Modular Ontology Design Approach for an ESPR Compliant Digital Product Passport Core Ontology

Riina Maigre*, Tarmo Robal and Hele-Mai Haav

Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia

Abstract

In the nearest future, digital product passports (DPP) will be enforced on products put on the EU market. Similar initiatives are taking place in other parts of the world as well. Ontologies are seen as a solution to capture the knowledge of the complex DPP ecosystem domain to provide unambiguous interpretation and semantic interoperability. However, building such ontologies remains a challenge due to the domain complexity and involvement of various stakeholders. In this paper, we aim to solve this problem by providing an ontology design approach for developing the DPP core ontology, targeting cross-sectorial information needs. For this, we explore and combine different ontology engineering methods, to allow engineering of a modular core ontology for an ESPR compliant digital product passport.

Keywords

Digital product passport, DPP core ontology, ontology design, modular ontology, ESPR, DPP

1. Introduction

Digital Product Passports (DPPs) are emerging as a key tool in the transition toward a more transparent and sustainable product economy. A DPP is a set of data specific to a product as a digital record consisting of a variety of information for transparency, traceability and sustainability for consumers, public officials and a multitude of other actors along the entire value chain. This information includes details about product materials, origin, manufacturing process, environmental impact, and compliance. It is expected that DPPs will help customers to make informed choices, thereby promoting circular economy (CE) through repairs, refurbishment, and recycling. This can only happen if reliable information about a product's origin, composition, environmental impact, and lifecycle is available and accessible across various actors along the entire product value chain. Therefore, DPPs are expected to play a crucial role in connecting digital information with physical products.

The efforts to establish DPP ecosystems started before the European Union (EU) adopted the Ecodesign for Sustainable Products Regulation (ESPR) [1] in June 2024. Yet, the adoption of the ESPR was a major and crucial milestone for DPPs (a central instrument in the ESPR), as it provided the legal framework, policy objectives, and implementation mechanism that make DPPs possible and mandatory in the EU. Our work here, focuses on the ESPR as the regulatory foundation for the development of a core ontology for DPPs, applicable to many sectors, for which details will be laid down by the upcoming Delegated Acts (DA) for sectors (e.g., textiles, furniture, batteries).

Our motivation is to create an ESPR-based, regulatory-compliant DPP core ontology (CO) to enable interoperability of DPP data [2] across sectors and the DPP ecosystem [3]. According to the requirements specification [4], the DPP CO is a shared formal specification of basic concepts and properties/relationships in the cross-sectorial DPP domain, implemented in the Web Ontology Language (OWL) or the Resource Description Framework (RDF). One of the goals of our ongoing research and development (tightly connected to the CIRPASS-2 European project) is to define ontology design approach for a DPP CO, and apply it on the developing of a EU DPP CO represented in OWL. For this purpose, we combine different ontology engineering methods with a special focus on modular ontology engineering

Baltic DB&IS 2026 Conference Forum and Doctoral Consortium, 28 June - 1 July 2026, Tartu, Estonia

*Corresponding author.

riina.maigre@taltech.ee (R. Maigre); tarmo.robal@taltech.ee (T. Robal); hele-mai.haav@taltech.ee (H. Haav)

0000-0001-9797-9128 (R. Maigre); 0000-0002-7396-8843 (T. Robal); 0000-0002-7378-3865 (H. Haav)

approaches. This works well with the three-level ontology architecture for the DPP domain outlined in [4].

Modularisation techniques allow large complex ontologies to be considered as consisting of smaller self-contained and interrelated modules (also formed as ontologies). There are two approaches for performing modularisation: ontology decomposition (extraction of modules from a large ontology) and ontology composition [5, 6]. The latter refers to developing modules independently (possibly by independent teams) so that they can be integrated coherently and uniformly into one integrated ontology. According to the requirements set in the specification document [4], we base our modular ontology design on the ontology composition approach.

This paper proposes the main steps for modular ontology design approach and discusses the pros and cons of applying this approach to the DPP CO development. In this paper, we do not focus on evaluation, maintenance and publication phases. We answer the following research question: *How can modular ontology design principles be applied in building a DPP core ontology?*

The rest of the article is organised as follows. In Section 2, we provide an overview of related work. Section 3 presents the design approach for the DPP CO development. Section 4 discusses the implementation of the DPP CO, and Section 5 sums up the paper.

2. Related work

Our work is driven by the concept of DPP, as specified by the ESPR, for which with the emergence of the topic, several works already existed prior to the adoption of the framework regulation specifying aspects of DPPs and their development. In [7], a systematic literature review on the implementation and adoption of DPPs is provided, concluding that one of the barriers for DPP adoption is a lack of standardized data formats, and reluctance of sharing data. A six-layered technical architecture for DPPs based on 50 peer-reviewed articles is proposed in [8], and a list of ten most prominent items of information required for DPPs based on literature in [9]. Several works [10, 11, 12] address ontologies as a solution to semantic interoperability and knowledge sharing issues for DPPs.

