

DELIVERABLE

Ontology transformation tools v1– Report v1

Deliverable number	D3.8
Deliverable name	Ontology transformation tools v1
Work package	WP3
Lead partner	VSE
Contributing partners	LIU
Deadline	2024-08-31
Dissemination level	Public
Date	2024-08-30



Funded by
the European Union

PROJECT INFORMATION

Project summary

Circular economy aims at reducing value loss and avoiding waste, by circulating materials or product parts before they become waste. Today, lack of support for sharing data in a secure, quality assured, and automated way is one of the main obstacles that industry actors point to when creating new circular value networks. Together with using different terminologies and not having explicit definitions of the concepts that appear in data, this makes it very difficult to create new ecosystems of actors in Europe today. This project will address the core challenges of making decentralized data and information understandable and usable for humans as well as machines. The project will leverage open standards for semantic data interoperability in establishing a shared vocabulary (ontology network) for data documentation, as well as a decentralized digital platform that enables collaboration in a secure and privacy-preserving manner.

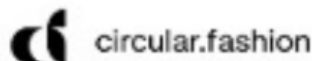
The project addresses several open research problems, including the development of ontologies that need to model a wide range of different materials and products, not only providing vertical interoperability but also horizontal interoperability, for cross-industry value networks. As well as transdisciplinary research on methods to find, analyze and assess new circular value chain configurations opened by considering resource, information, value and energy flows as an integral part of the same complex system. Three industry use cases, from radically different industry domains, act as drivers for the research and development activities, as well as test beds and demonstrators for the cross-industry applicability of the results. The developed solutions will allow for automation of planning, management, and execution of circular value networks, at a European scale, and beyond. The project thereby supports acceleration of the digital and green transitions, automating the discovery and formation of new collaborations in the circular economy.

Project start date and duration

1st of June 2022, 36 months

Project consortium

No	Partner	Abbreviation	Country
1	Linköping University	LIU	Sweden
2	Interuniversitair Micro-Electronica Centrum	IMEC	Belgium
3	Concular Ug Haftungsbeschränkt	CON	Germany
4	+Impakt Luxembourg Sarl	POS	Luxembourg
5	Circularise Bv	CIRC	The Netherlands
6	Universitaet Hamburg	UHAM	Germany
7	Circular.Fashion Ug (Haftungsbeschränkt)	FAS	Germany
8	Lindner Group Kg	LIN	Germany
9	Ragn-Sells Recycling Ab	RS	Sweden
10	Texon Italia Srl	TEXON	Italy
11	Rare Earths Industry Association	REIA	Belgium
12	Prague University of Economics and Business	VSE	Czech Republic



Document Reference

Project acronym	Onto-DESIDE			
Programme	Horizon Europe			
Grant agreement number	101058682			
Project URL	https://ontodeside.eu/			
EU Project Officer	Giuseppina LAURITANO			
Project Coordinator	Name	Eva Blomqvist	Phone	+46 13 28 27 72
	E-mail	eva.blomqvist@liu.se	Phone	
Project Manager	Name	Svjetlana Stekovic	Phone	+46 13 28 69 55
	E-mail	svjetlana.stekovic@liu.se	Phone	+46 701 91 66 76
Deputy PC	Name	Olaf Hartig	Phone	+46 13 28 56 39
	E-mail	olaf.hartig@liu.se	Phone	
Deliverable name	Ontology transformation tools v1			
Deliverable number	D3.8			
Deliverable version	V1.0			
Deliverable nature	Report			
Deliverable level	Public			
Due date	2024-08-31			
Delivery date	2024-08-30			
Keywords	Ontology Transformation, Transformation Pattern, Ontology Alignment			

Document Change Log

Version	Date	Description	Authors	Checked by
0.1	30.07.2024	Initial outline	Ondřej Zamazal	
0.2	6.8.2024	Initial sections draft	Ondřej Zamazal	
0.3	8.8.2024	System Architecture and Installation	Martin Ledvinka	Ondřej Zamazal
0.7	15.8.2024	Added several sections, and restructured	Vojtěch Svátek, Kateřina Haniková, Ondřej Zamazal	Ben De Meester
0.8	28.8.2024	Updated Section Progress in Ontology Alignment	Ondřej Zamazal, Jana Vataščinová, Huanyu Li	
0.9	29.8.2024	Updates based on comments of Ben De Meester	Vojtěch Svátek, Ondřej Zamazal	
1.0	30.8.2024	Preparation of final version	Ondřej Zamazal, Vojtěch Svátek	Ondřej Zamazal

Document Approval

Version	Date	Name	Role in the project	Beneficiary
0.7	22.8.2024	Ben De Meester	Deliverable internal reviewer	IMEC
1.0	30.8.2024	Eva Blomqvist	PC	LIU

Contents

Abbreviations	4
Summary	5
1 Introduction	6
1.1 Motivation	6
1.2 Deliverable Objectives	7
1.3 Deliverable Structure	7
2 Background	8
3 Methodology	11
4 Transformation Patterns: Concepts and Outline of Formal Framework	13
4.1 Notion of transformation pattern	13
4.2 Outline of formal framework for transformation pattern systematization	13
4.2.1 Capturing ontological structures of RDF graphs via PURO	14
4.2.2 Representation of AOS in RDF: examples	15
5 Transformation Patterns: Empirical Results	17
5.1 Class by Attribute Type (CAT)	17
5.2 Object Property Chain Shortcutting Pattern	18
6 Ontology Transformation Tool	21
6.1 How to Use the Application	21
6.2 Installation and System Architecture	23
6.2.1 Architecture	23
7 Progress in Ontology Alignment	25
8 Conclusions	26

Abbreviations

Abbreviation	Explanation
CE	Circular Economy
OWL	Web Ontology Language
CEON	Circular Economy Ontology Network
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
SKOS	Simple Knowledge Organization System
PURO	Particulars vs. Universals and Relationships vs. Objects language
OAEI	Ontology Alignment Evaluation Initiative
W3C	World Wide Web Consortium
MISO	Multiple Indirectly Specified Objects
RML	RDF Mapping Language
JSON	JavaScript Object Notation
REST	REpresentational State Transfer
LLM	Large Language Model
OP	Ontology Pattern
AOS	Atomic Ontological Structures
CAT	Class by Attribute Type
UI	User Interface
MELT	Matching Evaluation Toolkit
IRI	Internationalized Resource Identifier

Summary

This deliverable describes the methodology, conducted research and first results achieved within the newly added WP3 task: T3.5 – Automated on-demand adaptation of ontologies. Additionally, it describes ontology alignment activities following D3.2.

1 Introduction

Semantic interoperability of data is one of the biggest barriers towards data sharing in the Circular Economy (CE) domain. This means that although concrete data formats may be agreed and standardised, at the syntactic level, it is still difficult to interpret the data correctly, and thereby data from different organisations can often not be integrated and used together. The Onto-DESIDe project will provide the technical foundations for semantic interoperability in information flows that has the potential to transform digitalisation and data sharing to support a (more) CE. The project makes use of open standards for semantic data interoperability in establishing a shared vocabulary, i.e., a network of ontologies for data documentation, as well as a decentralized digital platform that enables collaboration in a secure and confidential manner. Ontologies are a key enabler for semantic interoperability since they can provide formal definitions of concepts and their relations, for describing the data to be exchanged. As well as provide a way of expressing alignments, i.e., relations, between existing standards and models. What this project will develop is at its basis a technology for allowing data sharing about materials, components, and products, as well as actors, capabilities and processes, as part of circular value networks (CVNs), at a global scale and across industry domains. Metadata and structures for transforming data into information (semantic descriptions, vocabularies) will be open, and comply with FAIR principles (Findability, Accessibility, Interoperability, and Reusability), to enable the highest possible degree of semantic interoperability and automation in data sharing.

This deliverable ensues with D3.2 (Ontology network architecture, methodology and alignment plan – v.2). The main purpose of D3.8 is to present the first, proof-of-concept, version of software tool capable of *transforming* the structure of ontologies, thus adapting it to better serve particular application scenarios, with the help of *transformation patterns*. Aside the tool itself, the deliverable also describes the first version of transformation pattern library, the empirical analyses and experiments underlying the tool design, and a very early version of a theoretical framework underlying the design of transformation patterns. The deliverable also covers a new increment of T3.3 effort, related to *ontology alignment*. (Notably, ontology alignment and ontology transformation are in a way two sides of the same coin, as the two sides of a complex alignment pattern can be mostly also interpreted as the source and target pattern of ontology transformation.)

