Towards Conformance in ODRL 3.0

Víctor Rodríguez-Doncel[1,*], Nicoleta Roman[1]

[1]*Universidad Politécnica de Madrid, Madrid*

Abstract

The Open Digital Rights Language (ODRL) version 2.2 is a W3C standard for representing permissions, prohibitions, and obligations in digital policies. ODRL policies can be flexibly applied across various scenarios, such as governing access control or establishing rules whose violation can be monitored. However, the interoperability of ODRL processors is not guaranteed: conformance is currently an attribute of ODRL policies, but not of ODRL processors. This paper highlights the importance of adopting systematic procedures to gather use cases from the community, extract clear requirements, and establish robust conformance mechanisms –ideally through test suites– to draft an ODRL 3.0 specification that addresses these interoperability challenges. The article finally describes how conformance works in ODRL 2.2 and then offers recommendations for drafting conformance mechanisms in ODRL 3.0.

Keywords

ODRL, Digital Policy Language, Compliance, Access Control, Rights Expression Language, Interoperability.

1. Introduction

The Open Digital Rights Language (ODRL) is a policy expression language for specifying permissions, prohibitions, and obligations over digital and physical assets. The specification is defined in two W3C Recommendations: the ODRL Information Model 2.2[1] and the ODRL Vocabulary and Expression 2.2[2]. The first document describes in plain English the underlying concepts, entities, and relationships that define the core semantics of the ODRL policies. The second document defines a general vocabulary to be used in ODRL policies plus an OWL ontology including a SKOS Concept Scheme.

The custodian of these specifications is the ODRL Community Group, whose mission is to promote and advance the W3C ODRL recommendations, but also to maintain the community of implementers, to define and support ODRL Profiles for smaller communities, to collaborate with W3C on errata maintenance, and to plan a future enhancement like the ODRL Version 3.0. The W3C is a *de facto* standardization body for web technologies, and its Recommendations are their most official specifications –as of 2025, the W3C maintains 308 of these standards.

The whole point of standards is for heterogeneous systems to interoperate. Nuts and bolts need to have the same dimensions if they are to work together, even if manufactured by different companies; an HTML developer and a web browser developer must expect the same appearance for the same HTML document. Interoperability is the key feature. The term 'interoperability' has been defined as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged"[3], or "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units"[4]. Interoperability between two systems should be guaranteed if these two systems abide to the same (good) standard. Systems adhering to one standard are *implementations* of the standard, and they are said to be *conformant* if their abidance to the standard specification has been checked. Conformance is therefore a fundamental aspect of standardization in every branch of computer science because it grants interoperability, but perhaps even more so in decentralised web

ODRL and beyond: Practical Applications and challenges for policy-base access and usage control. OPAL 2025 Co-located with the Extended Semantic Web Conference, Portorož, Slovenia · June 1 or 2, 2025.

*Corresponding author.

vrodriguez@fi.upm.es (V. Rodríguez-Doncel); nicoleta.roman@alumnos.upm.es (N. Roman)
http://cosasbuenas.es (V. Rodríguez-Doncel)
0000-0003-1076-2511 (V. Rodríguez-Doncel)

environments where the W3C usually operates in.

ODRL was created with a modest ambition in mind. The official charter for the W3C Permissions and Obligations Expression (POE) Working Group that led to ODRL 2.2 explicitly excluded access control or enforcement mechanisms[1]. In 2016 it was out of scope.

These standards have served well during this time, but there is a growing interest in the community for a higher ambition for ODRL. Interoperability is now understood with access control enforcement mechanisms within scope. The question ODRL 2.2 is answering is mostly: '*how to represent a policy that...?*'. The question that an ODRL 3.0 is expected to answer is: '*how should a piece of software behave when processing a policy?*' –the question is still aligned with the ISO and IEEE definitions. This paper does not want to answer the question, instead, it focuses on the methodology to reach an answer. For this regard, the paper is organised as follows: first conformance with the current ODRL 2.2 version is explained. Then, conformance with a possible ODRL 3.0 is described in terms of how should it be. Finally, an overview of the related work is made.

