EMODnet Thematic Lot n° V – Biology

CINEA/EMFAF/2022/3.5.2/SI2.895681

Start date of the project: 10/05/2023 (24 months)

Operational Phase

Streamlining semantic interoperability [D5.1.1]

The European Marine Observation and Data Network (EMODnet) is financed by the European Union under Regulation (EU) 2021/1139 of the European Parliament and of the Council of 7 July 2021 establishing the European Maritime, Fisheries and Aquaculture Fund.

Disclaimer

Document info

Title (and reference)	D5.1.1 Streamlining semantic interoperability
WP title (and reference number)	WP5- Technical infrastructure
Task (and reference number)	Task 3: develop a complete and robust machine to machine (M2M) interface to transfer data and products in bulk, which is easily accessible for other machines and initiatives
Authors [affiliation]	Marc Portier [VLIZ] Giorgia Lodi [BUP] Peter Thijsse [MARIS] Paul Weerheim [MARIS] Gwenaelle Moncoiffe [BODC] Alessandra Kokkinaki [BODC] Cedric Decruw [VLIZ]
Dissemination level	Public
Submission date	2025-05-09
Deliverable due date	2025-05-09

Contents

Streamlining semantic interoperability

1 Phase V Semantic Alignment

1.1 Intro

Within the 2023-2025 EMODnet Biology Phase V activities a working group was organised to focus on enhancing interoperability between available semantic assets in the Marine Science Domain. This through seeking better alignment of the application and usage of semantic web technology for Linked Open Data (LOD) publication systems.

In the current movement of Open Science, Virtual Research and AI, the requirements for datasets and data products to be FAIR (Findable, Accessible, Interoperable and Reusable) **for machines** is getting more relevant than it always has been. There are many metrics to comply with, but the main one is for data and data products to be ready for (re)use by science and industry. This requires these are provided together with all essential and relevant context: i.e. sufficient information about the origin (provenance), context, and quality, interoperable with domain standards; and depends on the constant attention by data providers, producers, research infrastructures and data aggregators like EMODnet (Biology). In machine-2-machine practice this means annotating them with accessible metadata. The semantic-web approach to this is to expose this information using RDF. This builds connected knowledge-graphs that use URIs to identify each node and arc in it. Making sure that accessing these URIs provides descriptive representations (again in RDF) is what Linked Open Data, LOD-publishing is about.

This group developed its activities within Task 3: develop a complete and robust machine to machine (M2M) interface to transfer data and products in bulk, which is easily accessible for other machines and initiatives which are included in WP5- Technical infrastructure.

1.2 People

This group was built up of experts and stakeholders from various selected organisations:

Name	Affiliation	Comment
Alexandra Kokkinaki	NOC-BODC (UK)	semantic web tech expert, domain expert, SeaDataNet/NERC BODC vocabs collection systems management
Gwenaëlle Moncoiffé	NOC-BODC (UK)	semantic web tech expert, domain expert, SeaDataNet/NERC BODC vocabs collection systems management
Peter Thijsse	Maris (NL)	marine data management domain expert
Paul Weerheim	Maris (NL)	coding, semantic expert, SeadDataNet systems expert
Giorgia Lodi	BUP (IT)	semantic web tech expert
Cedric Decruw	VLIZ (BE)	coding, github workflows, docker
Marc Portier	VLIZ (BE)	systems architecture

1.3 Goals and tasks

To address this broad ambition the group identified these specific targets:

1. Change feeds for semantic artefacts
2. Translation management for the labels of those semantic artefacts
3. Usage guide and recommendations for semantic web technology

Some context and motivation on these:

Change feeds

The group identified a set of existing catalogs and collections of semantic "reference" entities that play a crucial role in the "semantic backbone" for the knowledge graph in the Marine Science Domain, and thus also in EMODnet Biology's dataset descriptions.

- @BODC
  - The NERC vocab server with its reference terms in skos-collections: `http://vocab.nerc.ac.uk/collection/{collection_id}/current/`
- @MARIS
  - The European Directories for Marine Organisations and Environmental Research Projects: `https://edmo.seadatanet.org/` `https://edmerp.seadatanet.org/`
- @VLIZ
  - The register of Marine Regions, and the one of Marine Species `https://marineregions.org/` `https://marinespecies.org/`

These reference terms (among others) provide the different categories or dimensions along which specific datasets and data points can be grouped and organized. As such they provide additional "incidental" machine to machine accessible links between them; just like coincidence in time and location are used to link and compare data. Similarly these extra links allow users to browse, discover and analyse data from apparently related research that might otherwise remain "unrelated" due to whatever distinct organisational, semantic, or technical workflow pipelines they have been using.

This assumes that the data-management platforms for these dataflows offer their end-users an easy way to pick these reference terms in their environment. In turn this requires an up to date availability of these reference terms, preferably with an associated human readable label (or even other textual associated info that can lead to lookup hits).

To enable such remote systems to stay in sync with the latest version of the managed terms it was decided to extend the LOD-systems in scope with a published change-feed. The Linked Data Event Stream (LDES) specification, already in use on the marineregions.org domain, was chosen as the way to encode these. The benefit of one common open standard semantic model for these change-feed was clear to those involved in this work: at client-side syncing with all of these data-systems exposing their change feed with LDES can now be achieved with the same code libraries, and further leverage built up developer experience, and deployment practice.

Translation management of labels

Extending data-management platforms with this ability to "lookup and select the above reference terms" includes thinking about how end-users finally can interact with these. A big part of that involves having the labels for these terms available in a range of human languages, not only in "English". While the latter assumption and mono-language-culture is rather established within academic circles, it deserves being

challenged, especially regarding its unintended exclusion of non academic participants and stakeholders into research activities and results. Both “citizen science” and “open science” movements raise proper attention to this.

Already leveraging the benefits of the shared change-feed standard (LDES) introduced above we have taken up the development of open source components that:

- can be hosted on free hosting platforms to
- orchestrate the translation work
- of the mentioned reference terms
- between multilingual groups of experts (in loose online groups)
- itself resulting in producing similar standard change-feeds

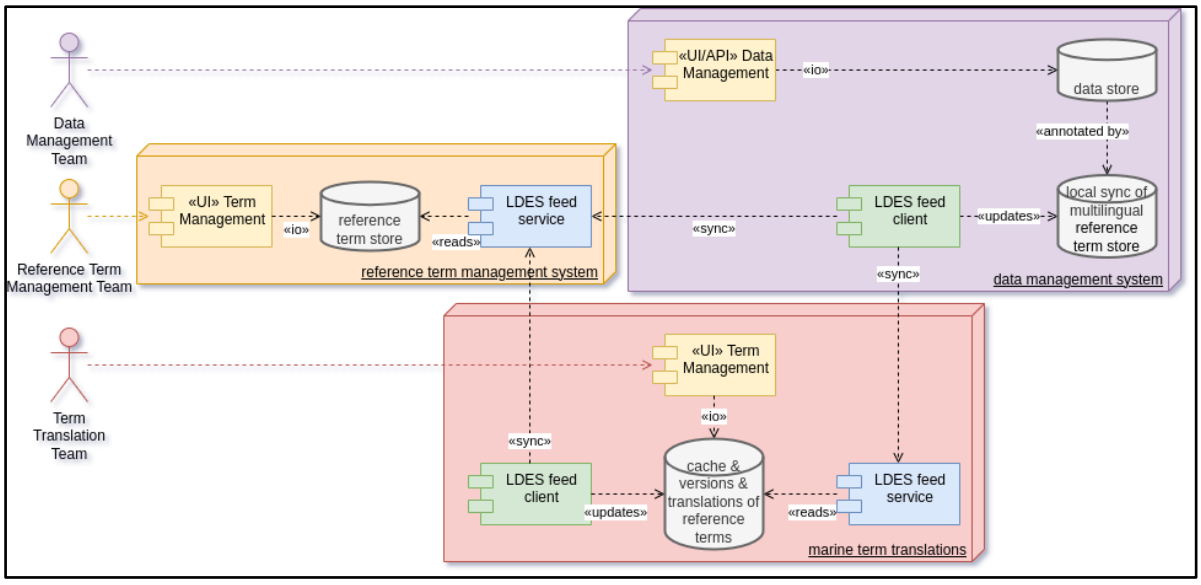

Figure 1. Technical diagram of the full data-flow of reference terms from management, via optional translation management to annotation usage in data management systems.

Usage guide on «Semantic Web Technology»

The integration, interpretation, and exchange of diverse datasets are central challenges in marine research, given the variety of disciplines, institutions, and countries involved. Semantic Web technologies offer powerful solutions to these challenges by providing a common framework for linking and understanding data across systems. Within this context, EMODnet Biology plays a key role by relying on and contributing to the development of a marine knowledge graph that builds on Semantic Web standards.

Therefore, together with the above very practical dataflows of semantic reference terms (and their translations) the working group has set out to provide a more general guide and selected best-practices from the semantic web domain to further enforce the alignment and share expectations around this technology in the domain of Marine Research.

The aim of this guide is to identify the foundational principles and technical components that underpin the Semantic Web, and, where relevant, to contextualise them within marine data management. It also seeks to cover selected methodological and technical aspects — such as the use of unique identifiers, shared data standards, formal semantics, and querying mechanisms — that are critical to applying Semantic Web technologies effectively.

This work will also serve as a foundation for assessing the current level of adoption and usage of Semantic Web technologies within the EMODnet Biology initiative. By doing so, it will help identify potential gaps,

areas for improvement, or opportunities to further enhance semantic integration and interoperability. Where appropriate, this can also help identify recommendations and considerations related to governance, planning, and long-term sustainability.

2 LDES implementations and deployments

2.1 Intro

As touched upon earlier the "Linked Data Event Stream" (LDES) specification allows to describe a so-called "change-feed" of any set of disclosed semantic entities. This boils down to a particular view on the full set of said entities that is optimised to allow synchronisation of external copies of them. This is achieved through a straightforward fragmentation of the full set ordered by their last modification time, with all pages in this view disclosed as a linked-list that can (if needed) be navigated down to older changes.