1.1 Motivation

In connection with deliverables D3.1 and D3.2 of Onto-DESIDe, the *CEON ontology network* was produced,¹ based on user stories and other requirements, formulated as competency questions (see Appendix B of D3.2). CEON, expressed in the Web Ontology Language (OWL) [3], is supposed to serve as a shared schema for exchanging data between partners in the CE value chains. The data is supposed to have the form of RDF knowledge graphs, which conform the schemas expressed by the ontologies.

The graph model of RDF is favorable with respect to reshaping the data according to the needs of applications. Such reshaping may include diverse operations, e.g., making shortcuts between parts of the graph, deriving 'statistical' triples (e.g., counting the nodes or edges of a certain kind), expressing types as individuals rather than OWL classes, or the like. Notably, while some reshaping can be made directly at the level of instance knowledge graph, typically it also requires additions at the ontology level: whether permanently/globally, or as a temporary/local modification.

Such additions could be carried out fully manually. However, the manual approach would have disadvantages: 1) it would be tedious and error-prone, especially for larger CEON ontologies; 2) it would be demanding for ontology engineers to remain aware of all ontology update options relevant for the given ontology fragment and application use case; 3) there would be no direct relationship between the manual edits of the ontology (in a tool such as TopBraid or Protégé) and the subsequent bulk transformation of the knowledge graph, which would presumably be carried out using the SPARQL language (as UPDATE operations or CONSTRUCT queries).

¹<http://w3id.org/CEON/>

For this reason, the Onto-DESIDe team at VSE started to develop a tool capable of semi-automatically carrying out ontology transformation, with support of declarative *transformation patterns*. This approach has potential for eliminating the drawbacks of the manual approach: 1) the patterns can be applied in bulk, and consistently, on multiple fragments of an ontology; 2) the reusable repertory of patterns may provide alternative transformation solutions the engineer will merely choose from but would not have to invent them anew; 3) ontology transformation patterns can be relatively easily associated with templated SPARQL queries/updates applicable on the instance knowledge graph, thus, again, saving effort and eliminating mistakes.

One of the possible uses of ontology transformation might be to facilitate ontology alignment:² if two ontologies use a different modeling style, they are hard to align using mainstream ontology alignment methods. In such cases, either so called *complex alignments* have to be sought, or one ontology can be adapted to the style of the other using ontology transformation.

1.2 Deliverable Objectives

The main objective of the deliverable is to demonstrate the feasibility of semi-automated ontology transformation in general and its application on Onto-DESIDe ontologies.

The deliverable eventually goes slightly beyond the original plan, in the sense of developing a new ontology transformation tool rather than reusing/adapting existing ones. The reason is that, after their careful examination in connection with the study of Onto-DESIDe ontologies, existing tools (in particular, the original version of the PatOMat tool [33]) were found inappropriate due to reliance on old standards and expectation of too much manual work on the side of knowledge engineers. Analogously, performing all experiments through manual calls of SPARQL processors would be too laborious. Developing a new prototype tool, although strongly inspired by an existing one (PatOMat) in its design, was identified as the best solution.

Nevertheless, the tool itself as well as other results presented in the deliverable are meant as preliminary. More powerful tooling is foreseen as subject of the next iteration of the deliverable, D3.9, which will wrap up the activities in T3.5.

1.3 Deliverable Structure

Section 2 reviews the history of research on the ontology transformation topic, as well as closely related topics including ontology alignment. Section 3 describes the methodology of the research underlying this (and the follow-up) deliverable, primarily with regard to ontology transformation. Section 4 is the initial account of a formal framework for describing ontology transformation patterns, which is to be significantly extended in the course of the project. Section 5 provides the preliminary results of automated sampling of transformation pattern instances and the subsequent manual classification, and explains the essence of the two transformation patterns applied in this first iteration. Section 6 is then devoted to the description of the first version of the prototype tool developed for executing ontology transformation; both the usage and installation of the tool are explained. Unlike the previous sections, Section 7 then switches to the other content thread of the deliverable, that of ontology alignment; the content is only brief and follows up with D3.2.

²As the relevance of ontology alignment for Onto-DESIDe was sufficiently described in D3.2, we do not elaborate on it here any further.

2 Background

Ontology designers usually choose only one of many possible ontological representations of the reality [25]. The different alternatives can be characterized as conforming to certain *modeling styles* or *modeling patterns*. For example, a complex relationship can be expressed as a chain of simpler relationships (e.g., a country is linked by *hasCapital* to a city, which is, in turn, linked by *hasMayor* to a person) or we can make a shortcut using complex relationship (e.g., *hasMayorOfCapital* directly linking a country with a person). The choice of modeling style impacts stream-line applications that consume the ontology and apply its ontological viewpoint, e.g., a knowledge graph *visualization* tool can either be capable of employing the chain property to downsize a diagram or not – depending on whether this property is an explicit part of the ontology.

One solution to the ontology rigidity problem is to enable the *transformation* of an ontology across different modeling styles/patterns. Ontology transformation can be fully automated, or semi-automated, where users can interact. A pre-requisite of automation is the availability of reusable transformation patterns – either a priori, or, potentially, via on-the fly discovery of such patterns by advanced AI methods.

In the remainder of the section, we briefly survey the history of ontology transformation and related areas.

The first, entirely manual approaches to ontology transformation [19] were motivated by the need to eliminate known antipatterns in the ontology, and were largely inspired by analogous approaches from conceptual modeling and databases. The implicit assumption was that there is a single ‘best-practice’ alternative to which the worse or entirely undesirable alternatives are to be transformed.

With the growing support of SPARQL [24] as a query language for knowledge graphs, it became natural to base ontology transformation approaches on this versatile instrument. To lift SPARQL more cleanly to schema transformation, the *SPARQL-DL* language [26] was developed, which operated in the OWL entailment space, defined by its underlying description logic (DL) semantics, rather than on asserted RDF graphs alone. Despite this higher level of abstraction, SPARQL-DL was not a pattern-based instrument.

Probably the first *declarative* approach to ontology (construction and) transformation was then *OPPL* [4], as a pattern-based ontology preprocessor with its associated macro-language. The language allowed to define the selection of relevant entities in the ontology and use them to construct new axioms or remove existing ones. Although the original purpose of OPPL was to enable the enrichment of lean ontologies with axioms in bulk, the advanced version, OPPL-2 [13], allowed to employ multiple variables in one script, thus enabling arbitrary transformations. A certain limitation of OPPL however was its monolithic process that did not allow human interaction while the patterns were instantiated by concrete ontology structures.

The idea of automated transformation of an ontology into its (semantically fully or partially equivalent) *structural variants* fitting particular tasks was probably first coined by Zamazal et al. in the *PatOMat* project [28]. A very simple formalization of the problem was proposed, and a three-step workflow was implemented, consisting of pattern occurrence detection in the source ontology, generation of transformation instructions, and, finally, actual transformation according to these instructions; the latter being largely based on OWL-API.³ An advantage to pure OPPL was the possibility of the knowledge engineer to intervene between the individual steps. The user interaction was supported by a graphical user interface, GUIPOT, and a graphical editor of transformation patterns was also provided [27]. The tools developed in PatOMat were used in a number of pilot use cases [33]. However, the high amount of user interaction in the to-be-transformed entity selection phase as well as the need to manually tune the new entities’ lexical labels (generated via simple lexico-syntactic heuristics) after the transformation prevented the whole approach from larger uptake.

Among the other approaches to the ontology transformation, we can mention *EvoPat* [21], which handled not only the schema transformation but also the instance data transformation. The mechanism used in both was the SPARQL language (using both its SELECT and UPDATE [9] variants). Although its main declared focus was ‘repairing bad smells’ as in the very early approaches, it seems to have been usable for a wide range

³<http://owlcs.github.io/owlapi/>

of transformations, including style-level ones. However, due to its RDF-triple granularity, expressing high-level logical patterns in it would have led to low comprehensibility of the code. It also did not address entity naming (neither in detection nor in transformation).

