cota, Marilena Daquino and Gian Luca Pozzato. Volume 49. Studies on the Semantic Web. Amsterdam: IOS Press, 2020, pages 21–38. DOI: 10.3233/SSW200033.

[19] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Valentina Presutti and Chiara Veninata. 'ArCo: The Italian Cultural Heritage Knowledge Graph'. In: *Proceedings of the 18th International Semantic Web Conference ISWC 2019, Part II.* Edited by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois and Fabien Gandon. Volume 11779. Lecture Notes in Computer Science. Springer, 2019, pages 36–52. DOI: 10.1007/978-3-030-30796-7_3.

[20] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Andrea Giovanni Nuzzolese, Valentina Presutti and Chiara Veninata. 'Pattern-based design applied to cultural heritage knowledge graphs'. In: *Semantic Web* 12.2 (2021), pages 313–357. DOI: 10.3233/SW-200422.

[21] Valentina Anita Carriero, Aldo Gangemi, Andrea Giovanni Nuzzolese and Valentina Presutti. 'An Ontology Design Pattern for representing Recurrent Events'. In: *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with the 18th International Semantic Web Conference (ISWC 2019).* Edited by Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda-Villalón, Karl Hammar and Cogan Shimizu. 2019.

[22] Valentina Anita Carriero, Aldo Gangemi, Andrea Giovanni Nuzzolese and Valentina Presutti. 'An Ontology Design Pattern for Representing Recurrent Situations'. In: *Advances in Pattern-Based Ontology Engineering, extended versions of the papers published at the Workshop on Ontology Design and Patterns (WOP)*. Edited by Eva Blomqvist, Torsten Hahmann, Karl Hammar, Pascal Hitzler, Rinke Hoekstra, Raghava Mutharaju, María Poveda-Villalón, Cogan Shimizu, Martin G. Skjæveland, Monika Solanki, Vojtech Svátek and Lu Zhou. Volume 51. Studies on the Semantic Web. IOS Press, 2021, pages 166–182. DOI: 10.3233/SSW210013.

[23] Valentina Anita Carriero, Paul Groth and Valentina Presutti. 'Towards improving Wikidata reuse with emerging patterns'. In: *Proceedings of the 3rd Wikidata Workshop 2022 co-located with the 21st International Semantic Web Conference (ISWC 2022)*. Edited by Lucie-Aimée Kaffee, Simon Razniewski, Gabriel Amaral and Kholoud Saad Alghamdi. Volume 3262. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[24] Valentina Anita Carriero, Paul Groth and Valentina Presutti. 'Empirical ontology design patterns and shapes from Wikidata'. In: *Semantic Web* (2023). under review.

[25] Sejla Cebiric, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou and Mussab Zneika. 'Summarizing semantic graphs: a survey'. In: *VLDB Journal* 28.3 (2019), pages 295–327. DOI: 10.1007/s00778-018-0528-3.

[26] Andrea Cimmino, Alba Fernández-Izquierdo and Raúl García-Castro. 'Astrea: Automatic Generation of SHACL Shapes from Ontologies'. In: *ESWC*. Edited by Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase and Michael Cochez. Volume 12123. Lecture Notes in Computer Science. Springer, 2020, pages 497–513. DOI: 10.1007/978-3-030-49461-2_29.

[27] Aaron Clauset, Mark EJ Newman and Cristopher Moore. 'Finding community structure in very large networks'. In: *Physical review E* 70.6 (2004), page 066111. DOI: 10.1103/PhysRevE.70.066111.

[28] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt and Marta Sabou. 'Criteria and Evaluation for Ontology Modularization Techniques'. In: *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization.* Edited by Heiner Stuckenschmidt, Christine Parent and Stefano Spaccapietra. Volume 5445. Lecture Notes in Computer Science. Springer, 2009, pages 67–89. DOI: 10.1007/978-3-642-01907-4_4.

[29] Marek Dudás, Steffen Lohmann, Vojtech Svátek and Dmitry Pavlov. 'Ontology visualization methods and tools: a survey of the state of the art'. In: *The Knowledge Engineering Review* 33 (2018), pages 1–39. DOI: 10.1017/S0269888918000073.

[30] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz and Francisco M. Couto. 'The AgreementMakerLight Ontology Matching System'. In: *Proceedings of On the Move to Meaningful Internet Systems: OTM Conferences 2013.* Edited by Robert Meersman, Hervé Panetto, Tharam S. Dillon, Johann Eder, Zohra Bellahsene, Norbert Ritter, Pieter De Leenheer and Dejing Dou. Volume 8185. Lecture Notes in Computer Science. 2013, pages 527–541. DOI: 10.1007/978-3-642-41030-7_38.