Ontology building is a complex engineering task. During the last decades, many well-known ontology development approaches have been proposed. A good overview and analysis of ontology engineering methodologies is presented in [13]. One of the most important early systematic methods is the METHONTOLOGY methodology [14] that combines software development activities with knowledge engineering methodologies. Another important ontology development methodology is the NeOn [15] that is devoted to building ontology networks. This approach pays attention to the activities related to collaborative ontology work performed by distributed teams. In relation with linked data applications, ontology development methodologies pay more attention to such ontology development aspects like publishing and accessibility [13].

Currently, most ontologies that are developed are intended to be used for applications, and this requires decreasing the level of complexity of ontology design and development approaches, as well as making the process faster. There are several ways of doing this. For example, Haav [16] presented a methodology for the construction of light-weight domain ontologies by bringing ontology modelling closer to domain experts being the actual domain knowledge holders. Another way to decrease the level of complexity of the ontology building is by using ontology modularisation techniques [17, 18]. Modular ontology engineering methods divide ontology into conceptually separated parts (modules) that are related to each other via relationships.

Several modular ontology development methodologies are presented in [17, 18, 19, 20]. The Modular Ontology Modeling (MOMo) methodology [20] provides steps for ontology requirements elicitation as well as for ontology design and building/coding. The main value of MOMo is its systematic use of ontology design patterns and tool support. The application of MOMo for ontology building is well explained and illustrated in [11]. However, MOMo does not include publication activities and maintenance in the method workflow. Its workflow ends with the implementation of the OWL ontology file. This is in contrast to [13], where these activities are well supported. In current work, we are

inspired by the MOMo but our ontology design approach has some differences compared to the MOMo as explained in the next sections.

This paper is a continuation of our previous work, which addresses ontology requirements for DPP development under the regulatory constraints of the ESPR [3, 4], and issues of semantic interoperability for DPPs [2]. The clear focus on the ESPR distinguishes our contribution to this topic from existing work.

3. A modular ontology design approach for the DPP CO development

3.1. Background and method overview

The DPP domain determined by the ESPR incorporates a wide range of knowledge of different ecodesign related topics calling for the collaborative development of ontologies in order to gain interoperability of software systems providing relevant and consistent DPP data. Ontologies developed independently by collaborative teams of domain experts and ontologists need to be merged into an integrated coordinated ontology – the DPP CO. Modularity is a key requirement to perform this task. It concerns ontology design, implementation, and maintenance. In addition, the modular ontology development approach supports reusability of ontologies as the modules of the DPP CO can be reused in other ontologies within similar domains. Likewise, external ontologies can be reused in the DPP CO in the form of a standalone module or nested into some existing modules. From the point of view of maintenance, smaller components (modules) are also easier to maintain.

Although modular ontologies are easier to (re)use and maintain, the implementation and integration of ontology modules formulated in a logic-based language such as OWL remain complex tasks. Modular ontology design approaches need to take into account that an OWL ontology represents a logical theory, making the process difficult. Current ontology engineering methods do not pay enough attention to the definition of the notion of modularity in a way that takes into account the semantics of the ontologies and their implications. Early research in [5, 18, 21] addresses semantics of ontology modularisation that is useful to reconsider. For example, in [5] an ontology module is defined as a reusable self-contained component of a larger ontology, but modules are not isolated entities or disjoint from each other and can be extended by introducing new concepts and relationships.

In the CIRPASS-2 European project, tightly connected to this work, our ultimate goal is to use the principles of modular ontology design to develop the DPP CO in practice. Prior to this, we had established the DPP CO requirements specification by defining the requirements for regulatory compliant core ontology development [3, 4], and suggested a three-level ontology architecture for the DPP domain as follows:

1. **The upper level** represents the core ontology as an extensible high level of abstraction of the DPP domain as described by the ESPR. The DPP CO can be extended or specialised for sector-specific ontologies (the middle level) while maintaining coherence and consistency with the DPP CO. Should other cross-sectoral ontologies be proposed for the EU DPP, semantic mappings can be established to these other ontologies.
2. **The middle level** represents sector-specific ontologies (e.g., textiles) holding classes and relationships as an interface between both the DPP core ontology and industry level ontologies.
3. **The lower level** addresses the industry level ontologies with concepts describing a certain aspect of the DPP data used in a specific industry.

Although there are several modular ontology development methods available, we could not find the one that met our practical needs. Therefore, the modular ontology design approach for the development of the DPP CO presented in this paper combines several ontology engineering methods widely used for the development of ontology networks and ontologies used for linked data applications[22]. Our starting point is the usage of principles of modular ontology development methodologies [11, 20, 15]. In our work, we are especially inspired by ideas of the Modular Ontology Modeling (MOMo) methodology by Shimizu et al. [20].