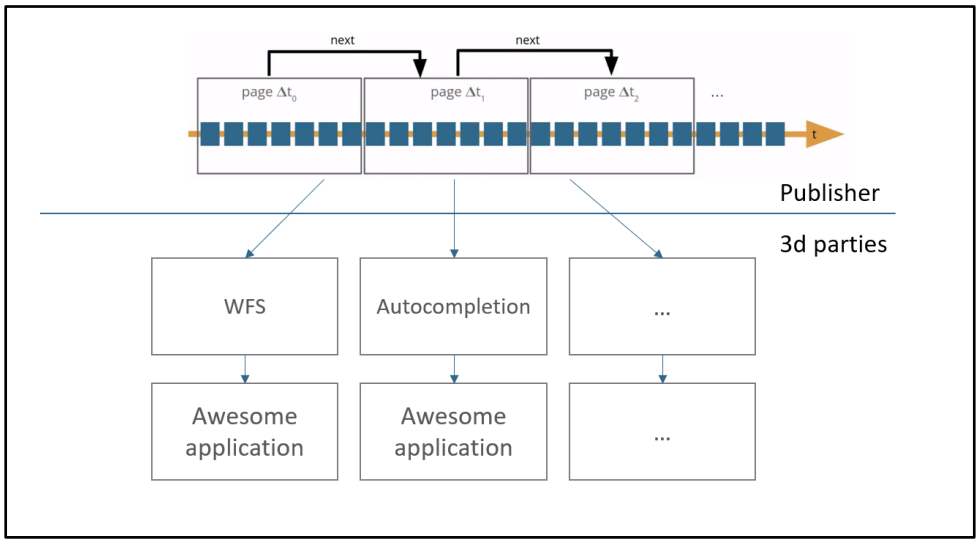

Figure 2. Diagram showing how publishing an LDES change-feed serves as a low cost effective mechanism to synchronise any number of dependent applications. (source: semic.eu - https://interoperable-europe.ec.europa.eu/collection/semic-support-centre/linked-data-event-streams-ldes)

The fact that the same structure (or model) of communicating "grouped changes in time" is applicable to very diverse actual models of the content-entities being communicated is a direct consequence of the general applicability of the RDF semantic model. In practice this domain-agnosticism allows the exploitation of the same components to actually establish the needed synchronisation of these reference terms in other data systems. This cross-domain reuse leads to lowering the cost and increasing the experience built up.

LDES is described at https://w3id.org/ldes/specification and has been adopted by Semic-EU and applied in the MareGraph project which influenced this work.

2.2 LDES feed Realisations

Within this line of work, the group successfully disclosed and consumed the LDES feeds of the selected sets of reference terms:

2.2.1 BODC

Individual LDES feeds have been established for the various available SKOS Collections at the NERC vocab server (NVS) as listed at `https://vocab.nerc.ac.uk/collection/`. These collections group a set of similar concepts and are generally referred to by their own "collection_id" (e.g. the 'P01' collection). The member-concepts in them are the actual reference terms that are managed by BODC, semantically exposed, and identified by their own URI (e.g. `https://vocab.nerc.ac.uk/collection/S25/current/BE006521/`). The grouping by collection_id is chosen to be fed into this URI-template for their corresponding change-feed in LDES format:

`http://vocab.nerc.ac.uk/ldes/{collection_id}/`
→ e.g. `http://vocab.nerc.ac.uk/ldes/P01/`

At BODC the technical architecture for their LOD exposure is organised through a central triple-store. This has led to an implementation that is generating the LDES fragments through querying its SPARQL endpoint. Since these fragments are meant to be immutable, they are simply materialised in static files (in RDF turtle format), stored on disc and are served directly from there. An open-access variant of the script producing these fragments is available at github in `https://github.com/vliz-be-opsci/BODC-ldes-demo`

During the collaboration in this project we also identified the opportunity in updating the references between BODC's species collection (S25) and the recent linked open data publication of WoRMS through aphia.org.

e.g. the turtle representation of

```
http://vocab.nerc.ac.uk/collection/S25/current/BE006521/
```

now contains

```
<http://vocab.nerc.ac.uk/collection/S25/current/BE006521/>
        rdf:type                skos:Concept;
        owl:sameAs              <https://aphia.org/id/taxname/137102> .
```

2.2.2 MARIS

At MARIS side, the available semantic representations of SeaDataNet's European Directories for Marine Organisations (EDMO) and Marine Environmental Projects (EDMERP) have been selected to be extended with an LDES feed.

The exposed entities of these directories are identified with URI following these patterns:

- (for organisations in EDMO) `https://edmo.seadatanet.org/report/{edmo-id}`
- (for projects in EDMERP) `https://edmerp.seadatanet.org/report/{edmerp-id}`

Their change-feeds are now available at:

```
https://edmo.seadatanet.org/ldes/feed
https://edmerp.seadatanet.org/ldes/feed
```

Apache Jena Fuseki is used as the triplestore behind the SPARQL endpoints of the SeaDataNet services. Ontop (https://ontop-vkg.org/) is employed to map a relational database to RDF triples, which are then loaded into Fuseki. To generate the LDES feed, the same R2RML mappings (https://www.w3.org/TR/r2rml/) are used to produce immutable fragments. These fragments are stored on disk in Turtle format.

During the development of the LDES feed, an inconsistent use of the SDN ontology URL link-def.seadatanet.org was identified. After discussing this issue within the working group, it was agreed to correct this and publish an official ontology definition in Turtle format, which will be hosted at ontology.seadatanet.org. After this work has been done the Ontop RDF mappings for CDI, EDMERP & EDMO have been updated accordingly.

2.2.3 VLIZ

Over the period of this project, in parallel, new and updated LDES feeds have been provided for the reference datasets Marine Regions (mr) at http://marineregions.org and Marine Species (aphia) at http://marinespecies.org. For these collections the main concepts (reference terms) are identified with URI following these patterns:

- (for geographic marine regions in mr) `http://marineregions.org/mrgid/{mrgid}`
- (for taxa names in aphia) `https://aphia.org/id/taxname/{aphia-id}`

These feeds are available at:

```
https://marineregions.org/feed
https://aphia.org/feed
```

and are produced directly from the backend (SQL server) of their respective management systems. While the custom made implementation code is not open, the design, architecture and implementation have been described in open access publications:

- Publishing the Marine Regions Gazetteer as a Linked Data Event Stream, Lonneville et al.
- Maregraph Deliverable D5.1 - WoRMS/LOD Gap Analysis

2.2.4 LDES Registration updates

At https://imec-int.github.io/ldes-registry/, an experimental register of available LDES feeds has been set up. This register functions as a helpful resource for the LDES community where the listed feeds of working examples are used as guiding examples, a basis for benchmarking reports, and a diverse test-base to verify compatibility of own approaches for consuming ldes- feeds.

The feeds developed in this work, have been submitted for addition to this registry through pull-requests: #23, #24, #25

2.3 Testing and consuming LDES

2.3.1 VSDS Testbed

VLIZ has internally set up an instance of the TestBed-Shacl-Validator provided by Informatie-Vlaanderen. The associated documentation of that project allows anyone with some basic knowledge of git, docker, and classic configuration file editing to set up their own instance.

This code repository provides a docker stack of which the combined services provide:

(1) the eu-itb (EU Interoperability testbed) a generic web based UI and backe-end for managing and executing test cases and delivering conformance declarations.
(2) a triple store (graphdb) that can be used to SHACL-validate the content of any managed graphs
(3) and the Linked Data Interactions Orchestrator (LDIO) a generic dataflow engine with special support for linked-data, including an LDES-client capable of syncing a change-feed into a triple store.

On this generic platform a custom test-scenario is configured that will effectively use the above to

(a) initiate a dialogue that asks for ldes-feed uri and uploaded SHACL file
(b) initially fetch (and then keep in sync) the dataset through its ldes-feed
(c) store that into the triple store
(d) and use that storage to validate the SHACL-conformance

The open-access configuration and build-code for the above test-scenario is maintained at https://github.com/MareGraph-EU/itb-testbed-config

Figure 3. screenshot of the interaction dialogue at the start of a specific test

This internal testbed-instance will keep track of every executed test case and allows exporting a PDF report for each of them.

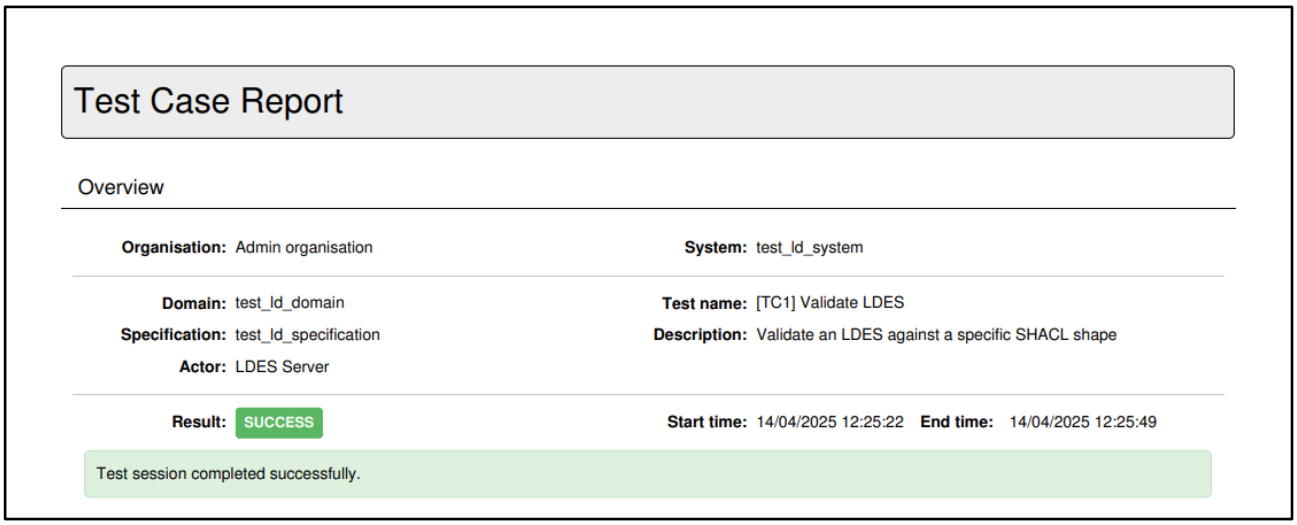

Figure 3. Screenshot of a test-conformance statement, exported as PDF

With the local instance at VLIZ, the following list of conformity checks for the LDES-feeds in scope was performed:

Table 1. Executed tests on the VLIZ testbed instance

source	LDES feed	SHACL applied
mr	https://marineregions.org/feed	https://github.com/lifewatch/marineregions-ontology/blob/master/shapes.ttl
aphia	https://aphia.org/feed	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/aphia_shapes.ttl
P01	http://vocab.nerc.ac.uk/ldes/P01/	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/BODC_concepts_ldes_shapes.ttl
P02	http://vocab.nerc.ac.uk/ldes/P02/	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/BODC_concepts_ldes_shapes.ttl
P06	http://vocab.nerc.ac.uk/ldes/P06/	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/BODC_concepts_ldes_shapes.ttl
S25	http://vocab.nerc.ac.uk/ldes/S25/	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/BODC_concepts_ldes_shapes.ttl
C19	http://vocab.nerc.ac.uk/ldes/C19/	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/BODC_concepts_ldes_shapes.ttl
edmo	https://edmo.seadatanet.org/ldes/feed	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/edmo_seadatanet_ldes_shapes.ttl
edmerp	https://edmerp.seadatanet.org/ldes/feed	https://github.com/vliz-be-opsci/shacl_shape_ldes_feeds/blob/main/edmerp_seadatanet_ldes_shapes.ttl

2.3.2 More LDES tools

Next to the above-mentioned testbed and ldes-registry, there are a number of useful tools to handle LDES and assist in inspection and conformity validation.

- The on-line LDES explorer at https://xplorer.ajuvercr.be/ allows to navigate (one by one) the linked fragments of an LDES feed, and see which members are in each of them.
- The aforementioned LDIO project from "Agentschap Digitaal Vlaanderen", the technical 'digital department' of the Flemish government is by itself a very comprehensive library various linked-data components (including an LDES client) that can be glued together in a handy orchestration framework. Its open codebase is written in Java and based springboot. It is available on github. Similarly, the docker images produced from it can be retrieved from the ghcr.io repository
- RDFConnect from IMEC/UGhent is a similar alternative. Also an RDF focussed orchestration framework, with its own mix of useful components, one of which is also an ldes-client. Open access code at github.