Aside generic methods specifically tailored for ontology transformation, we can also mention ontology management methods in which a specific kind of transformation is ‘hard-wired’, typically as a smaller component of a more complex task. An example is the use of *meta-modeling* of classes by instances, allowing for checking the semantic coherence of ontologies (with respect to background models such as OntoClean or PURO) using a conventional DL reasoner [10, 29]. A similarly single-purpose transformation is the conversion of OWL taxonomies to SKOS ones [18]; unlike logic-centric meta-modeling (but analogously to PatOMat [28]), the SKOSification of OWL classes also paid considerable attention to the issue of generated lexical labels. Yet another single-purpose transformation was proposed within a larger framework of *focused categorization* [31]. The framework aims at approximating the power of an ontology to provide subcategories for a given (so-called, focus) class, such that the subcategories can be not only explicit (named) subclasses of the class but also compound concept expressions. These compound expressions can be heuristically re-engineered, for example, as new property restriction concepts associated with domain/range and other asserted or inferred restrictions within the original ontology. Aside the calculation of the categorization power itself (primarily serving as guidance on whether the given ontology is to be reused or not), the re-engineered compound expressions can also automatically yield named classes to which they are equivalent. The addition of such a named class and the corresponding equivalence axioms is clearly an ontology transformation operation, which is supported by a software tool called *OReCaP*.⁴

From ontology transformation, we should single out the task of *ontology evolution* [32]. It primarily addresses the detection of changes in the modeled domain and the incorporation of these changes into the ontology. Therefore, compared to the ontology transformation, not only is the *form* (representation, patterns, etc.) altered but the underlying conceptualization has also changed previously. Ontology evolution also inherently includes *ontology versioning*, i.e., the management of different versions of the ontology, including the documentation of changes. The nature of the changes in the ontology is less central here: it could be simply a series of manual edits, as well as some kind of automated transformation, as characterized in the previous part of this subsection.

There is also a close connection between ontology transformation and *ontology alignment*. The corner of ontology alignment that bears the greatest similarity to ontology transformation is that of building *complex alignments* [37], now being also included in the regular OAEI evaluation campaign.⁵ Complex correspondences are such that have, at one or both sides, a compound concept/role expression such as a property restriction, inverse property expression or even a literal-level operation over the value of a data property [35, 36]. A specific representation language and API for complex correspondences, called EDOAL, was developed in 2011.⁶ Obviously, such expression structures can be generalized to recurring *alignment patterns* [22] (also expressible in EDOAL), which can possibly be arranged into a collection of design patterns and picked up by advanced alignment tools. However, it currently seems that finding complex correspondences represents a big challenge for state-of-the-art ontology alignment tools. Within the latest to date (2023) published results of the OAEI campaign, only one of the systems participating in the OAEI complex track was eventually capable of discovering any complex correspondence from the reference dataset, irrespective of its quality.

The ontology alignment patterns closely resemble ontology transformation patterns. The difference is mainly in their usage, which leads to establishing alignments, but does not generate new ontology structures and entity names. Consequently, it is likely that progress in the field of ontology transformation may also stimulate progress in the field of ontology alignment. Namely, advanced use of transformation patterns, bridging the encoding style differences between ontologies, could substitute complex alignment patterns: if one ontology (or its parts) were structurally assimilated to the other, state-of-the-art (such as transformer-based) matchers producing simple correspondences could then be applied. The complex correspondence would then be

⁴<https://fcp.vse.cz/orecap/>

⁵<https://oaei.ontologymatching.org/2023/complex/>

⁶<https://moex.gitlabpages.inria.fr/alignapi/edoal.html>

represented by the combination of the transformation trace and the simple correspondence.

A closely related notion, which is probably not much explicitly distinguished in literature, is what we can call *ontology pattern families*. By this newly coined term we denote a set of alternative ontology design patterns for expressing a certain state of affairs by different means, each having some advantages and disadvantages. Some examples of such material are:

- The W3C's Semantic Web Best Practices and Deployment Working Group pattern(s) on alternatively expressing a set of values of some quantity as a partition of a class or as a set of individuals [20]
- The alternative patterns for expressing a relationship to multiple indirectly specified objects (MISO) [30]; the existence of indirectly specified objects can be expressed, e.g., via an OWL property restriction, but also via a special 'placeholder' individual representing the whole set of unspoken objects, or can even be 'encoded' into the name of an object property.

While the family member patterns are primarily meant as guidance for the ontology engineer at *new ontology design time*, they could also mostly be converted to transformation patterns (to be applied on an existing ontology) or to alignment patterns (to be applied on ontology pairs).

Considering the pattern families, there is no explicit, formal means for expressing the semantic interconnection of their elements. While it is obvious, e.g., for the mentioned W3C's Semantic Web Best Practices and Deployment Working Group pattern [20], that an individual node in the 'value set' pattern semantically corresponds to a class node in the 'value partition' pattern (in the example from [20], the `:good_health` individual by Pattern 1 corresponds to the `:Good_health_value` class by Pattern 2), this is only stated in the pattern text.

Finally, we should mention methods that allow to automatically *generate RDF data from existing resources*. Namely, OWL ontologies can themselves be understood as RDF datasets (consisting of triples). Therefore, RDF generation methods can also be, to some degree, applicable as ontology transformation methods, provided they allow for RDF triples as input, too. As a probably most advanced representative of this family of methods/tools, we can consider *RML* [14] – a powerful mapping language allowing to transform data in various formats to RDF. Similarly to ontology transformation, new terms (IRIs) can be constructed, using one of three methods: through a constant-valued term map (the term is fixed as a part of the mapping rule), through a reference-valued term map (the term is retrieved via a dereferencing method relevant for the source data type, e.g., an SQL ID or an XPath/JSONPath expression; this approach is new compared to the RML's pre-cursor, the W3C-standardized R2RML⁷), or through a template-valued term map (the term is constructed using a string template). RML could presumably be applied for ontology transformation, in the specific cases when their structures consists of a large number of regular fragments (analogous to instance knowledge graph), and the cost of designing single-resource mapping rules would thus pay off.

⁷<https://www.w3.org/TR/r2rml/>

3 Methodology

This section is mainly devoted to the methodology of task T3.5. In this respect, it maps to the early research (and its results) described in this deliverable, and will, unless modified/extended, also apply to the research described in deliverable D3.9 as the next iteration. Given the relatively narrow focus of T3.5, the content of the chapter is relatively brief.

The methodology of T3.5 encompasses the following activities:

1. *Exploration* of Onto-DESIDE ontologies and their prospective usage, in relation to relevance of ontology transformation scenarios
2. Cataloging/adaptation of existing and formulation of new *ontology transformation patterns* at *conceptual level*, in parallel with the development of a general *framework* for the description of ontology transformation patterns
3. Implementation of *operational versions* of chosen ontology transformation patterns
4. *Empirical surveys* of ontology transformation pattern *matches* in ontologies: both in Onto-DESIDE ontologies and in a broader collection of ontologies from existing repositories
5. *Implementation of ontology transformation tool/s*
6. *Evaluation* of the tool/s.

The *exploration* of Onto-DESIDE ontologies is a long-term process, which is not yet reflected in the deliverable. It will require substantial interaction between the team involved in T3.5 and the use case partners. The discussions in use cases have already uncovered some scenarios where ontology transformation could be useful, namely:

1. Simplifying the view of the ontology to the user via various kinds of *shortcutting* (replacing chains of properties by new properties).
2. Bridging between different styles of modeling *roles* of certain objects – since the same physical object can play different roles while moving along a CE chain.

Upon discussing the results from 3.8 (in particular, transformation patterns and their matches in CEON ontologies), we assume that further opportunities for ontology transformation will be identified.

The *cataloging/adaptation of transformation patterns* initially relied on the familiarity of the VSE (and LIU) team with literature on such patterns. This yielded a seed for the collection. The space of transformation patterns was then partitioned using diverse features – for example, whether the pattern increases or decreases the size of the ontology, or whether a certain atomic transformation operation only affects one fragment of the ontology or multiple fragments of the same character. Next, new patterns were derived by altering the existing ones and by exploring not-yet-occurring combinations of features. After reaching a representative sample of patterns, a *formal framework* was outlined, which will allow for even more thorough analysis of the transformation pattern space.

The *operational versions* of transformation patterns⁸ are implemented, relatively straightforwardly, as JSON structures. They are accompanied by templated SPARQL UPDATE operations, whose role is to actually commit the transformation as soon as it is approved by the knowledge engineer.

Empirical surveys have a twofold role. First, they help determine whether the proposed transformation patterns conform to the reality of ontologies and *how efficient* they are. Lack of match either means that the pattern

⁸See <https://github.com/Onto-DESIDE-VSE/TransformationPatterns/tree/main/experiments>

(specifically, its left-hand side) is merely speculative and does not occur in real ontologies, or that there is some inadequacy in its implementation (operationalization), e.g., a redundant axiom is required. Second, the survey results yield sets of matches, which can be (manually, with some automatic support) translated to *training examples for LLM-based tools*, whose role is to generate labels for newly created ontology entities as a result of the transformation.