2. Conformance in ODRL 2.2

2.1. Conformance process

"2.2" is the peculiar version number given to the first official ODRL specification under the umbrella of the W3C (ODRL 1.1 was a 2002 W3C Community Note). W3C Recommendations are published as a result of a well defined process, documented in the W3C Process Document[2] and whose latest update was made in 2023. Specifications undergo several stages, starting from a First Public Working Draft, moving into different working draft versions –'Candidate Recommendation' and 'Proposed Recommendation' are the next intermediate steps. When ODRL 2.2 was published, there were specific criteria that had to be satisfied to progress from the Candidate Recommendation stage: the so called 'exit criteria'. These exit criteria specify the tests that will be prepared and executed –passing results on two implementations constitute the proof of interoperability.

The W3C POE Working Group was the group that specified ODRL 2.2 and it followed a rather standard approach, by defining use cases, requirements and validation tests for the exit criteria.

The "Use cases and Requirements" document describes 27 use cases collected from the community[3] (20 different people authored the use cases), carefully drafted following the recommendations from Cockburn [5], and leading to 38 requirements. It is worth mentioning that 41 requirements were proposed before the previous ODRL 2.0 was written in 2005[4].

Conformance to the normative part of ODRL was checked through the validation of these 38 requirements[5]. The validation function accepted as input a document with one ore more policies, plus additional information, and returned a boolean output (valid/non-valid) and optionally a comment (explanation, warning, etc.). In order to simplify the processing of the ODRL policies (validation, evaluation, or other operations) a first step of normalization was described –the validator operated only on normalised policies.

The three implementers in 2017 were Universidad Politécnica de Madrid (UPM), the International Press Telecommunications Council (IPTC) and Thomson Reuters (TR), and the performance of the implementations with respect to the exit criteria was described in the implementation report[6].

As of today, the only two official specs are the model and the vocabulary mentioned above –the last one including an OWL ontology. After their publication, the W3C Community Group has finalised only one official report presenting good practices in developing, defining and making public ODRL profiles. But a number of other documents in advanced draft state are worth to be considered: three

[1] https://www.w3.org/2016/poe/charter
[2] https://www.w3.org/policies/process/#rec-track
[3] https://w3c.github.io/poe/ucr/
[4] https://www.w3.org/2012/09/odrl/archive/odrl.net/2.0/v2req.html
[5] https://www.w3.org/2016/poe/wiki/Validation
[6] https://w3c.github.io/poe/test/implementors

profiles (for Data Spaces[7], for Big Data[8], for Temporal Data[9]), one Formal Semantics[10] and one collection of policy examples in the ODRL Implementation Best Practices[11].

Please note that besides the auspices of W3C, a number of implementations have flourished in the last few years. Most notably: (i) the profile for using ODRL with Verifiable Credentials, related to GAIA-X [12], (ii) the ODRL Profile for Access Control Policies in Solid[13], (iii) the ODRL Profile for Data Sovereignty[14], (iv) the IPTC RightsML Standard 2.0 [15], (v) the ODRL Profile for Access Control[16], (vi) the ODRL for Language Resources profile[6], (vii) the Regulatory compliance Profile[17] and (viii) the Privacy Paradigm ODRL Profile [18].

2.2. Normalisation

The ODRL specification allows multiple representations of the same information. The normalisation function applies a series of transformations to standardise policy expressions. Each transformation takes one or more policies as input and produces zero or more policies as output while preserving their overall semantics. The normalisation process consists of the following steps, which can be implemented in many possible ways –for example through SPARQL CONSTRUCT queries.