[31] Daniel Fernandez-Álvarez, Jose Emilio Labra-Gayo and Daniel Gayo-Avello. 'Automatic extraction of shapes using sheXer'. In: *Knowledge-Based Systems* (2021), page 107975. DOI: 10.1016/j.knosys.2021.107975.

[32] Mariano Fernández-López, Asunción Gómez-Pérez and Natalia Juristo. 'Methontology: from ontological art towards ontological engineering'. In: *AAAI Technical Report* (1997).

[33] Pablo Rubén Fillottrani and C Maria Keet. 'Patterns for heterogeneous tbox mappings to bridge different modelling decisions'. In: *Proceedings of the 14th European Semantic Web Conference (ESWC 2017).* Edited by Eva Blomqv-

ist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler and Olaf Hartig. Springer. 2017, pages 371–386. DOI: 10.1007/978-3-319-58068-5_23.

[34] Aldo Gangemi. 'Ontology Design Patterns for Semantic Web Content'. In: *Proceedings of the 4th International Semantic Web Conference (ISWC 2015).* Edited by Yolanda Gil, Enrico Motta, V. Richard Benjamins and Mark A. Musen. Lecture Notes in Computer Science. Springer, pages 262–276. DOI: 10.1007/11574620_21.

[35] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita and Jos Lehmann. 'Modelling Ontology Evaluation and Validation'. In: *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006).* Edited by York Sure and John Domingue. Volume 4011. Lecture Notes in Computer Science. Springer, 2006, pages 140–154. DOI: 10.1007/11762256_13.

[36] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari and Luc Schneider. 'Sweetening ontologies with DOLCE'. In: *Proceedings of the 13th International Conference Knowledge Engineering and Knowledge Management (EKAW 2002).* Volume 2473. Lecture Notes in Computer Science. Springer. Springer, 2002, pages 166–181. DOI: 10.1007/3-540-45810-7_18.

[37] Aldo Gangemi and Valentina Presutti. 'Ontology Design Patterns'. In: *Handbook on Ontologies.* Edited by Steffen Staab and Rudi Studer. International Handbooks on Information Systems. Springer, 2009, pages 221–243. DOI: 10.1007/978-3-540-92673-3_10.

[38] Aldo Gangemi and Valentina Presutti. 'Towards a pattern science for the Semantic Web'. In: *Semantic Web* 1.1-2 (2010), pages 61–68. DOI: 10.3233/SW-2010-0020.

[39] Aldo Gangemi and Valentina Presutti. 'Multi-layered n-ary Patterns'. In: *Ontology Engineering with Ontology Design Patterns - Foundations and Applications.* 2016, pages 105–131. DOI: 10.3233/978-1-61499-676-7-105.

[40] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio and Misael Mongiovì. 'Semantic Web Machine Reading with FRED'. In: *Semantic Web* 8.6 (2017), pages 873–893. DOI: 10.3233/SW-160240.

[41] Soudabeh Ghafourian, Amin Rezaeian and Mahmoud Naghibzadeh. 'Graph-based partitioning of ontology with semantic similarity'. In: *Proceedings of the 3rd International eConference on Computer and Knowledge Engineering (ICCKE 2013)*. IEEE, 2013, pages 80–85. DOI: 10.1109/ICCKE32111.2013.

[42] Jorge Gracia, Elena Montiel-Ponsoda, Philipp Cimiano, Asunción Gómez-Pérez, Paul Buitelaar and John McCrae. 'Challenges for the multilingual web of data'. In: *Journal of Web Semantics* 11 (2012), pages 63–71. DOI: 10.1016/j.websem.2011.09.001.

[43] ICOM/CIDOC CRM Special Interest Group. 'Definition of the CIDOC Conceptual Reference Model'. In: ICOM, 2017, 6.2.2 edn.

[44] Thomas R Gruber. 'A translation approach to portable ontology specifications'. In: *Knowledge acquisition* 5.2 (1993), pages 199–220.

[45] Michael Gruninger and Mark S. Fox. 'The role of competency questions in enterprise engineering'. In: *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*. 1994.

[46] Michael Grüninger and Mark S. Fox. In: *Benchmarking – Theory and Practice*. Edited by Asbjørn Rolstadås. Springer, 1995. Chapter The Role of Competency Questions in Enterprise Engineering, pages 22–31. DOI: 10.1007/978-0-387-34847-6_3.