Our approach facilitates incremental development, providing the following iterative major steps for ontology design:

1. **Identification of modules**. This step is based on requirements elicitation of the DPP domain as presented in the ESPR. The identification of modules is part of the ontology requirements specification [4]. We do not use Competency Questions (CQ) to identify patterns for key notions that correspond to modules as in MOMo [20], as we do not use patterns. However, we have developed a set of CQs for covering the whole DPP CO and the key notions are identified according to the text of the ESPR. In comparison, MOMo finds key notions using literature reviews and interviews. The identification of modules is further detailed in Section 3.2.
2. **Design and development of individual modules**. Our approach is based on iterative module development process depicted in Fig. 1. Modules design and development process gets input from module specification that should capture requirements from ESPR and other regulatory acts, if applicable. We detail this step in Section 3.3.
3. **Ontology composition**. This step consists of defining relationships between modules in order to compose the integrated ontology (DPP CO) that meets the given set of requirements. This step is similar to MOMo workflow step devoted to integration of modules to coherent ontology. We discuss this step in more detail in Section 3.4.
4. **Ontology implementation**. This step addresses ontology implementation in the form of an OWL file for publication and reuse. We propose an early implementation approach within our iterative development process. Contrary, in MOMo, OWL files are created from the ontology schema presented in the form of a diagram utilising the CoModIDE tool developed by the authors of MOMo methodology [20]. In our case, visualisation of the ontology code is performed using the WebVOWL ontology viewer/editor [23]. MOMo implements OWL code only from drawn schemes of ontology modules and their relationships. On the other hand, we use the Protégé ontology editor [24] for coding ontology in OWL. In contrast, MOMo uses their own tool incorporated to Protégé and the corresponding patterns libraries for creating ontology code (OWL). Our ontology implementation is further discussed in Section 4.

3.2. Identification of modules of the DPP CO

Ontology modularisation is an approach allowing an ontology to be perceived simultaneously as a whole and as a set of parts (modules). As stated above, our method uses ontology composition approach [5, 6], and for identifying possible modules, it adapts ideas from MOMo methodology [20]. The MoMo [20] defines the concept of a module as: *"An (ontology) module is a part of an ontology which captures a key notion, and its key relations to other notions"*. It is also emphasised in [20] that modules are guided not only by the semantics of the domain but also by the development context and use case.

The identification of modules is part of requirements elicitation. In [3, 4], we presented a set of requirements for the development of the DPP CO related to the ESPR [1]. These requirements also identified the set of key notions for establishing the following main DPP CO modules:

- **Product and DPP** – this module is the central and most important module of the DPP CO. It describes the concepts, properties, and relationships of products and their digital passports with respect to all other modules, as required by the ESPR.
- **Actor** – the role of this module is to capture the knowledge about Economic Operators (EO), actors, roles, and related concepts described in the ESPR.
- **Substances of Concern** – this module describes the substances of concern (e.g., chemicals contained in products or used during production) related to products, as laid out in the ESPR.
- **Lifecycle Events** – this module captures the concepts related to product or DPP life-cycle events (e.g., placed to market, or DPP created).
- **Environmental Impact and Life-Cycle Assessment** – this module captures the concepts related to the environmental impact of the product, such as emissions to air, water, or soil, or environmental footprint, including carbon footprint.

- **Product Conformity and Compliance** – the module captures concepts related to product conformance and conformity (e.g., compliance documentation, technical documentation, and conformity certificates of a product).

The aim of such modularisation was to separate different aspects of a product for the DPP into separate manageable modules, each having a specific target focus as described in the ESPR, to be developed and managed by dedicated teams.

According to modular ontology design, an ontology is represented as a set of interconnected ontologies (modules) that fulfil the requirements of the ontology as a whole. Thus, the DPP CO is an ontology composed of the above-listed modules. As a whole, it meets the requirements derived from the ESPR [4]. However, the requirements for individual modules must also be specified. In our case, this is done by the domain experts of a particular module.

There is a general understanding in the previous work [5, 25] on some main characteristics of an ontology module as follows:

1. A module of an ontology must be minimal. This assumes that the modules can have very little in common and, therefore, as little interaction as possible should be required between the modules.
2. Self-containment states that every module should be able to exist and function without any other module. Self-containment does not consider relationships with other modules, which may be different if the module is reused in another ontology.
3. Integrity ensures the correctness of other related modules and the integrated ontology. Any knowledge that can be inferred from a module should be possible to be inferred from the integrated ontology.

3.3. Design and development of individual modules

For the design and implementation of individual modules, we propose an iterative ontology development process depicted in Fig. 1. This process is based on our previously created and used practical methodology for the development of e-Government domain ontologies [16]. To meet the goals of the DPP CO development, the method is adapted to fulfil the requirements of modular design, meaning more attention is paid to the step of integration with other ontologies or modules, as this is crucial for modular design. Accordingly, module development starts from partial satisfaction of requirements. During each of the iterations, the ontology modules will be improved until the requirements for both the individual modules as well as the integrated ontology are met. The main iterative steps of the ontology development process are given in Fig. 1 as the following sequence: specification, conceptualisation (definition of classes, class hierarchy, properties, etc.), integration with other modules, implementation, and evaluation.