2.4 Lessons learned

Implementation of these LDES feeds has been fairly straightforward, as a starting point it requires basic web-development knowledge together with some experience with producing RDF serialisations: producing either json-ld, turtle, or preferably both is quite achievable through the many available (for most programming languages) RDF-libraries. Conceptually however, the backend data system needs to be able to deliver the basic model making up the LDES feed: fragments of timed changes to entities. For that, at a minimum, all concepts to be exposed in the feed need to have a clear tracking of last-modification, and some way of selecting ordered and paged lists of them based on this timestamp. Obviously, taking some considerations about how many and which LDES feeds to expose is an essential part of the early design. This also includes the need for design decisions concerning the URIspace of the exposed semantic assets: the feed, its fragments, and the members in there. We refer to the ldes-registry (or our own implementations) as a source of inspiring examples for those.

The LDES specification itself is relatively young and still in the status of “Draft Community Report”. While the changes to it are increasingly of the type “minor impact”, the shared understanding and acquired “best practices” around practical application are still in a rather small group, and in active expansion. One example of this during the work done within EMODnet Biology has been the late appreciation (and importance for performance in consumption) of applying proper annotation of the type and associated values of the tree:relation between the various fragments.

Similarly the number of tools that are ready for use are still rather limited, their early releases are mainly focused on the “good weather” cases where everything just works, less so on providing comprehensive logging output when they don’t. This makes these tools in their current state less useful in a context of debugging one's own development. A concrete manifestation of this observation is in the testbed: When reporting “success” the offered assurance and relief is as clear as genuine. However, as long as that level is not reached, the search for possible causes and fixes is not helped by the current report nor anything else in the UI. This quickly leads to growing a behind-the-scenes view into the system that is of a more intimate nature then one would hope to need.

3 Term Translation Flow Management

3.1 Conceptual Systems and Interface Design

The goal of this part of the project was to provide a flexible and open system to enable small communities of multilingual experts to collaborate on providing linked, but separately managed, human language translation for certain reference terms.

As pictured in Figure 1 earlier, the upfront choice was to leverage the earlier LDES creation. This in both directions:

- LDES consumption means a common ingest and sync approach can be used across all the sources;
- LDES production of translations as they become available makes that to external applications the translated labels can be as easily integrated as the original ones.

3.2 Practical use case : vocab-search

The driving use case guiding this development and its design has been to ingest both terms and their translations into the VLIZ vocab server. This system ingests and indexes reference terms, and provides plugable web-widgets to perform the lookup into this index. Its capability to ingest from LDES feeds, makes it a perfect platform to test and showcase this approach.

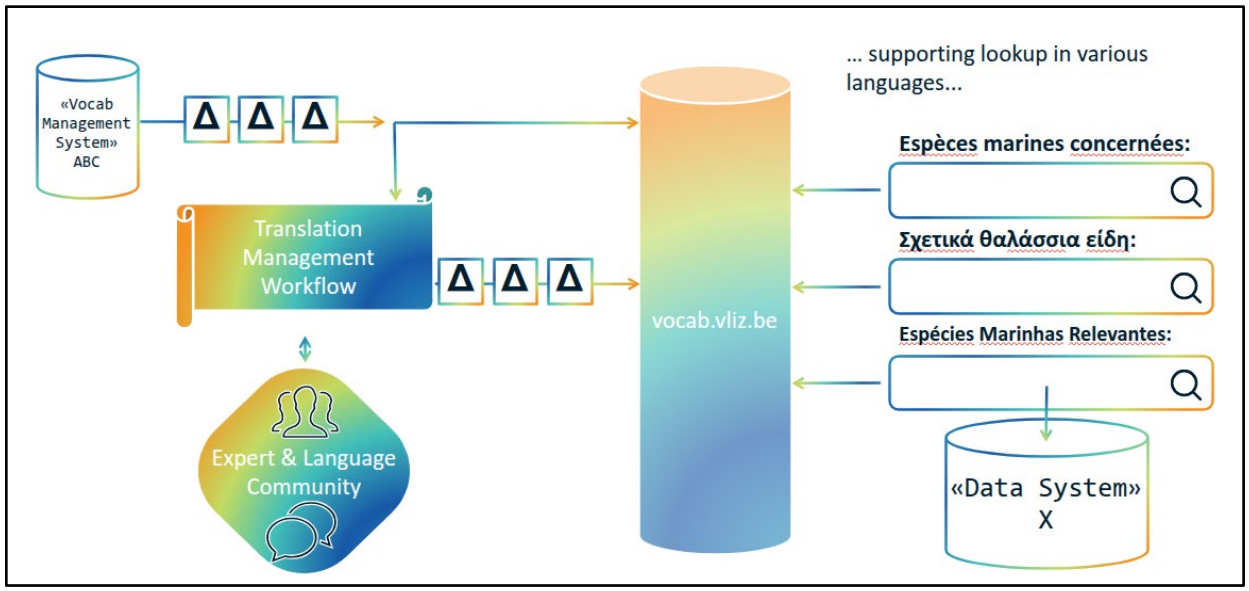

Figure 4. Reuse of LDES ingest to support multilingual term-lookups (from SEMIC-EU slide-deck)

3.3 Chosen implementation strategy

To coordinate the actual translations it was decided not to build a custom managed platform. Instead we designed a text based (yml) file representing individual entities requiring translation that leverages git features at its core to deal with tracking changes, coordinating work into branches, resolve conflicts, and provide an intrinsic provenance trail of what happened in the process.

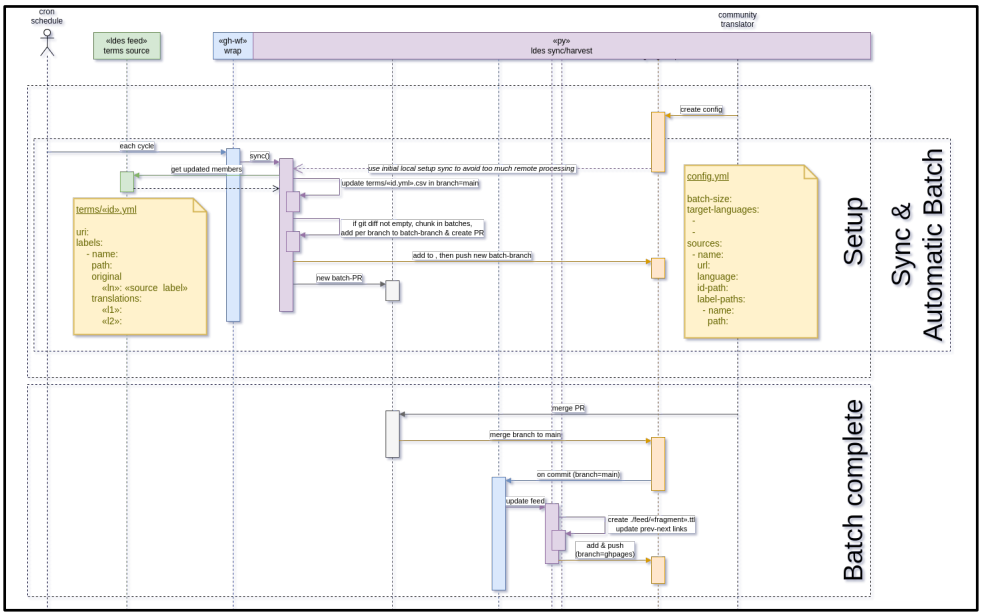

Figure 5. Interaction diagram of the basic git-based translation change-management and coordination

This allowed us to focus primarily on the development of code to deal with the LDES interfaces. As mentioned these cover:

- the ingest/consumption from configured (feed-)sources and the conversion to our yml-format
- the production of the resulting LDES feed containing the available new translations

Still, as we expect no git proficiency from our translations teams, we developed a custom user interface to assist in the actual translation work, hiding git concepts where it was possible. Screenshots of the basic operations below:

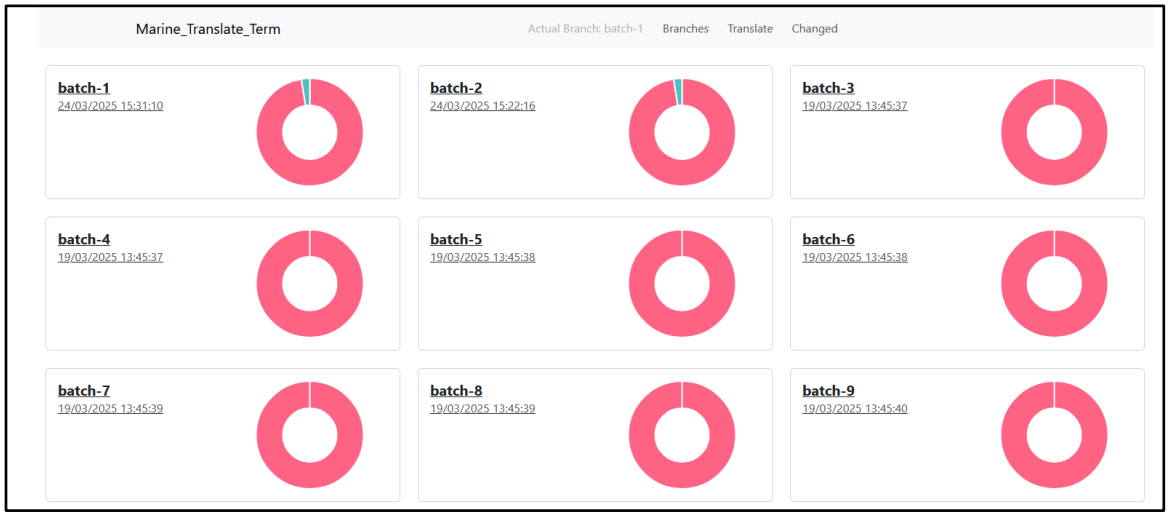

Figure 6. Overview of ongoing "batches" of remaining translation work and their level of completeness

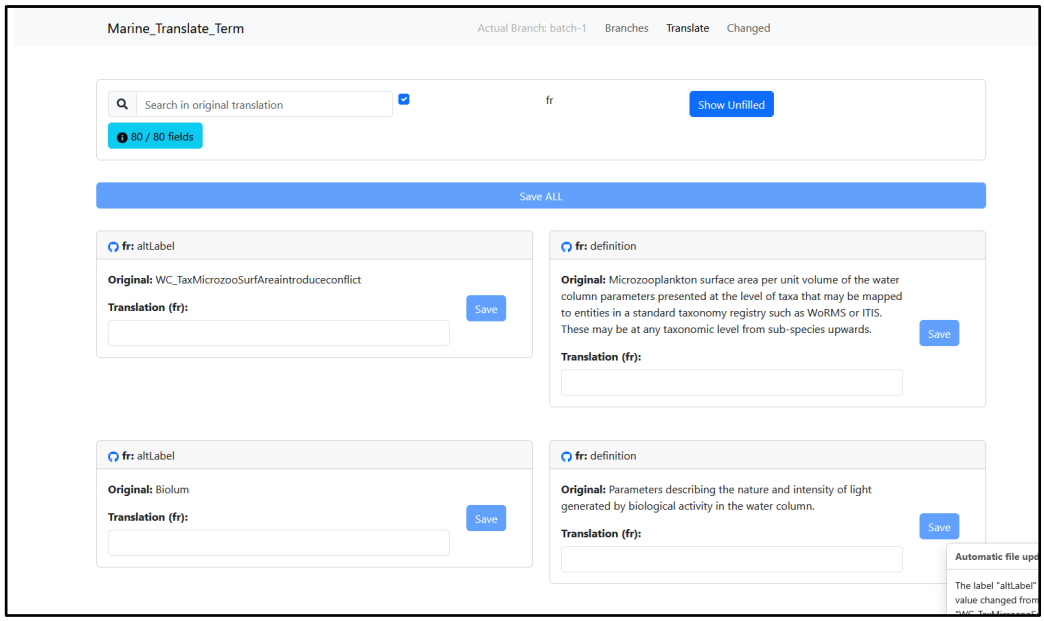

Figure 7. Core actual translation screen for individual terms and their label fields requiring translation.