[47] Tom Heath and Christian Bizer. 'Linked data: Evolving the web into a global data space'. In: *Synthesis lectures on the semantic web: theory and technology*. Synthesis Lectures on the Semantic Web 1.1 (2011), pages 1–136. DOI: 10.2200/S00334ED1V01Y201102WBE001.

[48] Tom Heath and Christian Bizer. ‘Linked data: Evolving the web into a global data space’. In: *Synthesis lectures on the semantic web: theory and technology* 1.1 (2011), pages 1–136.

[49] Martin Hepp. ‘Goodrelations: An ontology for describing products and services offers on the web’. In: *Proceedings of the 16th International Conference of Knowledge Engineering: Practice and Patterns (EKAW 2008)*. Volume 5268. Lecture Notes in Computer Science. Springer, 2008, pages 329–346. DOI: 10.1007/978-3-540-87696-0_29.

[50] Quinn Hirt, Cogan Shimizu and Pascal Hitzler. ‘Extensions to the Ontology Design Pattern Representation Language’. In: *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019)*. Edited by Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda-Villalón, Karl Hammar and Cogan Shimizu. Volume 2459. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pages 76–75.

[51] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi and Valentina Presutti, editors. *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*. Volume 25. SSWS. IOS Press, 2016.

[52] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi and Valentina Presutti, editors. *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*. Volume 25. SSWS. IOS Press, 2016. ISBN: 978-1-61499-675-0.

[53] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Alfa Krisnadhi and Valentina Presutti. ‘Towards a Simple but Useful Ontology Design Pattern Representation Language’. In: *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017)*. Volume 2043. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[54] Bernadette Hyland, Ghislain Atemezing and Boris Villazón-Terrazas. 'Best practices for publishing linked data'. In: *W3C Working Group Note* (2014), pages 1–22.

[55] Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Ronpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe and Pedro Szekely. 'KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis'. In: *Proceedings of the 19th International Semantic Web Conference (ISWC 2020) - Part II*. Edited by Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne and Lalana Kagal. Volume 12507. Lecture Notes in Computer Science. Springer. 2020, pages 278–293. DOI: 10.1007/978-3-030-62466-8_18.

[56] Megan Katsumi and Michael Grüninger. 'What is ontology reuse?' In: *Proceedings of the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016)*. 2016, pages 9–22.

[57] Holger Knublauch. *SHACL and OWL Compared*. https://spinrdf.org/shacl-and-owl.html/. 2017.

[58] Agnieszka Ławrynowicz, Jedrzej Potoniec, Michał Robaczyk and Tania Tudorache. 'Discovery of emerging design patterns in ontologies using tree mining'. In: *Semantic Web* 9.4 (2018), pages 517–544. DOI: 10.3233/SW-170280.

[59] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik and Jun Zhao. 'Prov-o: The prov ontology'. In: *W3C recommendation* 30 (2013).

[60] Ning Li, Enrico Motta and Mathieu d'Aquin. 'Ontology summarization: an analysis and an evaluation'. In: *Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010), co-located with the 9th International Semantic Web Conference (ISWC 2010)*. Edited by Asunción Gómez-Pérez, Fabio Ciravegna, Frank van Harmelen and Jeff Heflin. Volume 666. CEUR Workshop Proceedings. CEUR-WS.org, 2010.

[61] Matteo Lissandrini, Torben Bach Pedersen, Katja Hose and Davide Mottin. 'Knowledge graph exploration: where are we and where are we going?' In: *ACM SIGWEB Newsletter* Summer 2020 (2020), pages 1–8.

[62] James MacQueen. 'Some methods for classification and analysis of multivariate observations'. In: *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*. Volume 1. 14. 1967, pages 281–297.

[63] Antonello Meloni, Diego Reforgiato Recupero and Aldo Gangemi. 'AMR2FRED, A Tool for Translating Abstract Meaning Representation to Motif-Based Linguistic Knowledge Graphs'. In: *The Semantic Web: ESWC 2017 Satellite Events - ESWC 2017 Satellite Events*. Springer International Publishing, 2017, pages 43–47. DOI: 10.1007/978-3-319-70407-4_9.

[64] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro and Asunción Gómez-Pérez. 'Loupe - An Online Tool for Inspecting Datasets in the Linked Data Cloud'. In: *Proceedings of the Posters & Demonstrations Track of the 14th International Semantic Web Conference (ISWC 2015)*. Edited by Serena Villata, Jeff Z. Pan and Mauro Dragoni. Volume 1. CEUR Workshop Proceedings 1. CEUR-WS.org, 2015.