As regards *tool implementation*, a web application architecture relying on RESTful APIs was chosen. The front-end of the first version of the transformation tool is relatively austere, as it is primarily meant for knowledge engineers performing experiments. The final version is however expected to exhibit ‘softening’ features making it more convenient even for other stakeholders such as subject-matter experts.

For the tool *evaluation*, we foresee both *fine-grained functional testing*, for example, as regards the quality of new entity labels produced by LLMs (as well as more baseline template-based generators) and *overall user experience evaluation*.

Next, we will also very briefly sketch the methodology of the incremental effort in T3.3, related to *ontology alignment*:

1. Compared to the original effort, described in D3.2, *new alignment tools* have been put into place, to improve the solutions of the alignment tasks: *task-a*, *task-b* and *task-c* described in [12].
2. The obtained alignment were validated, for the moment by people without the CE domain expertise.
3. Ongoing work consists in the creation of reference alignments for five pairs of ontologies from task-a.

4 Transformation Patterns: Concepts and Outline of Formal Framework

In this section we first informally explain the structure of ontology transformation patterns, and then present a sketch of a formal framework for their systematization.

4.1 Notion of transformation pattern

In our approach, ontology transformation revolves around the notion of *transformation pattern*.⁹ A transformation pattern technically consists of three parts: source ontology pattern, target ontology pattern, and naming transformation.

An *ontology pattern* (OP) is an OWL ontology fragment with variables, expressible as a set of triple patterns forming a SPARQL query pattern.

A *naming transformation* is a representation of the semantic link between the source and target pattern, in lexical terms. This representation may have various forms, from simple regex operations through complex symbolic templates referencing lexical databases (e.g., WordNet) to pre-trained or prompted large language models.

Regarding instructions dealing with naming, currently, there are two possible parts: *lex* and *ntp*. *lex* contains information about which textual information to use (e.g., by Large Language Model) when a new entity is generated. *ntp* includes instructions on automatically creating textual labels for given entities. Instructions contain a combination of text, variables from triple patterns, and naming instructions applied to variables, e.g., the instruction "label(?A) that label(?p) a noun_form(?C)" says that label naming instruction is applied on variables ?A and ?p, naming operation noun_form is applied to variable C, concatenated with the surrounding text.

The atomic application of a transformation pattern essentially consists in:

1. detecting an instantiation of the *source pattern* in an ontology,
2. adding an instance of the *target pattern* to this ontology (possibly to its new, separately stored version),
3. generating a *label* (and/or possibly) IRI fragment for the newly creating entities, using the *naming transformation*,
4. optionally, *deleting* the instance of the source pattern, or a part of it, from the ontology.

Technical details of the ontology transformation process designed for the purpose of this deliverable is explained in Section 6.1, devoted to the ontology transformation tool.

4.2 Outline of formal framework for transformation pattern systematization

First of all, we should clarify that the main focus of this work is to tackle transformations that not only affect the ontology Tbox (i.e., schema-level axioms, e.g. additions of restrictions) but also change the shape of the instance data (Abox, knowledge graph) in RDF. In the project we primarily address the transformations that do have effect on the instance data structures, i.e., we ignore, e.g., transformations that merely consist in adding a

⁹Several transformation patterns in JSON are available at <https://github.com/Onto-DESIDe-VSE/TransformationPatterns/tree/main/submissions>

logical ‘rule’ to the ontology schema (which only affects inferencing and not the way the Abox can be shaped). This significantly reduces the space we have to explore – although it still remains surprisingly large.

Even if we are aiming at characterizing transformations within OWL ontologies, some aspects of preserving information along the semantic links can be better mapped in a language having greater expressiveness than OWL in some respects. Previous research indicates that the crucial features needed to capture such semantic links are n -ary (with n possibly greater than 2) *relationships* and *higher-order classes*. N -ary relationships cannot be expressed directly in OWL; higher-order classes are only possible in the OWL 2 Full dialect, which is usually being discouraged from due to the undecidability of main description logic reasoning operations. The *PURO modeling language*, in turn, [29] provides these feature while otherwise bringing no conceptual complexity over what OWL has. For this sake, we used it (at least, provisionally), as a starting point for characterizing the OWL transformation patterns. Here we will only explain PURO briefly and informally (full axiomatization is forthcoming).

4.2.1 Capturing ontological structures of RDF graphs via PURO

An analog of an RDF knowledge graph in PURO is called a *PURO model*. A PURO model contains, as building blocks, *objects*, *relationships*, *valuations* and *types*, which we jointly denote as *entities*. Compared to RDF/OWL, which is by itself relatively indifferent to what ontological category an ontology or knowledge graph entity belongs, the best practice in PURO modeling is to adhere to two ontological distinctions as much as possible: that between *particulars* and *universals*, and that between *relationships* and *objects* (or, generally non-dependent entities).¹⁰ Objects, relationships and valuations should, ontologically, correspond to particulars, while the types should correspond to universals.

The *types* can be types of objects (called level-1 types), types of relationships (called relations), types of valuations (called attributes), or types of types (called meta-types or higher-order types).¹¹

A *relationship* is associated with 2 or more individual *roles*, which are played by its participants. Each role has a role type. There can be more roles of the same role type associated with the same relationship. At most one role can be the originating role, for a given relationship.

A *valuation* is an assignment of a *quantitative value*.¹² The value may be a mere number, or a number with a unit of measure.

Now, let an *atomic ontological structures* (AOS) be one of the following:

- An entity *participates* in a relationship, in a certain role.
- An entity *instantiates* a type.
- An entity is *assigned* a *value* (possibly consisting of a numerical aspect and the unit) via a valuation.

Let all mentioned elements of each AOS be called its *formal constituents*. These are, namely:

- For participation: entity, relationship, role.
- For instantiation: entity, type.
- For value assignment: entity, valuation, value, (and numerical aspects and unit, if present).

¹⁰The first letters of these categories actually account for the PURO acronym.

¹¹Note that while we could, at the logical (FOL or higher-order) level, treat types as ‘unary relationships’, we avoid that as it could obscure the central distinction of universals and particulars.

¹²The discussion why PURO only considers quantitative values and whether it limits its expressiveness or not, is beyond the scope of this material.

4.2.2 Representation of AOS in RDF: examples

Let us now analyse a few artificial RDF data examples (in the Turtle syntax¹³), each expressing a similar state of affairs by different means.

1. `:John v:teaches :Peter ; v:teachesSubject :CorporateManagement .`
2. `:Peter v:taughtBy :John .`
3. `:TeachingContract123 v:teacher :John ; v:learner :Peter ;
v:subject :CorporateManagement .`
4. `:TeachingContract123 v:teacher "John456" ; v:learner :Peter ;
v:subject :CorporateManagement .`
5. `:John v:teacherIn :TeachingContract123 .`
6. `:CorporateManagement v:taughtByJohnTo :Peter .`
7. `:CorporateManagement v:taughtByLicenciatedTeacherTo :Peter .`
8. `:John a :ManagementCourseTeacher .`

In analyzing the examples, we will solely focus on one AOS: the *participation* of a person named John in some teaching *relationship*.

In the first example, the participation AOS is reflected in the RDF triple's subject and predicate: the PURO object John corresponds to the triple's subject `:John`, and his teacher (PURO) role is a semantic constituent of the triple's `v:teaches` predicate. The triple's predicate however also bears additional semantics, that of Peter being taught.

The second example is a minor variation, such that John is in the object position in a triple.

In the third example – where the teaching relationship is *reified* to a teaching contract object – we may say, conversely to the previous two, that the predicate of the first triple, `v:teacher`, *fully* expresses the PURO role of Peter's participating, and nothing else.

The fourth example is only a minor variation of the previous one: John is represented by a *string value* instead of an entity with IRI identifier.

The fifth example is similar to the third one, but John is in the subject (hence we do not have a string value variant here – literals cannot be subjects in RDF triples). Note the different *lexicalization* of the predicate; using just the token `teacher` would be confusing here.

The sixth example *encapsulates* John into the predicate name. The whole of participation AOS studied is thus here contained in the predicate alone, and fixed for all of its occurrences.

The seventh example, unlike all previous ones, only expresses the AOS *indirectly*, in the predicate: John as specific instance is only present through its type, which is 'Licenciated teacher'.