3.4 Implementation details and lessons learned

By sticking to git as a core, this instantly leverages, through selecting one of the available git-based service platforms, the opportunity of free hosting to deploy this system. This at the mild cost of needing some custom integration code with that platform, an experience already present in the team.

This approach however proved to be only partially tenable: due to authentication and interaction limitations on the platform we were forced to develop and deploy a separate “back-end-proxy”. This is provided as a ready-to-run docker-image, but will require a specific hosting environment to run. On the plus side it creates an extra opportunity for caching, tuning, and intermediate processing.

3.5 References

At https://github.com/marine-term-translations/ one will find the various repositories for

- code of technical components (UI, back-end, custom github actions)
- the actual translation-management (content of translated terms)
- source for static generated portal website (https://marine-term-translations.github.io/)

3.6 Next steps

While development and technical testing of these components has been completed, showcasing the targeted use case remains to be done. This will involve:

- setting up the hosting for the node-backend component
- organise and support a selected set of translation teams , and appreciate their user-testing
- organise for user feedback, and be responsive to their needs
- finetune the code
- setup the vocab-search use case for multilingual term lookup

The aim, is to take up these tasks in the extended two years of the EMODnet Bio Phase V

4 Semantic Web Tech Recommendations

4.1 Introduction

In the field of information technology, a big push for the concept of semantics and data semantics was given by the extensive research on the so-called **Semantic Web or Web of Data**. The Semantic Web is an evolution of the traditional Web, the latter consisting of a network of documents (or web pages), mainly intended for consultation by people, linked together without specifying any meaning in the link.

In the Semantic Web, the emphasis is on **data** rather than on pages or documents and, through the use of the Web's established **open standards**, a **native linkage** is made between data by defining their **explicit, shared and formal meaning**. In this case, the meaning is not only searchable by people but also **processable by software** to realise innovative applications, also capable of discovering new knowledge through the navigation of links and reasoning about the meaning attributed to the data.

These principles can serve as crucial pillars for achieving unambiguous **semantic interoperability**. By focusing on data as the primary entity and establishing native linkages with clearly articulated semantics, the challenges of disparate data formats and interpretations across different systems can be significantly mitigated. The insistence on shared and formal meaning ensures that the context and relationships between data elements are consistently understood, not only by humans but also by machines in data exchange. Ultimately, this approach transforms data from isolated silos into a **cohesive and understandable network**, fostering unambiguous semantic interoperability across diverse applications and domains.

To put in practice the foundations of the Semantic Web, Sir Tim Berners Lee provided a simple guide consisting in **four important principles**:

1. Give a unique identifier to all things in the world (to data);
2. The unique identifier must be on the Web so that people and software can search for things through traditional Web protocols;
3. When searching for things through the unique identifier, it is necessary to provide information through specific standards such as RDF and SPARQL;
4. Links to other existing data in the Web of Data must be included to create value from data integration.

Semantic Web technologies play a fundamental role in the domain of marine research, where vast and diverse datasets must be integrated, interpreted, and shared across disciplines, institutions, and countries. In this context, EMODnet Biology contributes significantly to the development of a marine knowledge graph that leverages Semantic Web standards to enable seamless data discovery, interoperability, and reuse.

To fully realise the potential of such technologies, it is essential to understand the foundational principles and technical components that underpin the Semantic Web. In the following sections, we outline the key semantic pillars, and we also explore how these foundations align with the FAIR principles, we emphasise the importance of metadata, and examine the role of controlled vocabularies, including detailed strategies for managing their versioning, as well as other key aspects and recommended guidelines for ensuring semantic consistency and long-term interoperability.

4.2 The Semantics Pillars

4.2.1 Give an identity to things of the world: URI/IRI

It is always good to structure data by favouring the identification of entities and reducing the use of textual descriptions (*things, not strings*). Entities can then be unambiguously identified through the use of unique

and persistent identifiers that can be referred to in a shared manner, thus solving the problem of identity even in data interchange.

A **Uniform Resource Identifier (URI)** is defined as a sequence of characters that uniquely and persistently identifies a resource (on the web) over time. The term IRI - Internationalized Resource Identifier is used when referring to an identifier that allows for the use of characters suitable for languages other than English.

In the management of URIs/IRIs it is necessary to enable:

- **content negotiation mechanisms** in the request for a resource: these allow different representations of a resource to be made available to the same URI/IRI in the case of multiple representations/formats of the same resource;
- **URI dereferencing mechanisms**: when used with a browser, URIs must return a web representation (e.g., an informative web page) of the resource they identify.

4.2.2 Represent and link data with shared and well-known standards of the Web

The use of open and shared standards, which inherently offer the possibility of linking data, facilitates data interoperability and integration.

Within the context of the Semantic Web, the following standards are the reference points for achieving these objectives.

RDF – Resource Description Framework

In the Resource Description Framework (RDF)[1], Semantic web resources are described in terms of entities, which have a type, and relationships between entities, which have a meaning. The data representation is the form of `subject-predicate-object` triple, where the subject is a node that always has a URI/IRI, the predicate is an edge that always has a URI/IRI, and the object can be a simple string, a number, a date, a boolean, or a node with a URI/IRI, thus potentially becoming the subject of another triple. This latter mechanism allows for the chaining of triples, thereby creating the foundation for natively linking data to form an interconnected graph of knowledge.

It is worth underlining that RDF per se is not a data format but there exists so-called serialisations of RDF that can be used according to a variety of contexts and applications. Among the most prominent serialisations are RDF/Turtle, RDF/XML, and JSON-LD.

RDF/Turtle[2] (Terse RDF Triple Language) stands out as a human-readable and concise format. Its straightforward syntax, using abbreviations for common RDF constructs, makes it easier for both humans to write and read, and for machines to parse. Due to its readability and efficiency, RDF/Turtle is generally recommended for data exchange, especially in research and development environments, semantic web applications, and when human interpretability is crucial. It excels in scenarios where data needs to be easily understood and manipulated by developers and domain experts.

RDF/XML[3] was one of the earliest serialisations of RDF and, as such, has a long history and a wide range of tooling support. Being based on XML, it benefits from the extensive ecosystem of XML parsers and validators. However, its verbosity can make it less appealing for human consumption and can lead to larger file sizes compared to other serialisations. RDF/XML finds its niche in systems where XML is already a dominant technology or where compatibility with legacy systems and existing XML infrastructure is a primary concern. It can be suitable for enterprise-level data integration scenarios where robust schema validation and XML-based processing are well-established.

[1] https://www.w3.org/TR/rdf11-primer/
[2] https://www.w3.org/TR/turtle/
[3] https://www.w3.org/TR/rdf-syntax-grammar/

JSON-LD[4] (JavaScript Object Notation for Linked Data) leverages the familiar JSON syntax, making it particularly attractive for web developers. It allows embedding Linked Data within standard JSON structures, facilitating the integration of semantic data into web applications and APIs. Its compatibility with the JavaScript ecosystem and its ease of use in web development make it a strong contender for exposing structured data on the web, enhancing SEO, and building data-driven web applications. JSON-LD is particularly well-suited for client-side processing in web browsers and for APIs that need to serve Linked Data in a format readily consumable by JavaScript.

In essence, while RDF/XML offers robustness and legacy compatibility and JSON-LD seamless integration with web technologies, RDF/Turtle could be seen as a preferred choice for its balance of human readability, conciseness, and parsing efficiency, making it a versatile option for a wide range of semantic web applications and data exchange scenarios. In general, the optimal choice of serialisation ultimately depends on the specific requirements of the context, including factors like human readability, machine processing efficiency, existing infrastructure, target applications and practices of the application domain of reference.

4.2.3 Provide a meaning to the things of the world

In order to ensure semantic interoperability, it is necessary that the **meaning of data** is made explicit and preserved during data interchange. This involves defining, in a more or less articulate manner, data types and relationships between them.

This is accomplished by defining and using digital artifacts such as **controlled vocabularies** and/or **ontologies**. In the following we briefly describe them, highlighting the differences in terms of semantic expressivity they can offer.

Controlled vocabularies

The term *controlled vocabulary* refers to a standardised list of concepts denoted by reference or preferred terms and codes. The list is useful for organising, describing, predefining and indexing knowledge in a domain (e.g., the controlled vocabulary of the measurement units).

There exist different types of lists of concepts that fall under the general term *controlled vocabulary*:

- **Code List:** A list of concepts identified by a code and denoted by a term. Usually, the list is structured in one level where all the concepts with their related terms and codes are defined.
- **Taxonomy:** The list of concepts, with their preferred terms and codes, is organised in a hierarchical structure, where broader categories may include narrower subcategories (e.g., Clupeidae – Clupea - Clupea harengus)
- **Thesaurus:** The list of concepts is organised in a more structured manner than the previous two types. In a thesaurus, concepts can be related to each other, not only in a hierarchical form but also in the form of correlation or to indicate that one term of a concept is synonymous with another term. In general, a thesaurus should also mandatorily include the definitions associated with each concept.

Good and widespread practice is to use a specific ontology to define controlled vocabularies of all types. This ontology is a Web standard and is called **SKOS - Simple Knowledge Organisation System[5]**. Additional extensions of SKOS like the Extended Knowledge Organization System[6] (XKOS) and SKOS eXtension for Labels[7] (SKOS-XL) can be used in combination in order to allow for the specification of additional classes and properties.

[4] https://www.w3.org/TR/json-ld/
[5] https://www.w3.org/TR/skos-reference/
[6] https://rdf-vocabulary.ddialliance.org/xkos.html
[7] https://www.w3.org/TR/skos-reference/#xl

Ontologies

In computer science, the term ontology refers to a **formal, shared and explicit specification of a representation (conceptualisation) of a knowledge domain**, defined on the basis of specific requirements. The representation consists of the definition of entities (or **classes**), entity **attributes** and **relationships** between entities (or properties).

An ontology offers a much stronger semantics than that offered by simple controlled vocabularies or dictionaries: through the definition of its logical axioms, i.e. statements that, going beyond the simple enunciation of universally true principles, allow the expression of specific relations and constraints between entities (classes) within the ontology (e.g., axioms of equivalence, disjunction between class members, etc.), it is possible to enable automated reasoning, verify consistency in definitions, and infer new knowledge.