1. N1. *Internalisation of parties and assets declared out of the policy.* The ODRL model permits two ways of linking parties or assets with policies. This transformation maps external references to Policy, Assignee, or Assigner using `odrl:hasPolicy`, `odrl:assignerOf`, and `odrl:assigneeOf`, generating new RDF triples with `odrl:target`, `odrl:assigner`, and `odrl:assignee` while preserving multiple instances of these entities.
2. N2. *Type declaration of policy elements.* This transformation infers class types for Policy, Set, Permission, Duty, Constraint, and LogicalConstraint based on the ODRL Ontology and Information Model 2.1.1 rules, ensuring correct classification even when using an OWL reasoner.
3. N3. *Application of inheritance rules.* This transformation applies the inheritance rules from the Information Model Section 2.9, ensuring policies become self-contained and valid by resolving `inheritsFrom` relationships in topological order and removing or relocating them as metadata.
4. N4. *Interiorizing policy-level properties.* This transformation process redistributes policy-level properties to each of the top-level Rules while removing them from the Policy level itself. Top-level Rules are those directly associated with the Policy and not nested within other rules. Examples of nested or second-level Rules include Duties within a Permission, as well as failure consequences or remedies.
5. N5. *Expansion from compound to irreducible rules.* This transformation expands any compound Rule (with two or more Assets, Assigners, Assignees or Actions) into a set of Rules each of them with at most one of these elements. The Policy Rule Composition in the IM describes accurately how to make this transformation.
6. N6. *Inferences derived from odrl:includedIn* This transformation transforms every Rule including the action B by adding the property-value `action` A for every A where A |includedIn| B. If an Action A is included in another Action B (`A includedIn B.`), all Rules defined for B must also hold for A, but not necessarily vice versa.

[7]https://w3c.github.io/odrl/profile-dataspaces/
[8]https://w3c.github.io/odrl/profile-bigdata/
[9]https://w3c.github.io/odrl/profile-temporal/
[10]https://w3c.github.io/odrl/formal-semantics/
[11]https://w3c.github.io/odrl/bp/
[12]https://gitlab.com/gaia-x/lab/policy-reasoning/odrl-vc-profile
[13]https://protect.oeg.fi.upm.es/solid-consent/
[14]https://profile.mydata-control.de/ods/
[15]IPTCRightsMLStandard2.0
[16]https://w3id.org/oac/
[17]https://ai.wu.ac.at/policies/orcp/regulatory-model.html
[18]https://w3id.org/ppop

7. N7. *Inferences derived from odrl:implies* This transformation updates every Permission or Duty with action A by adding the property-value `odrl:action` B for each A where A `odrl:implies` B. If an action A implies another action B (A `odrl:implies` B), a Prohibition of B conflicts with a Permission of A, but not necessarily vice versa (i.e., a Prohibition of A does not conflict with a Permission of B).
8. N8 and N9. *Inferences derived from odrl:partOf for Asset Collection and Party Collection* This transformation changes every Rule including the Asset B with the property `odrl:target` by adding the property-value `odrl:target` A for every A where `A odrl:partOf B`. The same applies to Party Collection.
9. N10. *Policy Replacement.* This transformation applies to every policy A if the triple A `isReplacedBy` B exists. It involves two steps: removing all rules of A, and if policy B is identified by a dereferenceable URI, retrieving B and adding its rules to A.

Up to date, the UPM implementation can determine whether an ODRL policy is normalised or not. However, this check is not made by any official W3C piece of software. We believe normalisation in ODRL ensures that policies with equivalent meanings are represented in a consistent form, facilitating comparison, interoperability, and automated reasoning. Also, normalisation prevents from the need of performing OWL reasoning at later stages. New features of the ODRL language may entail new normalisation rules.

2.3. Validation

The validation function could be implemented as a sequence of individual validation operations –and SHACL shapes were provided for most of them. The order of the validation operations was irrelevant, and the result of each validation operation was a simple true or false. All the validations function took one or more Policies as input.

These were the validations that were considered as exit criteria to evaluate the different implementations. In addition, other two validations were defined, which could not be tested with SHACL: whether a policy together with the ontology was consistent under an OWL reasoner, and the strict validation of cardinalities.

As an example, a SHACL shapes implementing one of the rules is shown, extracted from Information Model Section 2.6.7.

```
odrl:RemedyShape
    a sh:NodeShape ;
    sh:targetObjectsOf odrl:remedy ;
    sh:property [
        sh:path odrl:consequence ;
        sh:maxCount 0 ;
    ] .
```

Listing 1: Implementation of validation rule V8

3. Conformance in ODRL 3.0

This section describes some possible ODRL processors, i.e. use cases and proposes the type of requirements to be defined towards achieving a conformance mechanism in ODRL 3.0

3.1. Use Cases

ODRL 2.2 specified how an ODRL policy is structured. ODRL 3.0, as many understand it, should specify how ODRL processors behave. The behaviour of software is what is standardised. However, different pieces of software can be conceived. If use cases for ODRL 2.2 primarily focused on representation needs, use cases for ODRL 3.0 must also address processing requirements.