Finally, the eighth example uses an *RDF instantiation* – possibly still for expressing a PURO participation rather than a PURO instantiation, provided the RDF instantiation merely means that John is, in this period, active in teaching someone Corporate Management, but he does not have a permanent status, degree, or a similar condition warranting that "Management course teacher" would be his 'natural' type.

¹³For better readability, the IRI of data instances are lexicalized rather than opaque IDs (in most real knowledge graphs, the lexical information would rather be in associated string labels). The underlying namespaces are minimized: the empty prefix for the instance data and `v:` for any vocabulary entities.

In the future, we assume that the framework will be able to capture the space of transformation patterns, among other, depending on whether all constituents of the AOS are preserved or not, whether directly or indirectly, and through what elements of RDF triples.

5 Transformation Patterns: Empirical Results

To see how transformation patterns¹⁴ can be applied to existing ontologies, we launched automated data sampling from ontology repositories, followed by a manual annotating campaign (in July 2024), aiming to label the instantiations of selected transformation patterns. The essence of the atomic annotation task was to:

1. Analyze if the given instantiation of the pattern left-hand side is indeed (semantically) amenable to the pattern.
2. If not, put down an explanation of reasons.
3. If yes, suggest a label for the target entity generated via the transformation pattern.

While the results of the manual process are still to be extended, refined and summarized, in this section we provide the preliminary figures for the automated sampling as well as for the categorization of ‘amenability’ of the pattern occurrences.

As a starting point, we chose two transformation patterns, “Class by Attribute Type (CAT)”¹⁵ and “Object Property Chain Shortcutting Pattern”¹⁶. To gather material for annotations, we tried to detect the (source) pattern within ontologies from the Archivo repository (currently containing 1811 ontologies)¹⁷ and within Onto-DESIDE ontologies.¹⁸

We randomly selected 30% of all ontologies for Archivo in the case of the CAT pattern, i.e., 543, and 20% of all ontologies for the OP Chain Shortcutting pattern, i.e. 362. The OP Chain Shortcutting pattern was more frequently present than the CAT pattern. Therefore, we used fewer ontologies. For Onto-DESIDE ontologies, we downloaded 39 ontologies from the Circular-Economy-Ontology-Catalogue.¹⁹

Each pattern was directly used as a SPARQL query, and we randomly selected at most five pattern matches for each ontology.

With this setting, we found, for the CAT pattern, 56 ontologies in the Archivo sample that corresponded to the SPARQL detection query and 9 ontologies from Onto-DESIDE ontologies. We found 71 ontologies in the Archivo sample for the OP Chain Shortcutting pattern query and 7 in CE-specific ontologies.

The following subsections briefly describe the two transformation patterns and results from an annotation task.

5.1 Class by Attribute Type (CAT)

This pattern is based on the ‘class by attribute type’ alignment pattern from the thesis of F. Scharffe [23]. CAT aims at objects whose values for a certain property are of a *particular class*. This is transformed into a new structure, “class – subclass of – class”. See the example in Figure 1.

A similar pattern, “Class from property value,” differs in that it aims at objects whose value for a specific property is an *instance* of a particular class (in Figure 2). We plan to prepare its transformation pattern JSON implementation and its empirical evaluation within ontologies. Since both patterns are very similar, it would be natural to transform them together. The ontology transformation tool enables loading more transformation patterns at once.

¹⁴Corresponding transformation patterns are available at <https://github.com/Onto-DESIDE-VSE/TransformationPatterns/tree/main/submissions>

¹⁵<https://github.com/Onto-DESIDE-VSE/TransformationPatterns/blob/main/experiments/CAT1.md>

¹⁶<https://github.com/Onto-DESIDE-VSE/TransformationPatterns/blob/main/experiments/pat-op-ch-sh.md>

¹⁷<https://archivo.dbpedia.org/>

¹⁸From the CE Ontology Catalogue at <https://github.com/LiUSemWeb/Circular-Economy-Ontology-Catalogue/tree/main>

¹⁹The Catalogue has currently 46 ontologies. However, several ontologies were not available at the time of our experiment.

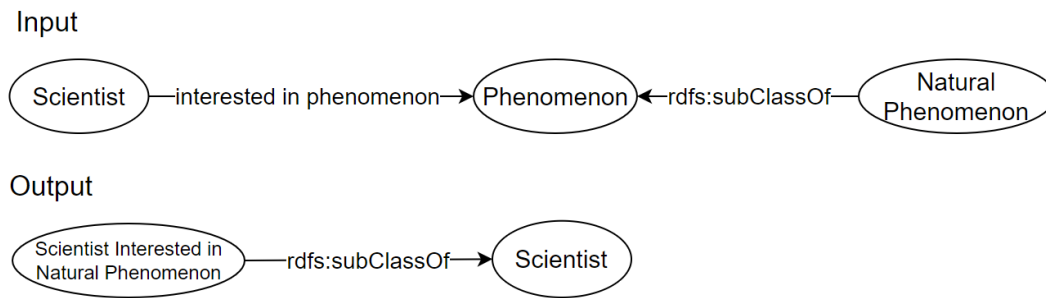


Figure 1: CAT example from the Modern Science Ontology (modsci) [7]

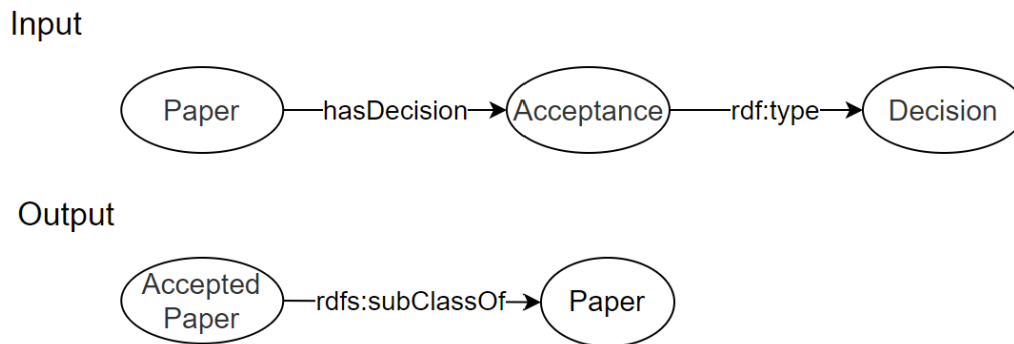


Figure 2: Class from property value pattern

SPARQL query for collecting data was used as followed:

```

SELECT ?A ?p ?B ?C
WHERE {
  ?p rdfs:domain ?A .
  ?p rdfs:range ?B .
  ?C rdfs:subClassOf ?B
}
  
```

The annotation aimed to suggest a new subclass (output in Figure 1). From a sample of 543 ontologies for Archivo, the CAT pattern has been detected in 56 ontologies. Some ontologies have unclear structures, some need more annotation documentation, and some are unsuitable for this pattern. Finally, there have been 30 ontologies having at least one meaningful pattern occurrence. Another example in Figure 3 is from the second dataset focused on Onto-DESIDE ontologies.

Note also the connection to the AOS examples from the previous section. The CAT transformation pattern applied on the `v:teachesSubject` predicate and `pm` individual `:CorporateManagement` being an instance of `v:ManagementCourse`, from the first example, could yield the instantiation to `v:ManagementTeacher` from the eighth example.

5.2 Object Property Chain Shortcutting Pattern

The second pattern is focused on shortening a property path. The goal is to create a shortcut in the property chain by creating a new object property leading from the beginning to the end of a chain. The property can be used as an extension of an ontology, and it does not have to replace the original structure. This shortcut can

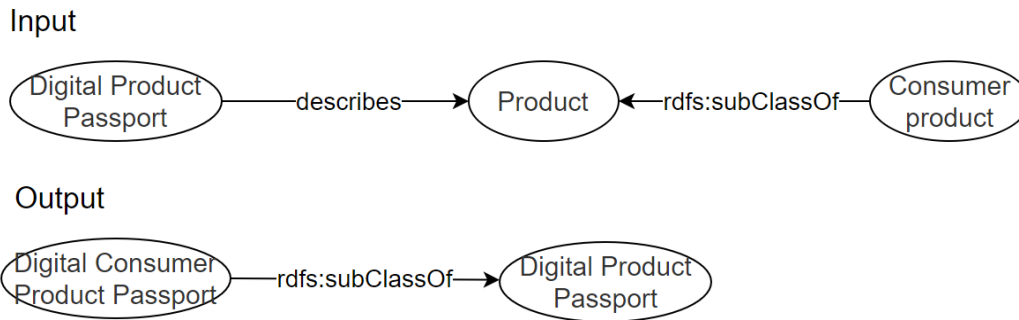


Figure 3: CAT Example from the Digital Product Passport Ontology [15]

lead, for example, to less complicated queries.