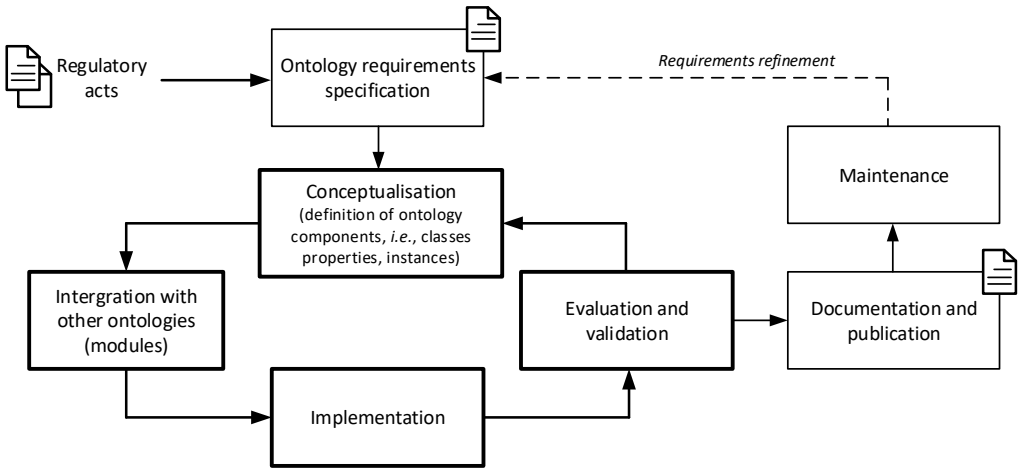

Figure 1: Iterative ontology module development process for the DPP core ontology.

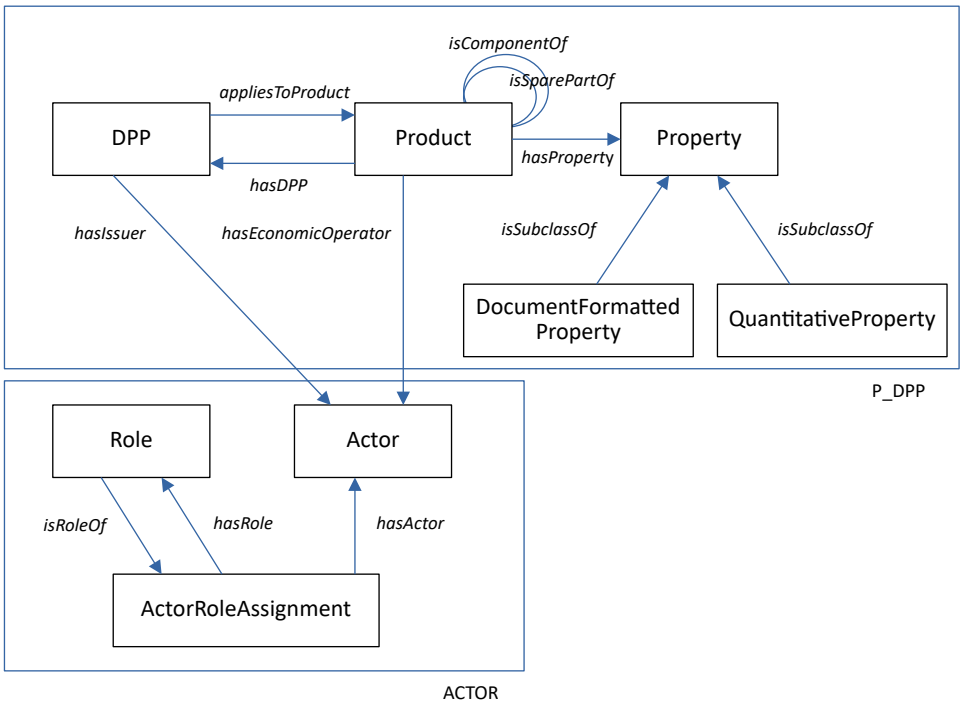

Figure 2: An example of the ACTOR and P_DPP modules and possible relationships between them in the DPP CO.

The iterative ontology module engineering method is recommended to be used by teams working separately on modules development. However, this is not obligatory and teams can use other methodologies for module engineering.

As examples, let us consider two developed modules of the DPP CO: the *Product and DPP* (P_DPP) module and the *Actor* module, depicted in Fig. 2. The basic structure of the P_DPP module is formed by the following classes: *DPP*, *Product*, and *Property* related via appropriate relationships (see Fig. 2). The *Property* class has two direct subclasses: *DocumentFormattedProperty* and *QuantitativeProperty*. The *QuantitativeProperty* class has many subclasses to define different measurable parameters of a product required by the ESPR (e.g., *QualityIndicator* with subclasses *Reliability* and *Durability*). In addition, it includes four subclasses related to the concepts that are not directly defined in the ESPR but are rather mentioned there. For example, we have included the following classes and some of their subclasses that express the environmental impact related concepts as follows: *CircularEconomyIndicator*, *EnvironmentalPollution*, *ResourceConsumption*, and *WasteGenerationAmount*. These class descriptions need to be extended/specialised according to the corresponding Delegated Acts (DA) and/or industry standards. This can be done at the *middle-level* of the three-level ontology architecture model by using subclass relationships between the DPP CO classes and the corresponding classes from external sectorial level modules. The *Product* class has two transitive object properties: *isSparePartOf* and *isComponentOf*.