Validation rule
V1. A valid document must contain at least one Policy
V2. Every policy must have at least one rule
V3. Every policy must be identified by a URI
V4. Every rule in an offer must have exactly one assigner
V5. Every rule in an agreement must have exactly one assigner
V6. Every rule in an agreement must have exactly one assignee
V7. Every rule must have exactly one target
V8. No remedy can refer to a duty that includes a consequence duty
V9. Every AssetCollection with a refinement must have a source
V10. Every PartyCollection with a refinement must have a odrl:source
V11. In absence of profile, only ODRL Core conflict strategies can be used
V12. In absence of profile, only ODRL actions can be used
V13. In absence of profile, only ODRL left operands can be used
V14. In absence of profile, only ODRL operators can be used
V15. In absence of profile, only ODRL logical constraints can be used
V16. Every constraint must have exactly one right operand
V17. Every constraint must have exactly one left operand
V18. Every constraint must have exactly one operator
V19. The values of every logical constraint must be constraints
V20. The values of the property rightOperandReference must be URIs
V21. Every value of the property odrl:datatype must be of type rdfs:Datatype
V22. Every prohibition must be disjoint with permissions and obligations
V23. Every policy must have at most one conflict strategy
V24. No policy can inherit directly or indirectly from itself

Table 1
Validation Rules in ODRL2.2

ODRL processors can do a very diverse range of things with ODRL policies, but only some of them may deserve the effort of being standardised. Among those that might be standarisable, the following can be mentioned. Please note that one single ODRL processor may give rise to several actual use cases, but for the sake of brevity, this section identifies 'ODRL processor' with 'use case'.

1. *ODRL Validator*. Piece of software that computes whether a certain policy is syntactically valid or not. Please note that there is a definition in the current ODRL 2.2 Information Model: "A system that checks the conformance of ODRL Policy expressions, including the cardinality of properties and if they are related to types of values as defined by the ODRL Information Model, and the Information Model's validation requirements." Syntactic validity is determined by the 24 rules listed above –but please note that other syntactic checks can be made.
2. *ODRL Normaliser*. Piece of software that determines whether an ODRL is its normal form or not. The software can also convert any valid ODRL policy into a *normalised ODRL policy*. Whereas there is currently no formal definition of the normalised ODRL policy, this is something ODRL 3.0 should include. This normalisation functionality was implemented in the UPM ODRL validator, but never exposed on its own. The ODRL normaliser should, for example, run an OWL reasoner, and make explicit the triples that can be inferred. The normalised policy can thus be processed by a JSON parser, instead of having to include OWL reasoning. Example of ODRL 2.2: if a permission is granted to execute the action `odrl:transfer`, then it is also granted the permission to execute `odrl:give` and `odrl:sell`. ODRL parsers should not have to care with these inferences.
3. *ODRL Converter*. Piece of software that transcodes ODRL policies across the different serialisations. Whereas the serialisation of an ODRL policy (JSON-LD, XML/RDF, NT, Turtle...) can be mostly

automated –some details related to anonymous nodes or the different JSON varieties do not grant homomorphic transformations–, an advanced ODRL converter should also transform ODRL expressions in different ODRL versions (1.1, 2.0, etc.). Please note that currently, a policy cannot declare to which ODRL version it abides to.

4. *ODRL Profiler*. Piece of software that determines to which ODRL Profile a certain ODRL policy conforms to.
5. *ODRL Authoriser*. Piece of software that, departing from a set of policies, a request, and state of the world (any context information including history), determines whether access should be granted or not. Please note that the authoriser might answer these slightly different questions:
   - Do I have access to a certain resource? (a query is made but access is not granted)
   - Can you please give me access to the resource (access is granted if rightful)
   - What do I need in order to access to a certain resource?
6. *ODRL Monitor*. Piece of software that, departing from a set of policies and the history of a world, determines whether violations have happened –a policy monitoring scenario where ODRL policies can be violated is understood.
7. *ODRL conflict resolutor*. Piece of software that identifies identifies and resolves conflicts between multiple ODRL policies, or that helps license composition[7]. For example, if policy A obliges doing a certain action and policy B forbids doing that action, then the set of policies that includes A and B is inconsistent. This satisfiability analyser would offer a conflict resolution system, ensuring policies are logically consistent.