SPARQL query used for detecting the pattern (at the level of TBox) is the following:

```

SELECT DISTINCT ?B ?p ?A ?r ?C
WHERE {
  ?p rdfs:domain ?B .
  ?p rdfs:range ?A .
  ?r rdfs:domain ?A .
  ?r rdfs:range ?C .
}

```

The example in Figure 4 is taken from the annotated dataset from Archivo. From a sample of 362 ontologies from Archivo, the OP Chain shortcutting pattern has been detected in 71 ontologies. It turned out that twenty-eight ontologies contain at least one meaningful example of the object property shortcutting pattern. The example in Figure 5 is from the dataset containing Onto-DESIDE ontologies.

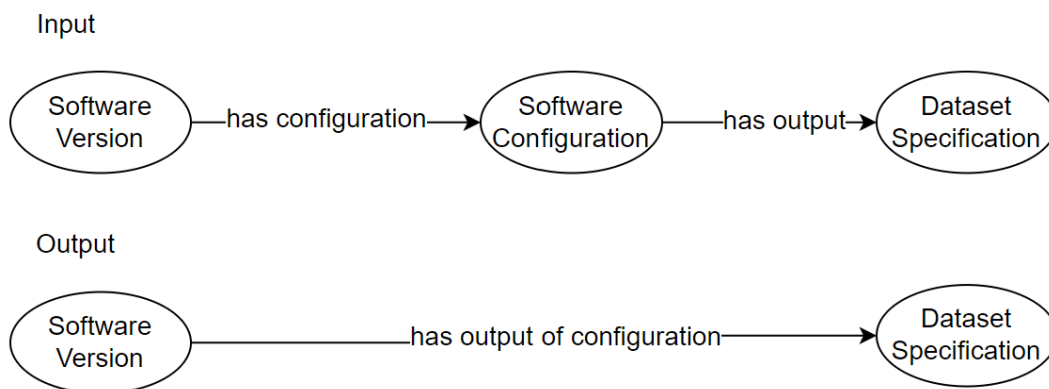


Figure 4: OP Chain Shortcutting Pattern Example from the Software Description Ontology [8]

These empirical results are the first experiments that should verify if we can find real examples of these two initial patterns. The results from these annotating efforts will be used to train some LLM to name new classes or properties automatically.

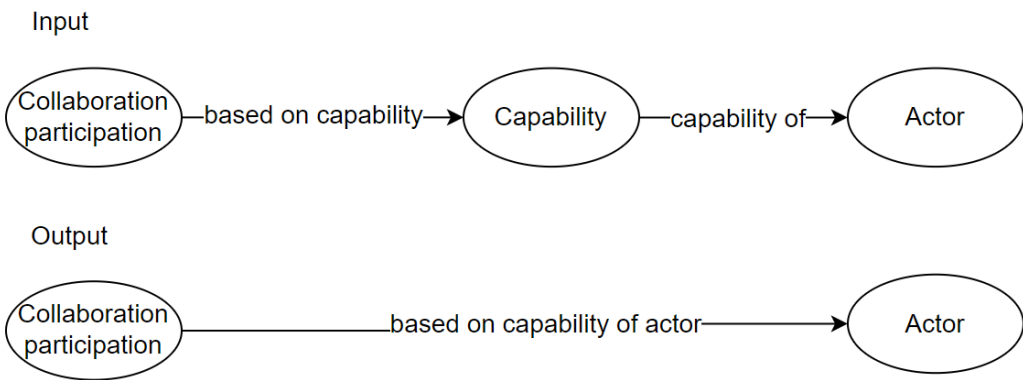


Figure 5: OP Chain Shortcutting Pattern example from the CEON Ontology [1]

6 Ontology Transformation Tool

The PatOMat2 application²⁰ aims at pattern-based transformation of existing ontologies. It allows bridging different modeling styles of web ontologies.

6.1 How to Use the Application

The PatOMat2 application enables the load of ontology and transformation patterns. For example, ontology: *BPO*: Building Product Ontology,²¹ and transformation pattern: *CAT* (Class by Attribute Type).²²

The *op_source* and *op_target* contain an object *triples* containing an object *triple* being an array of individual triple strings.

The *op_source* for the CAT is as follows:²³

```
"op_source": {
  "triples": {
    "triple": [
      "?p rdfs:domain ?A",
      "?p rdfs:range ?B",
      "?C rdfs:subClassOf* ?B"
    ]
  }
}
```

The *op_target* for the CAT is as follows:

```
"op_target": {
  "triples": {
    "triple": [
      "?p rdfs:domain ?A",
      "?p rdfs:range ?B",
      "?C rdfs:subClassOf ?B",
      "?G rdfs:subClassOf ?A",
      "?G owl:equivalentClass _:restriction",
      "_:restriction rdf:type owl:Restriction",
      "_:restriction owl:onProperty ?p",
      "_:restriction owl:someValuesFrom ?C"
    ]
  }
}
```

The first three triples are repeated. There are four new triples in *op_target*. Out of these, three triples deal with an existential restriction.

²⁰<https://owl.vse.cz/patomat2/>

²¹BPO is CE-specific ontology available at <https://annawagner.github.io/bpo/>

²²The transformation pattern is available at <https://github.com/Onto-DESIDe-VSE/TransformationPatterns/blob/main/submissions/CAT1ntp-tp-lite2x.json>

²³It corresponds to the examples of the CAT patterns in Section 5.1.

The CAT also contains the naming transformation property "ntp." The ntp has instructions on how to generate new labels, e.g.,

```
"ntp": {
  "?G": [
    "label(?A) that label(?p) a label(?C)",
    "make_passive_verb(?C) head_noun(?A)"
  ]
}
```

There are naming instructions for the newly generated entity ?G. The number of naming instructions corresponds to several generated labels for the entity ?G. We plan to distinguish between preferred label and alternative labels using *skos:prefLabel* and *skos:altLabel*.²⁴ The instructions include several naming operations:²⁵

- *label(?X)*: getting the label of given entity, otherwise it returns the local IRI fragment
- *nominalize(?X)*: getting the noun form of given entity name
- *passivize(?X)*: changing the tense of the verb of entity name into passive one
- *head_noun(?X)*: getting the head (main) part of the entity name

Initially, the application queries the ontology using SPARQL based on the *op_source* in the uploaded transformation pattern. Inside the *op_pattern* could be used SPARQL specifics. For example, the CAT uses a property path (*rdfs:subClassOf**) to detect indirect subsumptions.

The detection results are displayed as *matching bindings* and *new entities*. For example, we can see one matching binding for each variable (?p, ?B, ?A, ?C) from the *op_source* in Figure 6.

- **p**: <<https://w3id.org/bpo#isComposedOfEntity>>
- **B**: <<https://w3id.org/bpo#Entity>>
- **A**: <<https://w3id.org/bpo#Assembly>>
- **C**: <<https://w3id.org/bpo#DynamicEntity>>

Figure 6: Matching binding for CAT transformation pattern applied on BPO ontology.

New entities are displayed with their randomly generated IRIs and labels generated based on ntp instructions. For example, for ?G variable the label is generated as shown in Figure 7 for the matching binding from Figure 6. Further, the user can directly edit the generated label.

Users can select one or more displayed *pattern matches* to perform *ontology transformation*. Based on the transformation pattern, transformation can be done either by *inserting triples* (option "Apply only inserts") or by *inserting and deleting triples* (option "Apply deletes and inserts"). The information about transformation is presented as *Transformation SPARQL* (for example in Figure 8), where there are DELETE DATA and INSERT

²⁴<https://www.w3.org/2004/02/skos/>

²⁵Naming operations will be available in the following application version. The label function is already available.



- **G:** <<https://w3id.org/bpo/98373>>
 entity that is connected
 from a dynamic entity 

Figure 7: Generated IRI and label for a new entity for CAT transformation pattern applied on BPO ontology.

```
DELETE DATA {
  <https://w3id.org/bpo#isConnectedFrom> rdfs:domain <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo#isConnectedFrom> rdfs:range <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo#DynamicEntity> rdfs:subClassOf <https://w3id.org/bpo#Entity> .
}

INSERT DATA {
  <https://w3id.org/bpo#isConnectedFrom> rdfs:domain <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo#isConnectedFrom> rdfs:range <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo#DynamicEntity> rdfs:subClassOf <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo/98373> rdfs:subClassOf <https://w3id.org/bpo#Entity> .
  <https://w3id.org/bpo/98373> owl:equivalentClass _:restriction .
  _:restriction rdfs:type owl:Restriction .
  _:restriction owl:onProperty <https://w3id.org/bpo#isConnectedFrom> .
  _:restriction owl:someValuesFrom <https://w3id.org/bpo#DynamicEntity> .
}
```

Figure 8: Transformation SPARQL for selected pattern match for CAT transformation pattern applied on BPO ontology. The IRI of new entity is in bold.