There are Web standards that allow ontologies to be created. The standards are:

- **RDF Schema[8] (RDFS):** is a lightweight vocabulary that extends the RDF model to semantically describe RDF data. It provides basic elements to create ontologies (e.g., it allows defining whether one class is a subclass of another, the domain and range of properties between classes, etc.).
- **Web Ontology Language[9] (OWL):** is a knowledge representation language that builds on RDF and RDFS, but offers more expressive capabilities to define data semantics (e.g., cardinality constraints to specify the number of instances a property can have, logical operators to define complex relationships, etc.).

In general, while more lightweight ontology languages exist, the adoption of OWL, especially when representing data in the marine science domain (and in general in Life Science domains), offers distinct advantages rooted in its **rich expressivity** and **formal semantics**. This expressivity allows for the creation of highly detailed and interconnected knowledge representations that go beyond simple hierarchical structures often found in less expressive models, as those offered by the SKOS ontology.

For instance, marine ecological and environmental science involves intricate interactions between various species, their habitats, environmental factors (temperature, salinity, oxygen levels), and human activities (fishing practices, aquaculture). OWL's constructs allow for the precise modeling of these relationships. Thus, one can define that a specific fish species *feeds on* another species, that certain fishing gear *impacts* specific benthic habitats, or that changes in water temperature *affect* the distribution of a particular fish stock. By reasoning over species characteristics (e.g., reproductive rate, habitat specificity, etc.) and environmental threats, potentially vulnerable populations can be identified and by reasoning over the effects of climate change (e.g., rising sea temperatures, ocean acidification, etc.) on different species and ecosystems can aid in developing adaptive management strategies.

Relationships between controlled vocabularies and ontologies

From the above, it is evident that although ontologies and controlled vocabularies allow meaning to be attached to data and can both be defined to represent knowledge of a certain domain of interest, the level of semantic depth offered by the two types of artifacts is different.

The figure below informally shows the degree of semantic depth offered by controlled vocabularies of different types compared to ontologies.

[8] https://www.w3.org/TR/rdf11-schema/
[9] https://www.w3.org/TR/owl2-primer/

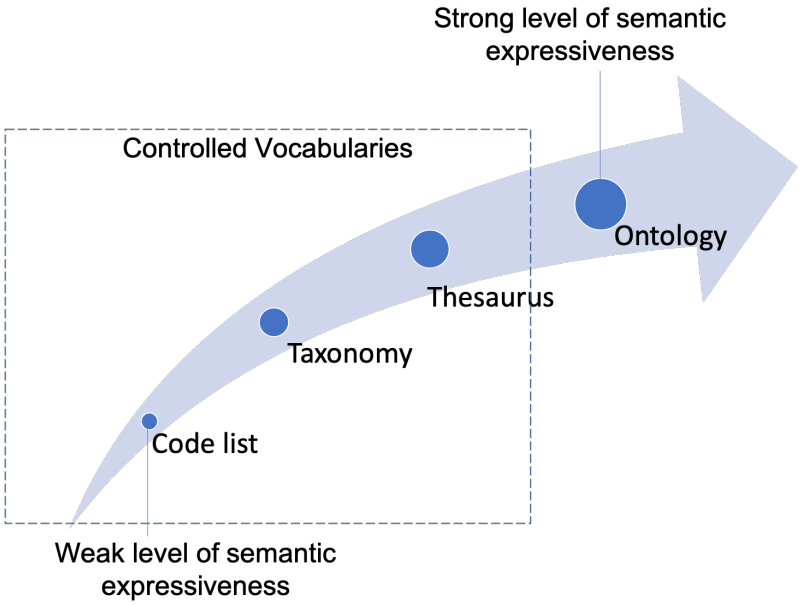

Figure 8. Semantic expressiveness of controlled vocabularies and ontologies

Therefore, to meet the specific requirements of domain applications and the intended semantic experience for end users, a flexible toolkit of potentially also combined instruments can be leveraged. For instance, one can construct complex OWL ontologies where concepts from controlled vocabularies, ranging in complexity, are instantiated as individuals of certain classes (e.g., the different types of ranks - *species*, *genus*, *family*, etc. - associated with a Taxon name). Typically, this relationship is not explicitly encoded within the ontology structure itself, but rather described in textual definitions within annotation properties such as rdfs:comment. However, in the EU, some Member States (e.g., Italy) adopt practices where specific annotation properties are defined to explicitly link concepts within the ontology to concepts defined separately in a SKOS-based controlled vocabulary. This can be afterwards processed by software in applications in order to validate data against the use of targeted controlled vocabularies for specific data types.

Application Profiles

An alternative or complementary practice with respect to what has been earlier discussed is to define so-called Application Profiles.

An **Application Profile** (AP) emerges as a practical solution, representing a contextually relevant and constrained subset of one or more existing ontologies. It defines the precise terms, the expected ways they should be used, and any additional restrictions necessary for a particular application, data exchange scenario, or community of users.

The **SHACL**[10] (Shapes Constraint Language) Web standard provides a robust mechanism for realising these Application Profiles. It allows the definition of *shapes*, which specify the expected structure and content of RDF data. These shapes act as constraints on the properties and types of resources. By defining a set of shapes that target specific classes or resources and impose conditions on their properties (such as required cardinality, data types, allowed value ranges also possibly coming from controlled vocabularies, or relationships to other resources), SHACL effectively formalises an AP. Validation against these shapes allows applications to then ensure that the data they consume or produce adheres to the specific requirements of their context.

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

While OWL can express constraints (e.g., through cardinality restrictions or property characteristics), these are typically interpreted as part of the logical definition of the ontology itself. Violations of these constraints might lead to logical inconsistencies within the knowledge model. The **Open World Assumption (OWA)**, a fundamental principle of RDF and OWL, states that the absence of a statement does not imply its falsity. If a piece of information is not explicitly present in the data, it is considered unknown, not necessarily false. This is crucial for dealing with the distributed and evolving nature of information on the web.

SHACL's approach to Application Profiles differs in its focus and interpretation of constraints. Instead of defining constraints as inherent parts of the domain's logical structure, as in OWL, SHACL treats them primarily as validation rules for specific data instances within the context of an application profile. When data violates a SHACL constraint, it is considered non-conformant to the profile for that specific use case. This does not necessarily imply a logical inconsistency in the broader knowledge model, as might be the case with an OWL violation.

Furthermore, while the Semantic Web operates under the OWA, SHACL introduces a degree of local "closedness" for the purpose of validation. When a SHACL shape specifies, for example, that a certain property must have at least one value, the absence of that value for a resource being validated against the shape is considered a violation of the application profile. This local "closedness" is specific to the validation context defined by the SHACL shape and does not contradict the broader OWA of the underlying RDF data.

In essence, OWL is about defining the shared understanding of a domain, including its inherent constraints. Application Profiles, realised through languages like SHACL, are about defining the specific expectations for how data conforming to those (or other) ontologies should be structured and populated for particular applications. SHACL provides a practical layer for ensuring data quality and interoperability within specific contexts, operating within the broader framework of the Semantic Web and its open world assumption but allowing for the definition and enforcement of application-specific data requirements.

In conclusion, the combined use of OWL ontologies, SKOS-based controlled vocabularies, and SHACL shapes for specific applications ensures that a clear and exhaustive semantic definition for data is provided to end-users, thus enabling interoperability, seamless data integration across diverse systems and contexts, and enhanced data quality, understanding, and reliable exchange.

4.2.4 Provide mechanisms to query the data

When dealing with semantics and interoperability, it is necessary to provide mechanisms that allow **data to be accessed and queried**, both in a person-machine and a machine-machine interaction.

To meet user needs and expectations, a key objective is to develop and manage multiple data querying Application Programming Interfaces (APIs). In the context of the Web, different types of APIs can be built ranging from simple HTTP APIs to more complex APIs capable of navigating data represented under the form of triples, as previously introduced. In this latter case, the data-oriented standard that makes it possible to navigate and query them, even if they come from different sources and are distributed over the network (federated queries) is the SPARQL Web standard.

SPARQL[11] is both a language for querying RDF graphs similar to SQL, and a protocol. The latter makes SPARQL a standard through which SPARQL queries and updates can be transmitted to a SPARQL processing service (exposed as a **SPARQL endpoint**) that returns the results via HTTP to the client application that requested them, thus enabling machine-to-machine interaction over the HTTP protocol.

In general, APIs are instrumental in creating digital ecosystems and facilitating coordinated digital interactions. However, this practice presents several challenges for data publishers. Firstly, the ongoing maintenance of multiple online APIs can be expensive, as the data publisher often bears the burden of the

[11] https://www.w3.org/TR/sparql11-query/

computational load generated by data consumers. Secondly, publishers face a significant and continuous effort to ensure their APIs remain current with evolving standards, technologies and ontologies. Critically, existing APIs can restrict data reuse and innovation because their inherent capabilities and limitations constrain how data consumers can operate. Users are largely limited to creating their own views or indexes on the data using their chosen technology, within the boundaries set by the provided APIs.

To overcome these possible limitations, other approaches are emerging for querying the data. One of these approaches, widely discussed in this document, is represented by the **Linked Data Event Stream**[12] (LDES) paradigm. LDES offers a foundational and adaptable API for accessing datasets. LDES allows publishing and consuming streams of data as a collection of immutable events using Linked Data principles. Fundamentally, an LDES, based on the RDF standard, represents a publishing approach where data providers enable numerous external parties to efficiently maintain synchronization with the most current version of a data source. In this respect, LDES presents a solution to the challenges of extensive API maintenance since with LDES, data consumers can establish automated processes to reproduce the historical evolution of a dataset and remain up-to-date with the latest changes.

The mechanism offered by LDES is to deliver a continuous sequence of unchanging data units that capture information updates originating from systems that are constantly generating data.

While Linked Data is a robust approach for representing and sharing information on the Web, its traditional focus has been on *static* datasets rather than dynamic events or modifications to that data. Linked Data Event Streams address this need by allowing applications to subscribe to a continuous flow of data and receive updates as they happen. LDES preserves the complete evolution of data objects over time and enables efficient publication and consumption of constantly changing datasets.

LDES also represents a significant architectural shift in how Linked Data is published and consumed. By design, LDES eliminates the traditional requirement for SPARQL endpoints, which had been the standard query interface for RDF data. Instead of maintaining complex query infrastructure that can become overloaded with concurrent requests, LDES publishers simply expose data as an append-only stream of immutable events that clients can process incrementally.

This approach, however, transfers more responsibility to the client side. LDES clients must implement capabilities to discover, retrieve, and process fragments of the event stream, track their synchronization state, and handle the reconstruction of current data states from sequences of events. Clients need to understand LDES-specific concepts like fragment navigation, member extraction, and version reconciliation. While this increases client-side complexity compared to simply writing SPARQL queries, it creates a more scalable and resilient ecosystem by reducing server load and single points of failure.