[65] Nandana Mihindukulasooriya, Mohammad Rifat Ahmmad Rashid, Giuseppe Rizzo, Raúl García-Castro, Oscar Corcho and Marco Torchiano. 'RDF shape induction using knowledge base profiling'. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018*. Edited by Hisham M. Haddad, Roger L. Wainwright and Richard Chbeir. ACM, 2018, pages 1952–1959. DOI: 10.1145/3167132.3167341.

[66] Eleni Mikroyannidi, Luigi Iannone, Robert Stevens and Alan Rector. 'Inspecting regularities in ontology design using clustering'. In: *Proceedings of the the 10th International Semantic Web Conference (ISWC 2011) - Part I*. Edited by Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy and Eva Blomqvist. Volume 7031. Lec-

ture Notes in Computer Science. Springer, 2011, pages 438–453. DOI: 10.1007/978-3-642-25073-6_28.

[67] Christine Parent and Stefano Spaccapietra. 'An Overview of Modularity'. In: *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Edited by Heiner Stuckenschmidt, Christine Parent and Stefano Spaccapietra. Volume 5445. Lecture Notes in Computer Science. Springer, 2009, pages 5–23. DOI: 10.1007/978-3-642-01907-4_2.

[68] Silvio Peroni. 'A simplified agile methodology for ontology development'. In: *OWL: - Experiences and Directions - Reasoner Evaluation - 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Revised Selected Papers*. Volume 10161. Lecture Notes in Computer Science. Springer, 2016, pages 55–69. DOI: 10.1007/978-3-319-54627-8_5.

[69] Silvio Peroni, Enrico Motta and Mathieu d'Aquin. 'Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures'. In: *Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008)*. Springer, pages 242–256.

[70] Helena Sofia Pinto and João P Martins. 'Ontologies: How can they be built?' In: *Knowledge and information systems* 6.4 (2004), pages 441–464. DOI: 10.1007/s10115-003-0138-1.

[71] Helena Sofia Pinto, Steffen Staab and Christoph Tempich. 'DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies'. In: *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004), including Prestigious Applicants of Intelligent Systems (PAIS 2004)*. Volume 16. IOS Press, 2004, page 393.

[72] Alessandro Piscopo and Elena Simperl. 'Who Models the World? Collaborative Ontology Creation and User Roles in Wikidata'. In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018).

[73] Seyed Amin Pouriyeh, Mehdi Allahyari, Krys Kochut and Hamid R. Arabnia. 'A Comprehensive Survey of Ontology Summarization: Measures and Methods'. In: *CoRR* abs/1801.01937 (2018).

[74] Valentina Presutti, Enrico Daga, Aldo Gangemi and Eva Blomqvist. 'eXtreme Design with Content Ontology Design Patterns'. In: *Proceedings of the 1st Workshop on Ontology Patterns (WOP 2009), co-located with the 8th International Semantic Web Conference (ISWC 2009)*. Edited by Eva Blomqvist, Kurt Sandkuhl, François Scharffe and Vojtech Svátek. Volume 516. CEUR Workshop Proceedings. CEUR-WS.org, 2009.

[75] Valentina Presutti, Giorgia Lodi, Andrea Giovanni Nuzzolese, Aldo Gangemi et al. 'The Role of Ontology Design Patterns in Linked Data Projects'. In: *Proceedings of the 35th International Conference on Conceptual Modeling (ER 2016)*. 2016, pages 113–121. DOI: 10.1007/978-3-319-46397-1_9.

[76] Kashif Rabbani, Matteo Lissandrini and Katja Hose. 'SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption'. In: *Companion of The Web Conference 2022*. Edited by Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman and Lionel Médini. Volume 2043. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[77] Cássio Reginato, Jordana Salamon, Gabriel Nogueira, Monalessa Barcellos, Vítor Souza and Maxwell Monteiro. 'GO-FOR: A Goal-Oriented Framework for Ontology Reuse'. In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE. 2019, pages 99–106.

[78] Pat Riva and Maja Žumer. *FRBRoo, the IFLA Library Reference Model, and now LRMoo: a circle of development*. Technical report. 2017.

[79] Marta Sabou, Vanessa López, Enrico Motta and Victoria S. Uren. 'Ontology Selection: Ontology Evaluation on the Real Semantic Web'. In: *Proceedings of 4th International EON Workshop 2006 Evaluation of Ontologies for the Web*

co-located with the WWW 2006. Edited by Denny Vrandecic, Mari Carmen Suárez-Figueroa, Aldo Gangemi and York Sure. Volume 179. CEUR Workshop Proceedings. CEUR-WS.org, 2006.