Please note there is a very broad range of applications dealing with ODRL that might be useful, but that are out of scope for the Working Group. For example: (i) *ODRL Exporter/Importer.* Piece of code that transforms an ODRL policy from/to another policy language. A case of particular relevance would be the transformation from ODRL policies into executable smart contracts –in a decentralised and transparent way. Please note that equivalent transformations have already been programmed for the much-similar MPEG-21 technologies[8]. (ii) *ODRL Policy Compliance Checker.* Piece of code that determines whether a certain ODRL policy abides with legislation at a certain time within a certain jurisdiction. ('Does the ODRL policy align with GDPR? '). (iii) *ODRL Visualizer.* A tool to graphically represents ODRL policies in an intuitive way. The direct visualisation of policies is very soon unpractical, as already pointed out by marketdata.md, so intelligent tools are necessary to visually summarise the main aspects of a policy. (iv) *ODRL Policy generators.* Tools to create ODRL policies, either visually, or with a text processed by an LLM-based software: 'Grant access to video files for one month, after which access expires'. (v) *ODRL Policy Explainers* Tools that take an ODRL Policy as input and create a human-readable explanation.

3.2. Requirements

The specification of interoperability requirements for ODRL-based software can be structured into distinct categories to facilitate a systematic approach. These categories would include the requirements related to the data model and syntax, the requirements related to the ODRL processors, and the requirements related to their interoperation through APIs and communication protocols.

Requirements related to the ODRL policy. The first category of requirements is concerned with the definition of the data model and the syntactic structure of ODRL policies, much as in ODRL 2.2. Although requirements are expressed in English, SHACL shapes should be easily derived. Besides the ODRL 2.2, new elements should support the different ODRL processors. The satisfaction of these requirements should lead to being able to say: 'this *ODRL prolicy* is conformant with ODRL 3.0 or not'.

Requirements related to the ODRL processors. Different requirements should be specified for each of the ODRL processors: *validator*, *normaliser*, *converter*, *profiler*, *authoriser*, *monitor*, *conflict resolutor*. The satisfaction of these requirements should lead to being able to say: 'this ODRL processor is a normaliser and validator, but not an authoriser' etc. These would be the *conformance levels* of ODRL.

Requirements related to message exchange. ODRL processors do not work alone if they are to interoperate with other ODRL processors. The interoperability mechanisms constitute the third category, addressing the means by which these processors communicate and exchange policies and other messages (e.g. an authorisation results). Protocols may take the form of UML sequence diagrams and UML state machine diagrams. Alternatively, the specification may define a simple HTTP Rest API that ODRL implementers may abide to –the use of HTTP messages in the Linked Data Platform should be a reference design. Additionally, a standardised data exchange format must be established to ensure consistency in policy processing across platforms.

3.3. Validation and Conformance

The W3C provides 13 Requirements and 23 Good Practices on how W3C Recommendations should be[19], and their very first Requirement is: 'Include a conformance clause.':

> The conformance clause provides the answers to the important questions: what may conform and how? The conformance clause defines at a high-level, what is required of implementers of the specification. [...]. The conformance clause may partition the technology into functional subsets, such as profiles, modules or other structures [...].

Conformance is an attribute that can be given to ODRL policies and to ODRL processors. To ensure conformance in ODRL-based implementations, validation and conformance mechanisms must be explicitly defined for the policy and for each of the processors .

A suite of conformance tests should be developed to assess adherence to the specification, providing a benchmark for evaluating software implementations. Please note that if an ODRL 3.0 is formally defined, exit criteria will be necessary, and these tests will be published as formally as possible. Indeed, the W3C recommends creating different conformance labels: 'Create conformance labels for each part of the conformance model' is their sixth requirement.

In addition to these validations, other documents should help implementers. New best practice documents should be drafted –the current Best Practices document only describe how to define profiles and how to represent policies for different situations. If other W3C specifications have a 'Primer', in this new, more complex scenario, having a 'ODRL Primer' would be advisable as well. Finally, it is worth noting that an ODRL Test Suite was started in 2024, and the final spec could follow its style supported by the github repo[20].