The Actor module includes three basic classes: Actor, Role and *ActorRoleAssignment*. These two modules are related via two relationships: the *DPP* class is related via *hasIssuer* relationship to the *Actor* class and the *Product* class is related via *hasEconomicOperator* relationship to the *Actor* class.

3.4. Ontology composition

We consider the composition of an integrated ontology at two levels: conceptualisation and implementation. In addition to conceptualisation activity of modules (see Fig. 1), we need to identify additional ontology elements in order to be able to build a conceptually coherent and formally consistent composed ontology. This goal makes ontology composition a complex task. Basically, relationships between modules should be carefully identified and reuse of external ontologies or their parts should be decided.

As modules are developed by different teams of experts, this process should be well curated. It is important to ensure that reusable external ontologies are permanently available and stable.

Choices of implementation of relationships between modules, for example, in the Web Ontology Language (OWL), have influence (give feedback) on the design of the modules. The ontology composition process is dependent on the representation of relationships between different modules. After several iterations of the DPP CO development and targeting the properties of ontology modules stated above, we specified a set of requirements for modularisation and implemented these as follows:

1. Modules should be self-contained. In our case, this also means that all ranges of object properties of the module should be defined.
2. Complex class axioms (OWL restrictions) related to the classes of another module, if these exist, are defined within the module that requires these axioms and not as mapping relations. This is according to the self-containment requirement of a module.
3. Relationships between modules for creating a whole integrated ontology (DPP CO) are defined between corresponding classes using object properties in OWL. For example, the relationship named *hasIssuer* represents the relationship between the DPP and Actor classes as the object property in OWL (see Fig. 2). Equivalent or subclass relationships are allowed only for representing relationships of external modules (ontologies). For example, in the case of extending/specialising the DPP CO within sectorial ontologies.
4. Reasoning over the whole ontology (DPP CO) should be made possible. The consistency check of separate modules and the integrated ontology (DPP CO) should be performed. This can be done using DL reasoners such as Pellet [26].
5. The propagation of changes in modules to the entire ontology and other related modules is important and should be guaranteed. For example, this can be performed using ontology versioning at the implementation stage.

The implementation of the composition of the DPP CO is further detailed in Section 4.

4. Implementation of the DPP CO

As mentioned in Section 3, the presented methodology for the design and development of the DPP CO (see Fig. 1) relies on an incremental process that makes it possible to start ontology implementation from a small number of key modules and a limited number of key concepts testing the results of each phase. This also enables to detect and resolve design and implementation issues early, and allows modules to have different life-cycles while being developed by different orchestrated teams. The resulting ontology modules and the integrated DPP CO are represented in OWL-2[1], but can easily be converted into other widely used formats, such as JSON-LD[2], RDF/XML[3], or Turtle[4], if necessary.

We started our early implementation of the DPP CO with three interrelated modules as follows: P_DPP, ACTOR, and SOC, containing 94 classes, 69 properties, and 6 individuals. After two iterations of the DPP CO development, the DPP CO for these modules contains 64 classes, 72 properties, and 27 individuals. At each iteration, the consistency of individual modules and the integrated DPP CO was successfully checked with Pellet reasoner. Currently, we are in the third development iteration. The results of the development iterations are available in the Digital Product Passport Vocabulary Hub[5] using the Semantic Treehouse (STH) platform. The results of these iterations use different schemes for the implementation of namespaces (and IRIs) to experiment and understand the pros and cons of each approach. In further subsections, we consider these approaches.

[1]https://www.w3.org/TR/owl2-overview/
[2]https://json-ld.org/
[3]https://www.w3.org/TR/rdf12-xml/
[4]https://www.w3.org/TR/rdf12-turtle/
[5]https://dpp.vocabulary-hub.eu

4.1. Choices of namespaces for modules

In general, ontology modules are self-contained, independent, and reusable knowledge components. From the implementation point of view, this means that ontology composition can be implemented by integrating modules in OWL-2, where the *owl:imports* statement enables to import one or more OWL files to bring them together into one file (ontology). An OWL-2 ontology import provides access to the entities, individuals, and related axioms of other ontologies, accordingly ensuring ontology modularisation. Imports are managed using HTTP-based uniform resource identifiers (URI or IRI) of an ontology. In our case, modules are developed by different teams working on the project. Furthermore, some modules that are available on the Web may be reused in the integrated ontology.

The ontology IRI identifies the ontology itself (i.e., ontology file), while the namespace is used to identify entities, assuring that ontological terms are unique within the ontology. The namespace does not need to be identical to or be derived from the ontology IRI, although this is often the case.

If ontology development teams are not coordinated with each other or external ontologies are reused, then it is better to have separated namespaces for modules in order to avoid term collisions. In the case of coordinated governance of ontology development teams, all developed modules may have the same namespace.