DATA parts. However, one can opt out of DELETE DATA parts by choosing the "Apply only inserts" option. It employs *SPARQL Update*.

Finally, the summary is displayed (for example, as in Figure 9) when the transformation is applied on the ontology and the transformed ontology is downloaded.

6.2 Installation and System Architecture

The source code of the PatOMat2 application is available at GitHub²⁶ and is licensed under the MIT license. The tool is a relatively simple web application that is deployable using Docker Compose and is directly on the host system. The repository contains exact instructions and requirements for both modes of operation (i.e., local-based and server-based) and a table describing configuration parameters that can be used to adjust its behavior.

In essence, installation should be as simple as downloading the `docker-compose.yml` file and a proxy configuration directory (`nginx`) to a system with installed Docker Compose and running `docker compose up`.

6.2.1 Architecture

PatOMat2 is a web application with a separate backend and frontend. The backend is written in Java using Spring Boot. Its internal architecture follows a *domain-driven approach* [5] where domain entities contain both data and relevant processing logic. The backend exposes a REST API that the frontend (and possibly other

²⁶<https://github.com/Onto-DESIDE-VSE/patomat2>

Transformation Summary

Number of new entities:	1
Added statements:	<pre> <https://w3id.org/bpo/98373> <http://www.w3.org/2000/01/rdf-schema#label> "entity that is connected from a dynamic entity" . <https://w3id.org/bpo/98373> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <https://w3id.org/bpo#Entity> . _:genid2d2184a4b1161343c39d777880ceb1b585472drestriction <http://www.w3.org/2002/07/owl#someValuesFrom> <https://w3id.org/bpo#DynamicEntity> . <https://w3id.org/bpo/98373> <http://www.w3.org/2002/07/owl#equivalentClass> _:genid2d2184a4b1161343c39d777880ceb1b585472drestriction . _:genid2d2184a4b1161343c39d777880ceb1b585472drestriction <http://www.w3.org/2002/07/owl#onProperty> <https://w3id.org/bpo#isConnectedFrom> . </pre>
	CLOSE

Figure 9: Transformation summary for selected pattern match for CAT transformation pattern applied on BPO ontology.

clients) can use. The backend uses RDF4J [2] (formerly known as Sesame) to work with the ontological data. However, its use is separated from the application logic by generic interfaces that allow, if need be (for example, due to higher expressiveness of the data), switching to a different ontology processing library (like Jena,²⁷ or OWL API²⁸). The frontend is written in TypeScript using Vue with Material UI for some basic styling.

The application does not use authentication and is thus free to use by anyone. To protect against denial of service attacks, the size of the uploaded files and the number of concurrent live user sessions are limited. It ensures that it does not use too many server resources and that new users can work with it only if the session limits are not exceeded. If they are, the user is asked to try again later, when the utilization of the application may be lower. When a user's session ends, all the data uploaded to the server are automatically deleted to free up disk space. The possible number of concurrent sessions and the maximum upload size are configurable as well.

²⁷<https://jena.apache.org/>

²⁸<http://owlcs.github.io/owlapi/>

7 Progress in Ontology Alignment

As a follow-up of D3.2 concerning ontology alignment, we added the following three matching systems: ATBox matcher (ATM) [11], Matcha [6], and LogMapLt [16]. ATM and Matcha achieved good results within previous OAEI editions (for ATM in 2022, for Matcha in 2023). LogMapLt is a lightweight variant of the already involved LogMap matching tool, which efficiently applies string-matching techniques. To contribute to the Onto-DESIDE endeavor in alignment production, we followed the pipeline proposed in D3.2 for three tasks as described in [12]:

- **Task a:** How are existing CE ontologies aligned to each other?
- **Task b:** What are the common concepts between CEON and specific domain ontologies?
- **Task c:** How is CEON aligned to top-level ontologies?

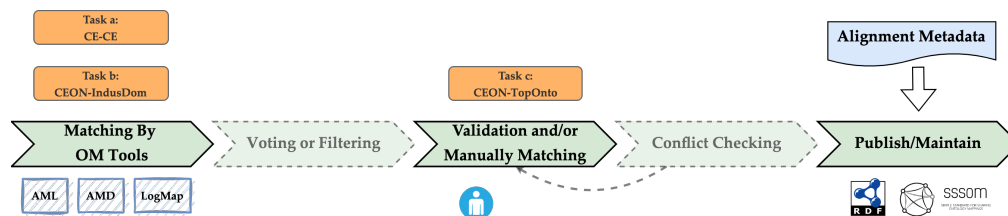


Figure 10: A pipeline producing ontology alignments; originally proposed in [17]

From the pipeline perspective, ATM, Matcha, and LogMapLt are integrated using the MELT client.²⁹ MELT is a matching evaluation toolkit employed in the last several OAEI editions. Therefore, many matching tools could be used and integrated within the pipeline in this way. Corresponding alignment results are published at the Onto-DESIDE GitHub repository.³⁰ We also provided ontology alignment validation for **Task b** and its *Sustainability* domain. However, since one evaluator (non-domain expert) has done it, we potentially should complete it with other evaluators to increase its fidelity.

In collaboration with Linköping University (LIU) and Prague University of Economics and Business (VSE), we prepared a new OAEI 2024 track related to the Onto-DESIDE project, Circular Economy track,³¹ to enhance alignment production further. Initially, we selected five ontology pairs within the **Task a**, which is about matching CE-specific ontologies. We focused on ontology pairs, where CEON is involved. We were working on building reference alignment to enable the automatic evaluation of matching tools within the CE domain. Three evaluators were involved, each manually matching five ontology pairs. Eventually, it turned out that four of five ontology pairs have merely trivial ontology mappings. For the OAEI track, we kept one ontology pair.

So far, we have considered simple ontology alignment, where a single entity from each ontology is involved, e.g., *O1:Document=O2:Manuscript*. However, there can be more difficult interoperability issues involving complex structures. Complex ontology matching is a process of matching complex structures in ontologies. The resulting complex ontology alignment is close to transformation patterns, as mentioned in sections 1 and 2. In [34], we experimented with complex ontology matching using SPARQL and Large Language Models (LLMs) to support an ontology complex alignment production. The proposed approach is based on an alignment pattern structurally detected within the ontology pair on input. Detected complex correspondence candidates are verbalized to be validated by the LLM. While in [34], we provided a zero-shot prompting preliminary experiment and evaluation on ontologies from the conference organization domain, we further plan to apply the approach to Onto-DESIDE ontologies.

²⁹<https://dwsllab.github.io/melt/matcher-evaluation/client>

³⁰<https://github.com/LiUSemWeb/Circular-Economy-Ontology-Catalogue/tree/main/alignments>

³¹<https://oaei.ontologymatching.org/2024/ce/>

8 Conclusions

The deliverable presents the to-date results from task T3.5, on ontology transformation (or, adaptation), as well as a new increment of the research in T3.3, on ontology alignment.

Both work threads will be carried on to the next related deliverable, D3.8. In particular, the follow-up research in ontology transformation will consist of applying the Large Language Model in label generation, improving the tool GUI to fit the lay users, and further ontology matching to support Onto-DESIDe needs.