[80] Cogan Shimizu, Karl Hammar and Pascal Hitzler. 'Modular Graphical Ontology Engineering Evaluated'. In: *Proceedings of the 17th International Conference of Semantic Web (ESWC 2020)*. Edited by Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase and Michael Cochez. Volume 12123. Lecture Notes in Computer Science. Springer, Cham, 2020, pages 20–35. DOI: 10.1007/978-3-030-49461-2_2.

[81] Martin G. Skjæveland, Henrik Forssell, Johan W. Klüwer, Daniel P. Lupp, Evgenij Thorstensen and Arild Waaler. 'Pattern-Based Ontology Design and Instantiation with Reasonable Ontology Templates'. In: *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017)*. Edited by Eva Blomqvist, Óscar Corcho, Matthew Horridge, David Carral and Rinke Hoekstra. Volume 2043. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[82] Blerina Spahiu, Andrea Maurino and Matteo Palmonari. 'Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL'. In: *Proceedings of the 9th Workshop of Ontology Design and Patterns (WOP 2018) co-located with the 17th International Semantic Web Conference (ISWC 2018)*. Edited by Martin G. Skjæveland, Yingjie Hu, Karl Hammar, Vojtech Svátek and Agnieszka Lawrynowicz. Volume 2195. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pages 52–66.

[83] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula and Andrea Maurino. 'ABSTAT: Ontology-driven Linked Data Summaries with Pattern Minimalization'. In: *Proceedings of the 2nd International Workshop on Summarizing and Presenting Entities and Ontologies (SumPre 2016) co-located with the 13th Extended Semantic Web Conference (ESWC 2016)*. Edited by

Andreas Thalhammer, Gong Cheng and Kalpa Gunaratna. Volume 1605. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[84] Heiner Stuckenschmidt and Michel C. A. Klein. 'Reasoning and change management in modular ontologies'. In: *Data & Knowledge Engineering* 63.2 (2007), pages 200–223. DOI: 10.1016/j.datak.2007.02.001.

[85] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez and Mariano Fernández-López. 'The NeOn methodology for ontology engineering'. In: *Ontology engineering in a networked world*. Edited by Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta and Aldo Gangemi. Springer, 2012, pages 9–34. DOI: 10.1007/978-3-642-24794-1_2.

[86] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta and Aldo Gangemi, editors. *Ontology Engineering in a Networked World*. Springer, 2012. DOI: 10.1007/978-3-642-24794-1.

[87] Elodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez and Cassia Trojahn. 'Survey on complex ontology matching'. In: *Semantic Web* 11.4 (2020), pages 1–39. DOI: 10.3233/SW-190366.

[88] Rocco Tripodi and Arianna Graciotti. *Software for knowledge extraction from text - context - 1st version (v1.0)*. Deliverable 4.5. Polifonia Grant 101004746, 2022.

[89] Rocco Tripodi, Eleonora Marzi, Andrea Poltronieri, Valentina Presutti, Angelo Pompilio, Antonella Luporini, Peter van Kranenburg and Enrico Daga. *Plurilingual corpora containing source texts in English, French, Spanish and German (v1.0)*. Deliverable 4.1. Polifonia Grant 101004746, 2021.

[90] María Poveda Villalón, Mari Carmen Suárez-Figueroa and Asunción Gómez-Pérez. 'The landscape of ontology reuse in linked data'. In: *Proceedings Ontology Engineering in a Data-driven World (OEDW 2012)*. Informatica, 2012.

[91] Denny Vrandečić and Markus Krötzsch. 'Wikidata: A Free Collaborative Knowledgebase'. In: *Communications of the ACM* 57.10 (Sept. 2014), pages 78–85. DOI: 10.1145/2629489.

[92] Dongkuan Xu and Yingjie Tian. 'A comprehensive survey of clustering algorithms'. In: *Annals of Data Science* 2.2 (2015), pages 165–193. DOI: 10.1016/j.engappai.2022.104743.

[93] Ziqi Zhang, Anna Lisa Gentile, Eva Blomqvist, Isabelle Augenstein and Fabio Ciravegna. 'Statistical knowledge patterns: Identifying synonymous relations in large linked datasets'. In: *Proceedings of the 12th International Semantic Web Conference (ISWC 2013) - Part I*. Edited by Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty and Krzysztof Janowicz. Volume 8218. Lecture Notes in Computer Science. Springer, 2013, pages 703–719. DOI: 10.1007/978-3-642-41335-3_44.