There are some design and development choices that we would like to discuss and pay attention to, especially with respect to implementing relationships between the modules. We discuss namespaces and IRI options such as using a common namespace for all modules or using unique namespace per module, and the consequences for the applicability of the corresponding ontology code (document).

4.1.1. Using unique namespace per module

For the first iteration of our DPP CO, we used the unique namespace per module option, as we expected multiple diverse teams of ontologists working separately on implementation of the DPP CO modules, as well as we had foreseen reuse of already existing modules available on the Web (e.g., SI ontology for units[6]). In this case, the integrated DPP CO as well as modules use unique namespaces and the general form of module term IRIs look like `https://example.org/moduleA#ClassX`. Table 1 provides examples of IRIs from two modules: P_DPP (classes Product and EU_DPP) and ACTOR (class Actor) with an object property *hasIssuer* connecting these modules, presented on Fig. 2. The object property *hasIssuer* has the domain EU_DPP and the range is left open (which we resolve in Section 4.1.2).

Table 1
Examples of IRIs for modules using unique namespace per module

Module	Class	IRI
P_DPP	Product	`https://dpp.taltech.ee/EUDPP#Product`
	EU_DPP	`https://dpp.taltech.ee/EUDPP#EU_DPP`
ACTOR	Actor	`https://dpp.taltech.ee/EUDPP/ACTOR#Actor`
	Object property	
EUDPP	hasIssuer	`https://dpp.taltech.ee/EUDPP#hasIssuer`

The use of unique namespace per ontology module has both advantages and disadvantages. The advantages include the following:

1. Clear provenance of modules. It is easy to understand which module defined a term from its namespace. This helps in tracing.
2. Avoidance of name collision. Independent namespaces reduce accidental name clashes between modules developed by different teams.

[6] https://si-digital-framework.org/SI

3. Modular reuse. Users of the module can import only the namespaces they need and reuse modules in different combinations more easily.
4. Independent versioning and release. Each module can be versioned, published, and evolved separately without unexpected changes to other modules.

The disadvantages of using a unique namespace are related to the declaration and management of multiple prefixes, which may increase the complexity of integration and querying of the ontology, as well as the human readability of its documentation. However, the major disadvantage is the conceptual fragmentation, as the terms conceptually related inherently belong to different modules in different namespaces rather than to one coherent domain (e.g., DPP) model. For the latter reasons, we opted for single namespace for all modules in the further development iterations, despite that initially the idea of a unique namespace per module was attractive.

4.1.2. Using single (common) namespace for all modules

For the second, and especially for the third iteration, we have opted for the application of common namespace option for all modules developed in the framework for the DPP CO. In this approach, a single namespace is used for all DPP CO modules, including the integrated CO. The DPP CO single namespace for the CIRPASS-2 European project is https://w3id.org/eudpp. The namespace of the modules is the namespace of the CO. However, each module defines its IRI as a combination of the namespace + module name, e.g., for the Actor module the IRI is https://w3id.org/eudpp/ACTOR.

Utilising a common namespace for ontology terms yields some principal benefits as follows:

1. All entities share a single predictable prefix, easing authoring and human readability.
2. Many ontology editors and code generators assume a single base IRI, making generation and serialisation simpler.
3. Single namespace facilitates RDF querying by allowing users to define just one prefix for integrated CO terms.
4. Single namespace makes it easier to refactor terms from one ontology to another as the need arises.

Using a common namespace has the consequences as follows:

1. Modules lose a clear independent identity; it becomes harder to tell which module defined a term.
2. It embeds a risk of name collisions as accidental duplicate local names across modules lead to unintended overrides or confusion.
3. When many teams contribute, a single namespace requires strict coordination and naming policies, creating governance problems.
4. Using the same namespace creates more difficult versioning and provenance tracking.
5. Users may be confused when module-scoped namespaces are expected, and one cannot rely on the namespace to discover the module boundaries or intent.
6. Reuse (soft reuse) of independently developed ontology modules is not easy (needs care).

When modules are small, tightly coupled, conceptually related, governed by a consortium of collaborators, or a single coherent vocabulary is to be created, the use of a common namespace is beneficial. The use of a single namespace may be challenging when different uncoordinated communities are developing modules independently, considering versioning and reuse. In this case, separate namespaces scale better for multi-contributor projects.