References

- [1] Eva Blomqvist. Circular Economy Ontology Network (CEON) - Actor ODP, 2024. URL: <https://liusemweb.github.io/CEON/ontology/actorODP/0.2/index.html>.
- [2] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International semantic web conference*, pages 54–68. Springer, 2002.
- [3] World Wide Web Consortium et al. Owl 2 web ontology language document overview. *Word Wide Web Consortium*, 2012.
- [4] Mikel Egaña, Alan L. Rector, Robert Stevens, and Erick Antezana. Applying ontology design patterns in bio-ontologies. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, volume 5268 of *Lecture Notes in Computer Science*, pages 7–16. Springer, 2008. doi: [10.1007/978-3-540-87696-0_4](https://doi.org/10.1007/978-3-540-87696-0_4).
- [5] Eric Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
- [6] Daniel Faria, Marta Contreiras Silva, Pedro Cotovio, Lucas Ferraz, Laura Balbi, and Catia Pesquita. Results for matcha and matcha-dl in oaei 2023. In *Ontology Matching workshop (OM-2023) at ISWC*, pages 164–169. CEUR, 2023.
- [7] Said Fathalla, Christoph Lange, and Sören Auer. An Upper Ontology for Modern Science Branches and Related Entities. In *The Semantic Web*, volume 13870, pages 436–453. Springer Nature Switzerland, 2023. URL: https://link.springer.com/10.1007/978-3-031-33455-9_26, doi: [10.1007/978-3-031-33455-9_26](https://doi.org/10.1007/978-3-031-33455-9_26).
- [8] Daniel Garijo, Varun Ratnakar, Yolanda Gill, and Deborah Khider. The Software Description Ontology, 2021. URL: <https://w3id.org/okn/o/sd/1.9.0>.
- [9] Paula Gearon, Alexandre Passant, and Axel Polleres. SPARQL 1.1 update. W3C recommendation, W3C, March 2013. <https://www.w3.org/TR/2013/REC-sparql11-update-20130321/>.
- [10] Birte Glimm, Sebastian Rudolph, and Johanna Völker. Integrated metamodeling and diagnosis in OWL 2. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2010. doi: [10.1007/978-3-642-17746-0_17](https://doi.org/10.1007/978-3-642-17746-0_17).
- [11] Sven Hertling and Heiko Paulheim. Atbox results for oaei 2022. In *Ontology Matching workshop (OM-2022) at ISWC*. CEUR, 2022.
- [12] Li Huanyu, Eva Blomqvist, and Patrick Lambrix. Experimental ontology alignment results in the circular economy domain. In *The second international Workshop on Knowledge Graphs for Sustainability - KG4S at ESWC*. To Appear, 2024.
- [13] Luigi Iannone, Mikel Egaña Aranguren, Alan L. Rector, and Robert Stevens. Augmenting the expressivity of the ontology pre-processor language. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-432/owled2008eu_submission_16.pdf.

- [14] Ana Iglesias-Molina, Dylan Van Assche, Julián Arenas-Guerrero, Ben De Meester, Christophe Debruyne, Samaneh Jozashoori, Pano Maria, Franck Michel, David Chaves-Fraga, and Anastasia Dimou. The rml ontology: A community-driven modular redesign after a decade of experience in mapping heterogeneous data to rdf. In *International Semantic Web Conference*, pages 152–175. Springer, 2023.
- [15] Maike Jansen, Eva Blomqvist, Huanyu Li, and Robin Keskisärkkä. Digital Product Passport Ontology Design Pattern, 2023. URL: <https://liusemweb.github.io/DPP0/>.
- [16] Ernesto Jiménez-Ruiz. Logmap family participation in the oaei 2023. In *Ontology Matching workshop (OM-2023) at ISWC*. CEUR, 2023.
- [17] Patrick Lambrix and Rajaram Kaliyaperumal. A session-based ontology alignment approach enabling user involvement 1. *Semantic Web*, 8(2):225–251, 2017.
- [18] Nor Azlinayati Abdul Manaf. *Transforming Ontologies in the Web Ontology Language (OWL) to Vocabularies in the Simple Knowledge Organization System (SKOS)*. PhD thesis, The University of Manchester, 3 2015.
- [19] Farhad Mostowfi and Farshad Fotouhi. Improving quality of ontology: An ontology transformation approach. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages 61–61. IEEE, 2006.
- [20] Alan Rector. Representing specified values in owl: "value partitions" and "value sets". W3C recommendation, W3C, May 2005. <http://www.w3.org/TR/2005/NOTE-swbp-specified-values-20050517>.
- [21] Christoph Rieß, Norman Heino, Sebastian Tramp, and Sören Auer. Evopat - pattern-based evolution and refactoring of RDF knowledge bases. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 647–662. Springer, 2010. doi:10.1007/978-3-642-17746-0_41.
- [22] François Scharffe, Ondrej Zamazal, and Dieter Fensel. Ontology alignment design patterns. *Knowl. Inf. Syst.*, 40(1):1–28, 2014. doi:10.1007/s10115-013-0633-y.
- [23] François Scharffe. *Correspondence patterns representation*. PhD thesis, University of Innsbruck, March 2009.
- [24] Andy Seaborne and Steven Harris. SPARQL 1.1 query language. W3C recommendation, W3C, March 2013. <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>.
- [25] Cogan Shimizu, Karl Hammar, and Pascal Hitzler. Modular ontology modeling. *Semantic Web*, 14(3):459–489, 2023. doi:10.3233/SW-222886.
- [26] Evren Sirin and Bijan Parsia. SPARQL-DL: SPARQL query for OWL-DL. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL: <https://ceur-ws.org/Vol-258/paper14.pdf>.
- [27] Ondrej Sváb-Zamazal, Marek Dudás, and Vojtech Svátek. User-friendly pattern-based transformation of OWL ontologies. In Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez, editors, *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, volume 7603 of *Lecture Notes in Computer Science*, pages 426–429. Springer, 2012. doi:10.1007/978-3-642-33876-2_39.

- [28] Ondrej Sváb-Zamazal, Vojtech Svátek, and Luigi Iannone. Pattern-based ontology transformation service exploiting OPPL and OWL-API. In Philipp Cimiano and Helena Sofia Pinto, editors, *Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, volume 6317 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2010. doi:[10.1007/978-3-642-16438-5_8](https://doi.org/10.1007/978-3-642-16438-5_8).
- [29] Vojtech Svátek, Martin Homola, Ján Kluka, and Miroslav Vacura. Metamodeling-based coherence checking of OWL vocabulary background models. In Mariano Rodriguez-Muro, Simon Jupp, and Kavitha Srinivas, editors, *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013*, volume 1080 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013. URL: https://ceur-ws.org/Vol-1080/owlled2013_6.pdf.
- [30] Vojtech Svátek, Ján Kluka, Miroslav Vacura, Martin Homola, and Marek Dudás. Patterns for referring to multiple indirectly specified objects (MISO): analysis and guidelines. In 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, editors, *Advances in Pattern-Based Ontology Engineering, extended versions of the papers published at the Workshop on Ontology Design and Patterns (WOP)*, volume 51 of *Studies on the Semantic Web*, pages 1–24. IOS Press, 2021. doi:[10.3233/SSW210004](https://doi.org/10.3233/SSW210004).
- [31] Vojtech Svátek, Ondrej Zamazal, Viet Bach Nguyen, Jirí Ivánek, Ján Kluka, and Miroslav Vacura. Focused categorization power of ontologies: General framework and study on simple existential concept expressions. *Semantic Web*, 14(6):1209–1253, 2023. doi:[10.3233/SW-233401](https://doi.org/10.3233/SW-233401).
- [32] Fouad Zablith, Grigoris Antoniou, Mathieu d’Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. Ontology evolution: a process-centric survey. *The knowledge engineering review*, 30(1):45–75, 2015.
- [33] Ondrej Zamazal and Vojtech Svátek. Patomat - versatile framework for pattern-based ontology transformation. *Comput. Informatics*, 34(2):305–336, 2015. URL: <http://www.cai.sk/ojs/index.php/cai/article/view/1138>.
- [34] Ondřej Zamazal. Towards pattern-based complex ontology matching using sparql and llm. In *SEMANTICS (Posters & Demos)*. Accepted, 2024.
- [35] Lu Zhou, Michelle Cheatham, Adila Krisnadhi, and Pascal Hitzler. A complex alignment benchmark: Geolink dataset. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018*, pages 273–288, Cham, 2018. Springer International Publishing.
- [36] Lu Zhou, Cogan Shimizu, Pascal Hitzler, Alicia M. Sheill, Seila Gonzalez Estrecha, Catherine Foley, Duncan Tarr, and Dean Rehberger. The enslaved dataset: A real-world complex ontology alignment benchmark using wikibase. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20*, page 3197–3204, New York, NY, USA, 2020. Association for Computing Machinery. doi:[10.1145/3340531.3412768](https://doi.org/10.1145/3340531.3412768).
- [37] Lu Zhou, Élodie Thiéblin, Michelle Cheatham, Daniel Faria, Catia Pesquita, Cássia Trojahn dos Santos, and Ondrej Zamazal. Towards evaluating complex ontology alignments. *Knowl. Eng. Rev.*, 35:e21, 2020. doi:[10.1017/S0269888920000168](https://doi.org/10.1017/S0269888920000168).