In conclusion, to effectively address the diverse needs and expectations of data consumers, it could become crucial to **offer a range of data access and querying mechanisms**, essentially providing different types of APIs tailored to specific use cases and enabling a wider spectrum of potential reuses and innovative applications. The rationale behind offering various API types stems from the fact that different data consumers have distinct needs, technical capabilities, and intended uses for the data. A one-size-fits-all API often falls short of effectively serving this diverse landscape, leading to inefficiencies, limitations, and ultimately hindering the full potential of the data.

4.3 FAIR principles

In the era of data-driven research and society, the FAIR principles have emerged as a cornerstone for effective data management and stewardship, particularly within the context of open science and when opening up data in various contexts. FAIR is an acronym that stands for:

[12] https://w3id.org/ldes/specification

Findable: Data and metadata should be easy to discover for both humans and computers. This involves assigning persistent and unique identifiers (like DOIs, URIs), providing rich and descriptive metadata, and ensuring that data and metadata are indexed in searchable resources.

Accessible: Once found, data should be accessible under clearly defined conditions. This doesn't necessarily mean open access, but it does require that the steps to access the data (even if restricted) are well-documented, and that metadata remains accessible even when the data itself is not. Access should ideally be through standardized communication protocols.

Interoperable: Data should be able to integrate and interact with other data or applications. This requires using standardized data formats, controlled vocabularies, and ontologies to ensure that data can be understood and processed by different systems and researchers across disciplines. Metadata, both general descriptive and about the content, should also use a formal, accessible, shared, and broadly applicable language for knowledge representation.

Reusable: The ultimate goal is to make data reusable for future research, innovation, and other purposes. This necessitates rich and well-documented metadata with clear information on data provenance, usage licenses, and adherence to relevant community standards. This allows others to understand the data's context, how it was created, and the conditions under which it can be used.

The FAIR principles are intrinsically linked to the goals of *open science*. When data is FAIR, it maximizes the return on investment in research, reduces redundancy, and promotes innovation by enabling a wider range of researchers to leverage existing datasets for new investigations.

However, the FAIR principles become paramount when opening up data in the general broad term. Simply making data publicly available without proper attention to findability, accessibility, interoperability, and reusability can limit its utility. FAIR principles provide the necessary guidance to structure and document open data in a way that makes it truly usable by others.

In conclusion, the FAIR principles are not just a set of guidelines; they are fundamental requirements for maximizing the potential of research data in the context of open science and when making data publicly available for the reuse by anyone for any purpose. By embracing FAIR, communities can move towards a more collaborative, transparent, and efficient data ecosystem, ultimately leading to greater scientific progress and societal benefits.

4.4 The role of metadata for data and semantic resources

Metadata for both datasets and semantic resources acts as a vital layer of descriptive information. It provides essential context, structure, and characteristics that go beyond the raw content itself, enabling effective discovery, understanding, interoperability, and management.

For datasets, metadata offers crucial details that allow users and applications to locate and assess their suitability. It describes the dataset's thematic content, relevant keywords, the geographical and temporal scope it covers, and its origin and creation process. This information is essential for search and filtering within data catalogs and repositories. Furthermore, metadata clarifies usage conditions, licensing terms, available formats, and citation guidelines, promoting responsible data access and reuse. Information about data quality, limitations, and versioning is also included, allowing users to make informed decisions about its applicability. Similarly, semantic resources (such as controlled vocabularies and ontologies) must be FAIR and should rely on the definition of metadata to facilitate their discovery and effective utilisation.

In the Semantic Web, this metadata is typically expressed using RDF, making it machine-readable and integrable within the interconnected web of data. This allows automated systems to process and understand the descriptions associated with datasets and semantic resources, enabling intelligent discovery, seamless data integration, and effective knowledge sharing.

There exists different standards that can be used for such a purpose. In general, DCAT, and its usage of other standards like Dublin Core, Prov-O, is recognised in Europe as the reference Web standard for metadata for datasets in catalogs and it has been also used as basis to define metadata for semantic assets (semantic resources). Despite other initiatives in the context of Open Science works, it could be worth adopting one common and well-known standard for metadata that appears to be used in a variety of contexts. This facilitates the discoverability in different scenarios of different types of resources.

Specific metadata can then be used to link together data and semantic resources used to describe that data. This is the case of specific properties like `dct:conformsTo` of Dublin Core and the IANA-registered "describedby" link relation in HTTP headers.

Within RDF metadata, the `dct:conformsTo` property serves as a direct link from a dataset (or a description of a resource) to the specific standard(s) and or Application Profile it is designed to follow. The value of this property is usually a URI that identifies the ontologies or an application profile the data is compliant with. This explicit declaration allows both machines and humans to understand the expected structure, vocabulary usage, and constraints associated with the data. Applications can leverage this information to validate the data against the specified profile, ensuring conformance and facilitating interoperability by establishing a shared understanding of the data requirements.

For non-RDF resources, the IANA-registered "describedby" link relation in HTTP headers provides an indirect way to associate a resource with its metadata. By using a Link header with rel="describedby", a server can point to a separate resource (which can be in RDF and not only) that contains metadata about the primary resource. This metadata document can then use the `dct:conformsTo` property to link the non-RDF resource to its intended Application Profile. This extends the reach of Application Profile awareness beyond RDF data itself, allowing other web resources to also declare their adherence to specific data structures and vocabularies through their associated metadata.

In essence, `dct:conformsTo` provides a direct declaration of conformance within metadata, while the "describedby" link relation enables the discovery of such metadata for other types of web resources. Both mechanisms are important for data quality, facilitating interoperability, and promoting a shared understanding of data expectations across the Semantic Web ecosystem, enabling applications to effectively process and exchange data based on agreed-upon profiles.

4.5 Controlled vocabularies and versioning policies

Controlled vocabularies serve as foundational semantic assets in knowledge organisation systems, providing a set of standardised terms or concepts that are used to represent and organise information within a specific domain or application. In the Semantic Web context, these vocabularies function as crucial building blocks that facilitate machine-readable data exchange and semantic interoperability across diverse systems and domains.

As already introduced, over the years SKOS has emerged as a W3C Recommendation and Web standard designed for the representation of diverse forms of structured controlled vocabularies in machine-readable formats compatible with Semantic Web technologies. SKOS provides a lightweight, intuitive RDF-based model for expressing the basic structure and content of concept schemes, allowing vocabulary concepts to be published, linked, and shared across applications.

More in detail, SKOS provides a standardised model for expressing concepts, their associated labels (in multiple languages, where needed), semantic relationships that define their connections (such as broader-narrower and associative links), and their organisation within coherent concept schemes. The primary objective of SKOS is thus to enable the straightforward publication and utilisation of controlled vocabularies as linked data on the Web, thereby fostering enhanced interoperability and the seamless sharing of knowledge across a multitude of applications.

In a nutshell, SKOS controlled vocabularies typically consist of:

- **concept schemes:** the overall controlled vocabulary or classification scheme;
- **concepts:** individual entities within the vocabulary;
- **relationships:** hierarchical, associative, and equivalence relationships between concepts;
- **labels:** preferred and alternative terms for concepts in different languages;
- **codes:** classification codes to uniquely identify concepts within vocabularies;
- **definitions:** precise meanings of concepts in natural language.

In essence, SKOS allows representing controlled vocabularies as flexible, interoperable semantic assets that can be: (i) published and shared on the web; (ii) linked with other data resources; (iii) incorporated into diverse applications; (iv) extended and combined across domains.

The role and importance of controlled vocabularies in the marine domain are well recognised and understood, particularly as the volume and diversity of data continue to grow. Key initiatives mentioned in this document, together with the EMODnet initiative itself, promote the use of common vocabularies as fundamental prerequisite for achieving consistency and interoperability. These vocabularies consist of standardised concepts/terms that span a wide range of disciplines relevant to marine science, helping to eliminate ambiguities in data integration, enhancing human understanding, and allowing data to be interpreted by machine, unlocking opportunities for automated data processing, integration, distribution, and reuse.

As the body of knowledge in a domain expands and deepens, the vocabulary that represents it must also adapt. Therefore, when SKOS vocabularies serve as representations of dynamic knowledge, they must also possess the capacity to evolve in response to changes.

4.5.1 The critical role of versioning

Controlled vocabularies evolve over time due to changes in domain knowledge, operational requirements, and user needs. This adaptation to changes can manifest in various forms, including the introduction of novel concepts to reflect emerging areas of interest, the removal or marking as obsolete of concepts that are no longer relevant, the refinement of existing concepts to more accurately capture their nuanced meanings, or the restructuring of the intricate relationships that exist between them.

Implementing robust versioning policies is thus essential for maintaining the integrity and usability of these semantic assets throughout their lifecycle as they evolve over time. Yet, the management of evolving SKOS vocabularies presents a unique set of challenges, including the crucial requirement of ensuring that systems and applications that rely on older versions continue to function as expected without disruption, and the ongoing need to communicate any and all changes in a clear and effective manner to the diverse community of users (and applications) who depend on the vocabulary for their work.

Furthermore, the interconnected nature of the Semantic Web introduces an additional layer of complexity, as modifications to one SKOS vocabulary can potentially have far-reaching consequences for a multitude of other linked data resources that depend upon it.

Versioning policies for controlled vocabularies are thus essential for:

- backward compatibility: ensuring systems using previous versions continue to function
- change tracking: documenting the evolution of vocabularies and concepts over time
- reproducibility: enabling exact reproduction of analyses that referenced specific vocabulary versions
- interoperability: facilitating data exchange between systems using different versions
- data integrity: maintaining the meaning of annotated and interlinked data across time

4.5.2 Versioning strategies

Versioning allows potentially complex changes to be tracked in a methodical manner, ensuring that the vocabulary remains an accurate reflection of the current understanding of the domain while simultaneously preserving a valuable historical record of how the entities of interest were previously conceptualised and interconnected.

To properly understand versioning within SKOS vocabularies, it is essential to clearly define the boundaries of the entities subject to version control. In the context of SKOS vocabularies, versioning can be applied at two distinct levels of granularity. These levels correspond to two core entities: the controlled vocabulary itself, represented as a SKOS `ConceptScheme`, and the individual entries within that vocabulary, represented as SKOS `Concepts`. Both the `ConceptScheme` and the `Concepts` it contains are considered versionable entities, and versioning provides a structured and systematic framework for effectively managing the evolution of both entire controlled vocabularies and individual concepts.

Depending on which of these two entities are versioned — only the `ConceptScheme`, or both the `ConceptScheme` and the `Concepts` — the versioning strategy and implementation approach will differ. Versioning can thus occur at different levels of granularity: *vocabulary-level* versioning and *concept-level* versioning[13]. Each scenario introduces different requirements and implications for how changes can be tracked. Both versioning approaches are complementary and serve different stakeholder needs. Vocabulary managers must determine which approach (or combination of approaches) best suits their governance requirements, user community needs, and technical infrastructure capabilities.