4.2. Implementation of relationships between modules

We store the relationships of modules in a separate ontology file, named Connector, to keep the architecture simple and manageable over several teams. This separation ensures that the connector always provides a good overview of module relationships, and changes, for example to include a new

Listing 1: Example (excerpt) of the Connector implemention in OWL.

```
1 <!-- https://w3id.org/eudpp#hasIssuer -->
2
3 <owl:ObjectProperty rdf:about="https://w3id.org/eudpp#hasIssuer">
4     <rdfs:domain rdf:resource="https://w3id.org/eudpp#DPP"/>
5     <rdfs:range rdf:resource="https://w3id.org/eudpp#Actor"/>
6     <rdfs:comment xml:lang="en">Linking DPP with its issuer (Actor from ACTOR
          module).
7         DPP is modelled in the EU DPP CO PDPP module.
8         Actor is modelled in the EU DPP CO ACTOR module.</rdfs:comment>
9     <rdfs:label xml:lang="en">Has issuer</rdfs:label>
10 </owl:ObjectProperty>
11
12 <!-- Description of class origin is given by rdfs:isDefinedBy and module IRI -->
13 <owl:Class rdf:about="https://w3id.org/eudpp#Actor">
14     <rdfs:isDefinedBy rdf:resource="https://w3id.org/eudpp/ACTOR"/>
15 </owl:Class>
16
17 <owl:Class rdf:about="https://w3id.org/eudpp#DPP">
18     <rdfs:isDefinedBy rdf:resource="https://w3id.org/eudpp/P_DPP"/>
19 </owl:Class>
```

module, need to be done only in this single file. It also enforces the designers (ontologist) to re-think the connections and their necessity between modules, keeping modules self-sufficient. On the one hand, by keeping the relationships between modules in a separate file, we gain good control over module integration and have better control over the DPP core ontology. On the other hand, modules that are connected, could be developed separately, and then integrated later to the DPP CO. For example, the conceptual relationship *hasIssuer* between the modules P_DPP and ACTOR presented in Fig. 2 is implemented in OWL as shown in Listing 1.

We implement this separation of relationships into a separate ontology file (Connector) by defining the object property in the connector file and specifying the domain and range using the IRI of the corresponding classes in the modules being connected. This is represented by lines 3–10 in Listing 1. To identify the origin of classes per module, we use the *rdfs:isDefinedBy* property to indicate the modules by their IRI, as indicated by lines 12–19 in Listing 1.

5. Conclusion

In this paper, we presented a modular design approach for the development of the ESPR-compliant DPP core ontology and some practical recommendations based on our experience in ontology engineering and the application of this method to the DPP CO development under the CIRPASS-2 European project. Our special focus was on practical aspects of applying modular ontology design principles in building the DPP core ontology. We combined elements of different ontology engineering methods, and explored the advantages and disadvantages of unique and common namespace for module implementation over several teams. To solve the module integration problem, we provided a simple and clear mechanism for ontology composition in single-namespaced ontologies composed of modules developed by separate teams.

Although our approach provides a set of practical guidelines for ontology modularisation and implementation of modular ontologies, there is still a need for more detailed guidelines for ontology and module reuse activities.

In the future, work will continue on the reuse of already existing ontologies or modules, and on connecting these to the DPP CO, for example, events described in the EPCIS ontology. Also, the functionality of the ontology regarding to its usability and effectiveness in practical applications should

be evaluated using competency questions. SPARQL queries developed to answer the CQs presented in [4] will demonstrate how the core ontology is able to address the competency questions.

Acknowledgments

This work was supported in part by the CIRPASS-2 European project which received funding under the Digital Europe Programme Grant No. 101158775. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] European Union, Ecodesign for sustainable products regulation (ESPR), 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1781/oj, official Journal (OJ) L 2024/1781, CELEX: 32024R1781.

[2] E. Hbeich, C. Bernier, S. V. Hooland, T. Robal, R. Maigre, Achieving semantic interoperability for digital product passports, Procedia Computer Science 277 (2026) 2981–2992. doi:https://doi.org/10.1016/j.procs.2026.02.334, 7th International Conference on Industry of the Future and Smart Manufacturing (former International Conference on Industry 4.0 and Smart Manufacturing).

[3] T. Robal, R. Maigre, H.-M. Haav, Ontology requirements for digital product passports based on the ecodesign for sustainable products regulation, Procedia Computer Science 277 (2026) 356–365. doi:https://doi.org/10.1016/j.procs.2026.02.077, 7th International Conference on Industry of the Future and Smart Manufacturing (former International Conference on Industry 4.0 and Smart Manufacturing).

[4] R. Maigre, H.-M. Haav, T. Robal, M.-A. Wolf, F. Danash, Ontology Requirements Specification for an EU DPP Core Ontology Proposal, 2025. doi:10.5281/zenodo.15270342.

[5] J. Pathak, T. M. Johnson, C. G. Chute, Survey of modular ontology techniques and their applications in the biomedical domain, Integrated Computer-Aided Engineering 16 (2009) 225–242. doi:10.3233/ICA-2009-0315.

[6] Y. Wang, J. Bao, P. Haase, G. Qi, Evaluating formalisms for modular ontologies in distributed information systems, in: M. Marchiori, J. Z. Pan, C. d. S. Marie (Eds.), Web Reasoning and Rule Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 178–193.

[7] F. Abedi, U. A. Saari, L. Hakola, Implementation and adoption of digital product passports: A systematic literature review, in: 2024 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC), 2024, pp. 1–9. doi:10.1109/ICE/ITMC61926.2024.10794320.