A robust versioning strategy for a SKOS controlled vocabulary involves the following core building blocks:

- **Vocabulary URI management strategy:** This defines how URIs for the SKOS `ConceptScheme` are structured and versioned. It determines whether the vocabulary URI changes with each version or remains stable, and how version identifiers (e.g., dates or version numbers) are represented, either within the URI itself or through associated metadata.
- **Concept URI management strategy:** This addresses how URIs for individual `Concepts` are managed over time. A key principle is persistence—once assigned, a concept URI should remain unchanged unless the concept's identity fundamentally changes. This helps maintain stable references across datasets and applications.
- **Metadata framework:** This involves the use of metadata to describe the temporal characteristics and validity/state of both `ConceptSchemes` and `Concepts`, such as creation and modification dates, validity periods, or deprecation status. This framework supports transparency and helps users understand the lifecycle of each element.
- **Version relationship expression:** This defines how relationships between different versions are explicitly represented, often through specific properties for well-known ontologies, or custom predicates. These links allow consumers of the vocabulary to navigate between versions and understand how concepts or schemes have evolved.

4.5.3 Vocabulary-level versioning

Vocabulary-level versioning addresses changes to the entire vocabulary as a cohesive unit. This approach can capture any type of change, from minor edits to the title of the controlled vocabulary up to major structural reorganisations, significant scope expansions, or methodological shifts affecting the vocabulary as a whole.

[13] versioning strategies and approaches discussed here can be also extended to other resources, in particular to SKOS `Collections`

Simplified vocabulary-level versioning

Single-URI evolution model

In its most basic implementation, a controlled vocabulary follows a streamlined evolution model where the vocabulary and its constituent concepts maintain stable, persistent URIs throughout their lifecycle. Under this approach, the vocabulary evolves and undergoes modifications over time, but its fundamental identity—expressed through its URI—remains constant. Similarly, concept URIs remain unchanged regardless of semantic refinements or adjustments to their terms, definitions, relationships, or other elements.

This approach treats the vocabulary as a living document that develops incrementally rather than as a series of discrete artifacts. The vocabulary URI always resolves to the current, authoritative version, providing users and applications with a single, consistent access point to the most up-to-date semantic content.

Metadata-based version documentation

While URIs remain stable, version information is carefully documented through metadata properties associated with the vocabulary. These metadata elements typically include elements like version number, last modification date, version notes summarising changes, status indicators (e.g., to identify deprecated concepts).

This metadata provides essential contextual information about the vocabulary's evolution without requiring structural changes to the URI scheme. People and applications can inspect this metadata to understand the vocabulary's maturity and recent modifications, but they cannot directly access previous versions through URI mechanisms.

This simplified versioning approach offers several practical advantages, from reduced integration complexity (as applications can rely on stable URIs), to implementation simplicity and lower maintenance overhead. However, this approach also has important limitations. In particular, there is no direct mechanism for accessing specific historical versions via URIs or content negotiation, and references to the vocabulary cannot be easily qualified with temporal context, as links always point to the current state. Moreover, users may find it difficult to precisely identify how the vocabulary has changed between specific points in time.

To address the need for historical version access while maintaining URI simplicity, external tools and repositories are typically employed to store and provide access to previous versions of the vocabulary. Common approaches include using a GitHub repository for maintaining the vocabulary in a version control system and publishing dated or versioned snapshots of the vocabulary in downloadable formats.

Advanced vocabulary-level versioning

For more complex requirements more sophisticated versioning strategies with version-specific URIs and content negotiation may be necessary.

Vocabulary URI management strategy

Distinct URIs for individual versions: When a new revision of a controlled vocabulary is created, it constitutes an entirely separate semantic artifact with its own persistent URI, allowing each version to exist as a discrete, independently referenceable entity.

Reference URI for latest version: A canonical "version-neutral" reference URI should be established that always resolves to the current, authoritative version of the vocabulary. This enables applications to automatically access the most recent version without requiring code changes when updates occur, while still allowing explicit reference to specific historical versions when needed.

Version-independent URI base: A base URI namespace for the vocabulary should be established that serves as the foundation for both the version-specific URIs and the current reference URI, creating a coherent identification framework.

Concept URI management strategy

Stable concept URIs: The preferred approach maintains persistent URIs for concepts across vocabulary versions, allowing a single concept to participate in multiple concept schemes representing different versions of the vocabulary. This persistence preserves semantic identity through evolutionary changes and allows referencing a SKOS concept always as the most current version.

Version-dependent concept URIs: An alternative approach ties concept URIs to the versioning scheme of the vocabulary itself, generating new URIs for all concepts when a version changes. This means that the URI of a concept would change even if the concept was not subject to any change. This requires explicit mapping relationships between concept versions but creates cleaner separation between versions. An explicitly versioned URI allows referencing a SKOS concept as it existed at a precise moment in time. (or, specifically, as it existed in a specific version of the enclosing controlled vocabulary).

Mapping infrastructure: When concept URIs change across versions, explicit mapping relationships (using predicates like skos:exactMatch, skos:closeMatch, etc.) must be established to maintain concept lineage and support migration between versions.

Metadata framework

Version documentation: Each vocabulary version requires explicit metadata including version number, release date, validity, change log and justification relative to previous versions.

Change documentation: Comprehensive documentation of changes between versions should be maintained, using the properties that SKOS natively provides (such as `skos:changeNote`) or potentially using dedicated change management vocabularies.

Deprecation patterns: Clear mechanisms for marking obsolete elements (entire controlled vocabularies and/or concepts) and indicating preferred replacements, allowing systems to gracefully handle superseded semantic content.

Version relationship expression

Version relationship predicates: Explicit semantic relationships between versions of a controlled vocabulary should be established using predicates like `dcterms:replaces`, `dcterms:isReplacedBy`, `prov:wasRevisionOf`, or similar terms from standard versioning vocabularies.

4.5.4 Concept-level versioning

Concept-level versioning tracks changes to individual concepts within the vocabulary. This more granular approach accounts for scenarios where specific concepts undergo refinement, expansion, or deprecation, and there is the need to make available and reference specific versions of individual concepts.

While in vocabulary-level versioning a controlled vocabulary can be versioned without explicitly versioning its concepts, in general concept-level versioning also induces vocabulary-level versioning, under the assumption that a change to one or more concepts constitutes a change to the enclosing controlled vocabulary and therefore requires a version increment of the controlled vocabulary itself. Yet, a version increment of the controlled vocabulary does not propagate to concepts and does not automatically result in a version increment of all of its concepts, as in the “Version-dependent concept URIs” strategy outlined before.

Vocabulary URI management strategy

The same principles and approaches outlined above for advanced vocabulary-level versioning apply.

Concept URI management strategy

Distinct URIs for individual versions: When a new revision of a concept is created, it constitutes an entirely separate semantic artifact with its own persistent URI, allowing each version to exist as a discrete, independently referenceable entity.

Reference URI for latest version: A canonical "version-neutral" reference URI should be established that always resolves to the current version of a concept. This enables applications to automatically access the most recent version without requiring code changes when updates occur, while still allowing explicit reference to specific historical versions when needed.

Mapping infrastructure: When concept URIs change across versions, explicit mapping relationships (using predicates like skos:exactMatch, skos:closeMatch, etc.) must be established to maintain concept lineage and support migration between versions.

Metadata framework

The same principles and approaches outlined above for advanced vocabulary-level versioning apply and extend to concepts concerning version documentation, change documentation and deprecation patterns.

Version relationship expression

The need to specify and manage semantic relationships between versions of a controlled vocabulary, as defined for advanced vocabulary-level versioning, also extends to concepts.

4.5.5 Version numbering schemes

As with software, version numbering schemes provide a structured way to distinguish between different iterations of a vocabulary or concept, and ensure consistency in usage, referencing, and maintenance. Several versioning models can be adopted, depending on the complexity of the vocabulary and the frequency and nature of its updates.

In its most basic form, a version identifier relies on a simple incrementing integer (e.g., 1 → 2 → 3 etc., or v1 → v2 → v3 etc.). This straightforward approach indicates successive editions or releases. Each new version number corresponds to a distinct version of a vocabulary or concept, regardless of the scope or impact of the changes.

However, as vocabularies grow in complexity or require more nuanced change tracking, more sophisticated versioning schemes, such as *semantic versioning*[14] (i.e., following patterns such as *major.minor.patch*, e.g., 2.3.1), may be adopted to convey different levels of change, such as major revisions, minor additions, or patches. A comprehensive reference for this version numbering scheme is provided by the DDI Alliance in its policy[15], with clarifications on and examples of changes to vocabularies and concepts that correspond to "major", "minor" and "sub-minor" revisions.

This approach aligns with broader semantic versioning principles, while addressing the specific needs of multilingual controlled vocabularies. In particular, the recommended approach for versioning controlled vocabularies is a three-level structure (X.Y.Z) where:

- **Major version** increments (1.0.0 → 2.0.0) indicate substantial changes that may break backward compatibility, such as:
    - Significant structural reorganisations

[14] https://semver.org/

[15] https://rdf-vocabulary.ddialliance.org/

- Deprecation of concepts
- Changes to fundamental semantic relationships
- Changes to code values
- Definition or term changes that alter meaning

- **Minor version** increments (1.1.0 → 1.2.0) reflect additions or enhancements (changes to form but not meaning) that preserve backward compatibility, such as:
  - Addition of new concepts within existing schemes
  - Expansion of concept relationships
  - Addition or rephrasing of (alternative) terms or definitions without meaning change
- **Sub-minor/Patch increments** (1.1.1 → 1.1.2) represent minor corrections or clarifications that don't affect the vocabulary's semantic interpretation, such as:
  - Fixing typographical errors in labels or documentation
  - Enhancing metadata or documentation without altering concepts
  - Translation updates
  - Addition of new language support

4.5.6 Ontology support for metadata and version relationship expression

As discussed in previous sections, to support effective management and use of SKOS controlled vocabularies, it is important to incorporate metadata that captures the lifecycle and versioning of both concepts and concept schemes. This involves using specific properties — and in some cases, classes — from SKOS and complementary ontologies to describe elements such as creation dates, modification history and change notes, validity and deprecation, and version relationships. These components enhance transparency and enable users to trace the evolution of vocabularies over time. In the following, we briefly outline relevant properties and structures that support these aspects; however, the list is not intended to be exhaustive.

Change management and versioning metadata

The following properties can be used to provide human- and machine-readable version information, document changes, and temporal references.

Ontology	Property	Description
OWL	`owl:versionInfo`	textual, human-readable representation of the version information for a resource
SKOS	`skos:changeNote`	description of fine-grained changes to a concept, for the purposes of administration and maintenance
SKOS	`skos:historyNote`	description of significant changes to the meaning or the form of a concept
DCAT / PAV	`dcat:version` / `pav:version`	version indicator (name or identifier) of a resource
ADMS	`adms:versionNotes`	description of changes between this version and the previous version of the resource
Dublin Core	`dcterms:issued`	release date of a version/resource
Dublin Core	`dcterms:modified`	modification date of a version/resource

Relationships between versions

In addition to using properties to provide metadata for summarising changes to a controlled vocabulary and for documenting modifications at the concept level, it is essential to establish explicit links between versioned instances of controlled vocabularies and concepts. This ensures that each version can be properly related to its predecessors and successors.

It is worth mentioning that DCAT version 3 has introduced properties to deal with versioning, building upon existing vocabularies, in particular the versioning component of the Provenance, Authoring and Versioning (PAV) ontology[16].

The following properties can be used for expressing these relationships. These properties allow building a version chain that can be navigated backward from a given version to the first one, as well as specifying a version hierarchy, by linking an abstract resource to its versions.

Ontology	Property	Description

Dublin Core / DCAT / PAV	`dcterms:hasVersion`
	`dcat:hasVersion`
	`pav:hasVersion`
		property intended for relating a non-versioned or abstract resource (the current/latest version of a controlled vocabulary or concept identified by a "version-neutral") to several versioned resources (the different versions of a controlled vocabulary or concept having a version-specific URI)

DCAT / PAV	`dcat:hasCurrentVersion` `pav:hasCurrentVersion`	property intended for relating a non-versioned or abstract resource (the current/latest version of a controlled vocabulary or concept identified by a "version-neutral") to a single snapshot corresponding to the current version of the resource (having a version-specific URI)
DCAT / PAV	`dcat:previousVersion` `pav:previousVersion`	property intended for relating a version of a resource to its previous version in a lineage; this allows specifying a version chain, consisting of snapshots/versions of a resource
XKOS	`xkos:follows`	property intended for relating a version of a resource to its previous version, for defining the succession in time of SKOS `ConceptSchemes`

Validity and deprecation of resources

For managing the lifecycle of controlled vocabularies or individual concepts, the preferred approach is of deprecating outdated resources and replacing them with new ones rather than directly deleting or altering the URIs of existing resources. This is in line with the requirement of having stable and persistent identifiers for resources. When managing the lifecycle of controlled vocabularies and concepts within a vocabulary, it is essential to have a clear strategy for handling deprecated concepts and their potential replacements. When a vocabulary or concept becomes outdated, is no longer recommended for use, or has been superseded by a new version, it should be explicitly marked as deprecated. Furthermore, if a deprecated entity or version has been directly replaced by one or more new entities, this relationship should be clearly and semantically indicated.

The following properties provide basic mechanisms for marking obsolete elements and indicating preferred replacements.

[16] https://pav-ontology.github.io/pav/

Ontology	Property	Description
OWL	`owl:deprecated`	boolean property to indicate whether a resource is deprecated
Dublin Core	`dcterms:replaces` `dcterms:isReplacedBy`	property intended for indicating that a version of a resource replaces (or is replaced by) another version
Dublin Core	`dcterms:validity`	property that can be used to specify the date (or temporal range) of validity of a resource/version
XKOS	`xkos:supersedes`	property intended for indicating that a version of a controlled vocabulary supersedes/replaces another version

When there is the need to express potentially complex correspondences and mappings between concepts in different versions of controlled vocabularies, more advanced constructs are needed, such as correspondences and concept associations[17] as defined in XKOS. The same applies when representing advanced resource lifecycles that go beyond the simple true/false nature of the `owl:deprecated` property. In this regard, DCAT version 3 introduces basic support for modeling resource lifecycle status[18] through the use of the `adms:status` property, which can be paired with an existing or custom status controlled vocabulary.

4.5.7 Summary and guidelines

The development and maintenance of SKOS controlled vocabularies that evolve over time require thoughtful versioning policies. By relying on persistent identifiers, semantic versioning, clear change categorisation, and comprehensive documentation, it is possible to ensure that controlled vocabularies remain valuable, usable, and interoperable resources over time.

- The first step towards effective SKOS vocabulary versioning is establishing a clear **versioning policy and workflow**, including decisions about
    - the specific **versioning strategy** (or a combination thereof) that will be adopted (vocabulary- and concept-level versioning)
    - the **version numbering scheme** (e.g., semantic versioning) and the specific **triggers** that will initiate the creation of a new version
- Successful controlled vocabulary management requires managing **URI stability and persistence**: long-term stability and reliability of the resources' identifiers is required, ensuring that they continue to resolve correctly over time
    - distinct URIs can be used to **identify specific versions** of the vocabulary scheme or concepts
    - when **version-neutral URIs** are used, a fundamental principle is to ensure that these URIs remain constant across all subsequent versions
    - a well-designed **URI scheme** (version-dependent and/or version-neutral) should be adopted for (versions of) controlled vocabularies and their concepts
    - ensuring the **continued accessibility of previous versions** of the vocabulary or concepts, through the establishment of a publicly available archive and/or via advanced URI dereferencing schemes, is considered a crucial best practice
- **Documenting changes** and providing comprehensive release/change notes is an indispensable aspect of effective SKOS vocabulary versioning

[17] https://rdf-vocabulary.ddialliance.org/xkos.html#correspondences
[18] https://www.w3.org/TR/vocab-dcat-3/#life-cycle

  - **release/change notes** should be prepared (and provided with the impacted resources using appropriate properties) to highlight the most significant updates, additions of new features or concepts, removals of deprecated items, and any other changes that users need to be particularly aware of;
  - provide **human-readable summaries of changes** for each version to enhance transparency and usability
- When managing the lifecycle of concepts within a SKOS vocabulary, it is essential to have a clear strategy for **handling deprecated concepts** and indicate their potential replacements
  - when a concept becomes outdated, is no longer recommended for use, or has been superseded by a new concept, it should be explicitly **marked as deprecated**, instead of deleting them, to support backward compatibility and traceability
  - if a deprecated concept has been directly **replaced** by one or more new concepts, this relationship should be clearly and semantically indicated using specific properties
- The **links between versions** should be clearly defined using available properties
  - this allows establishing a clear version history by linking each version to its predecessor and/or successor (**version chain**), and to its version-neutral representation (**version hierarchy**)
- Documenting and communicating **versioning information** effectively to users (and machines) is fundamental
  - this involves clearly stating the current version of the vocabulary and related information (e.g., temporal reference, nature of changes, etc.) in its associated **metadata**
- It is recommended to leverage existing **standard and consolidated Semantic Web ontologies** (such as SKOS itself, OWL, DCAT, Dublin Core Terms and XKOS) to semantically describe the **relationships** and **metadata** associated with different versions of a SKOS vocabulary or concepts

4.5.8 Versioning policy for vocabularies in BODC's NERC Vocabulary Server

When it comes to the management and versioning of controlled vocabularies, several well-established initiatives play a prominent role, including within the context of the EMODnet Biology initiative. Notably, the versioning practices adopted for SKOS vocabularies in the SeaDataNet initiative—hosted by BODC via the NERC Vocabulary Server—demonstrate advanced versioning policies. These practices are broadly aligned with the principles and strategies outlined earlier, offering a concrete example of how vocabulary governance and versioning can support semantic consistency and interoperability.

We summarise hereafter the key aspects related to versioning practices.

- Terminological resources represented by `skos:Concepts` are managed as `skos:ConceptSchemes` (referred to as thesauri) and `skos:Collections` (referred to as vocabularies)
  - `skos:Collections` are the most prominent (and numerous artifacts) form of concepts aggregation that is managed and made available
- For `skos:ConceptSchemes` a **basic vocabulary-level versioning** approach is used, where only the current version in made available and version information is provided via the `owl:versionInfo` property
- An advanced **vocabulary-level and concept-level versioning** approach is used for `skos:Collections` and `skos:Concepts`. More in detail:
  - both collections and concepts are versioned, with a version numbering scheme based on progressive integer

- for a collection, a version-neutral URI is defined to represent current/latest version, and version information is provided with the `t` property
  - yet, it seems there there are no resolvable version-specific URIs to access specific versions of a vocabulary/collection
- for concepts, each version of a concept has a distinct persistent and dereferenceable URI, and a version-neutral URI is defined to represent current/latest version; `owl:versionInfo` and `pav:version` properties are used to provide version information
- properties from Dublin Core Terms and the Provenance, Authoring and Versioning (PAV) ontology are used to relate different versions of a concept to its current version-neutral representation (`pav:hasVersion`/`dct:isVersionOf` and `pav:hasCurrentVersion`), while versions are chained with the `pav:previousVersion` property
- concepts that are no longer fit for purpose are deprecated (`owl:deprecated true`) but never deleted, so that their URIs remain active and dereferenceable
  - deprecated concepts and concepts that replace them are related using `dcterms:replacedBy`/`dcterms:replaces` properties
  - deprecation involves the latest/current version of a concept, but does not propagate to previous versions
- no specific properties (such as `skos:changeNote`) are used to document changes from one version of a resource to the next one

4.6 Next steps

The increasing complexity and heterogeneity of marine data present ongoing challenges for integration and interoperability across scientific, institutional, and national boundaries. In this section we have explored how Semantic Web technologies — through key enablers such as shared identifiers, formal ontologies, and interoperable data standards and APIs — provide a robust and scalable framework to address these challenges. By aligning marine data management practices with Semantic Web principles, initiatives like EMODnet Biology are already contributing to improve semantic interoperability. By aligning with these principles, EMODnet Biology can continue to evolve into a more integrated and semantically robust data infrastructure.

This work serves as a starting point for even more advanced semantic integration across the EMODnet Biology initiative. While it has not undertaken a full assessment of current adoption levels, it lays the groundwork for such an evaluation. This activity will also help to identify key gaps, inconsistencies, or areas where further guidance and technical support are needed.

The assessment of current practices within EMODnet Biology can highlight areas of promising potential progress, particularly in the use of controlled vocabularies, the adoption of FAIR principles, and the provision of Linked Data as Linked Data Event Streams. However, it can also help surfacing critical gaps, such as uneven use of ontologies, or limited alignment between institutional metadata practices.

The current landscape within EMODnet Biology already demonstrates a strong inclination toward embracing diversity in data provision mechanisms, reflecting an encouraging shift toward more open and flexible data access strategies. This includes the availability of SPARQL endpoints, RESTful APIs, and increasingly, the use of LDES for dynamic and scalable data dissemination. Such diversity supports a wide range of use cases and user needs, from complex semantic queries to lightweight, real-time data consumption. However, to fully harness the potential of these approaches, there is a growing need to address more advanced integration topics — particularly the alignment of LDES with established metadata frameworks such as DCAT-AP.

Addressing these issues will require both technical adjustments and coordinated governance strategies.

5 Outreach

Many parts of this work are openly available in various github resources mentioned throughout the text.

The design and results of it have been (or will be) presented at:

- SEMIC-EU pre-conference on 2024-06-26
- upcoming: 2025-05-22 a SEMIC hosted online event targeted to showcase LDES use in practice
- Proposed for presentation at a future EMODnet Technical Working Group meeting

6 ANNEX - List of abbreviations

Acronym	Full Term
API	Application Programming Interface
BODC	British Oceanographic Data Centre
CDI	Common Data Index
EDMERP	European Directory of Marine Environmental Projects
EDMO	European Directory of Marine Organisations
EMODnet	European Marine Observation and Data Network
EU	European Union
GHCR	GitHub Container Registry
GUI	Graphical User Interface
ITB	Interoperability Testbed
LDES	Linked Data Event Streams
LDIO	Linked Data Interactions Orchestrator
LOD	Linked Open Data
MR	Marine Regions
NERC	Natural Environment Research Council
NVS	NERC Vocabulary Server
RDF	Resource Description Framework
R2RML	RDB to RDF Mapping Language
SDN	SeaDataNet
SHACL	Shapes Constraint Language
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
TTL	Turtle (RDF serialization format)
UI	User Interface
URI	Uniform Resource Identifier

VLIZ	Vlaams Instituut voor de Zee (Flanders Marine Institute)
WoRMS	World Register of Marine Species
YAML / YML	YAML Ain't Markup Language