[8] H. Wicaksono, A. Mengistu, A. Bashyal, T. Fekete, Digital product passport (DPP) technological advancement and adoption framework: A systematic literature review, Procedia Computer Science 253 (2025) 2980–2989. doi:https://doi.org/10.1016/j.procs.2025.02.022, 6th International Conference on Industry 4.0 and Smart Manufacturing.

[9] M. Pourjafarian, C. Plociennik, S. Bergweiler, N. Moarefvand, J. Brozeit, M. Rezapour, M. Ruskowski, An ODP-based ontology for the digital product passport, Procedia CIRP 135 (2025) 930–935. doi:https://doi.org/10.1016/j.procir.2024.12.125, 32nd CIRP Conference on Life Cycle Engineering (LCE2025).

[10] E. Blomqvist, H. Li, R. Keskisärkä, M. Lindecrantz, M. A. N. Pour, Y. Li, P. Lambrix, Cross-domain modelling – a network of core ontologies for the circular economy, in: Proceedings of the 14th

Workshop on Ontology Design and Patterns (WOP 2023) - Colocated with the 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, 2023. CC BY 4.0.

[11] R. Kebede, A. Moscati, H. Tan, P. Johansson, A modular ontology modeling approach to developing digital product passports to promote circular economy in the built environment, Sustainable Production and Consumption 48 (2024) 248–268. doi:10.1016/j.spc.2024.05.007.

[12] M. Jansen, E. Blomqvist, R. Keskisärkkä, H. Li, M. Lindecrantz, K. Wannerberg, A. Pomp, T. Meisen, H. Berg, Designing an ontology network for digital product passports, in: The Semantic Web: ESWC 2024 Satellite Events, Springer Nature Switzerland, Cham, 2025, pp. 220–237. doi:https://doi.org/10.1007/978-3-031-78955-7_19.

[13] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, R. García-Castro, LOT: An industrial oriented ontology engineering framework, Engineering Applications of Artificial Intelligence 111 (2022) 104755. doi:https://doi.org/10.1016/j.engappai.2022.104755.

[14] M. Fernández-López, A. Gómez-Pérez, N. Juristo, Methontology: From ontological art towards ontological engineering, in: Proceedings of the AAAI Spring Symposium on Ontological Engineering, AAAI Press, Stanford, California, USA, 1997, pp. 33–40.

[15] M. C. Suárez-Figueroa, A. Gómez-Pérez, M. Fernández-López, The NeOn Methodology for Ontology Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 9–34. doi:10.1007/978-3-642-24794-1_2.

[16] H.-M. Haav, A practical methodology for development of a network of e-government domain ontologies, in: T. Skersys, R. Butleris, L. Nemuraite, R. Suomi (Eds.), Building the e-World Ecosystem, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1–13.

[17] M. d'Aquin, A. Schlicht, H. Stuckenschmidt, M. Sabou, Criteria and Evaluation for Ontology Modularization Techniques, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 67–89. doi:10.1007/978-3-642-01907-4_4.

[18] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler, Extracting Modules from Ontologies: A Logic-Based Approach, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 159–186. doi:10.1007/978-3-642-01907-4_8.

[19] F. Ensan, W. Du, A knowledge encapsulation approach to ontology modularization, Knowledge and Information Systems 26 (2011) 249–283. doi:10.1007/s10115-009-0279-y.

[20] C. Shimizu, K. Hammar, P. Hitzler, Modular ontology modeling, Semantic Web 14 (2023) 459–489. doi:10.3233/SW-222886.

[21] B. C. Grau, I. Horrocks, Y. Kazakov, U. Sattler, Just the right amount: extracting modules from ontologies, in: Proceedings of the 16th International Conference on World Wide Web, WWW '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 717–726. doi:10.1145/1242572.1242669.

[22] M. Poveda-Villalón, P. Espinoza-Arias, D. Garijo, O. Corcho, Coming to terms with FAIR ontologies, in: C. M. Keet, M. Dumontier (Eds.), Knowledge Engineering and Knowledge Management, Springer International Publishing, Cham, 2020, pp. 255–270. doi:10.1007/978-3-030-61244-3_18.

[23] S. Lohmann, V. Link, E. Marbach, S. Negru, WebVOWL: Web-based visualization of ontologies, in: P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, C. Ghidini (Eds.), Knowledge Engineering and Knowledge Management, Springer International Publishing, Cham, 2015, pp. 154–158.

[24] M. A. Musen, T. P. Team, The Protégé project: A look back and a look forward, AI Matters 1 (2015) 4–12. doi:10.1145/2757001.2757003.

[25] E. Blomqvist, K. Sandkuhl, Patterns in ontology engineering: Classification of ontology patterns, in: Proceedings of the Seventh International Conference on Enterprise Information Systems - Volume 3: ICEIS, INSTICC, SciTePress, 2005, pp. 413–416. doi:10.5220/0002518804130416.

[26] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Journal of Web Semantics 5 (2007) 51–53. doi:https://doi.org/10.1016/j.websem.2007.03.004, software Engineering and the Semantic Web.