4. Related work

4.1. Conformance

In the context of the World Wide Web Consortium, conformance plays a critical role in guaranteeing that web technologies function as intended across diverse environments. W3C standards, from HTML and CSS to SPARQL, rely on well-defined conformance criteria to ensure that different implementations interpret and process specifications correctly.

How is conformance validated in other W3C semantics documents? Well, the W3C has produced several 'Semantics' documents each of them with a different objective. The 'RDF1.1. Semantics'[21] defines a model-theoretic semantics to determine the validity of RDF inference processes. RDF processors must respect RDF entailment rules, meaning they should correctly infer implicit triples from given RDF data –conformance is tested with these rules. A similar approach is followed by the OWL Semantics, a recommendation providing the direct model-theoretic semantics for OWL 2 and defining the most common inference problems.

[19]https://www.w3.org/TR/qaframe-spec
[20]https://w3c.github.io/poe/testsuite/
[21]RDF 1.1 Semantics W3C Recommendation 25 February 2014

SHACL (Shapes Constraint Language) is a W3C standard for validating RDF data against a set of constraints to ensure data quality and consistency. Conformance to the W3C SHACL specification is validated through a test suite[22].

XPath (XML Path Language) is a language that can be used to clearly refer to elements and attributes in an XML document. XQuery (XML Query) is a query and functional programming language to query XML data. The "XQuery and XPath Formal Semantics"[23] intends to complement the specification by defining the meaning of XQuery/XPath expressions with mathematical rigor; thus clarifying the intended meaning of the English specification, and ensuring that no corner cases are left out. For that regard grammar productions are given. Conformance is validated with test suites, the reader is invited to check their .zip file[24].

The POWDER specification provides a mechanism to describe and discover Web resources, and it also includes a 'Formal Semantics' document[25]. Conformance to the specification is, again, described in terms of a test suite[26].

4.2. Current initiatives

This section presents several publicly described with recent activity. Most of them focus on the ODRL authoriser.

Open Digital Rights Enforcement Framework. Another implementation is the ODRE (Open Digital Rights Enforcement Framework) project[9], whose main focus is the development of a solution to define, use and enforce privacy policies in digital environments. Of most interest for this work is that ODRE framework includes an enforcement algorithm for ODRL policies and two open-source implementations in Python and Java –the vocabulary and mechanisms for conformance are well defined. This framework is further described in this CEUR volume.

Ghent's ODRL Evaluator. Wout and Esteves' recent work on the interoperable interpretation and evaluation of ODRL Policies[10] tries to overcome the limitations of current authorization protocols such as Web Access Control (WAC) or Access Control Policies (ACP) by providing fine-grained access control. The most remarkable feature for this work is the complete definition of everything that is necessary to provide test suites –they provide plenty of examples of sets that include an ODRL policy, a request, and the state of the world as input, along with the compliance report as output. This framework is also further described in this CEUR volume.

marketdata.md Project. The next implementation is the marketdata.md project, which is a digital rights SaaS solution designed to manage critical business data and ensure compliance in this area. This project is of interest for this work because it goes beyond the mere authorisation, proving the need for other ODRL processing tasks. Most notably, its studio workspace is a tool with user interface created to simplify the creation of valuable data products and policies, abstracting away technical details from the user. Additionally, a significant addition is the recommendation system which recommends ways to reuse components based on a recommendation system connected to all data. This knowledge base increases when other parties publish to everyone's advantage. Finally, this platform also addresses the challenges in managing digital rights in a data-rich environment, where custom contracts with embedded policies are not machine-readable.

Gaia-X Wizard. Gaia-X Wizard is a 2024 implementation of ODRL 2.2. It is a web application that aims to guide the users in creating and signing Verifiable Presentations (VPs) as well as to obtain Verifiable Credentials (VCs) that demonstrate Gaia-X compliance. Some of the offered functionalities are: (i) *Environment selection*, where users can choose between Clearing Houses (production environment requiring certificates from Trust Anchors) and the Gaia-X development environment (allowing

[22]https://w3c.github.io/data-shapes/data-shapes-test-suite/
[23]XQuery 1.0 and XPath 2.0 Formal Semantics (Second Edition) W3C Recommendation 14 December 2010 (revised 7 September 2015)
[24]https://www.w3.org/XML/xquery/test-suite/
[25]Protocol for Web Description Resources (POWDER): Formal Semantics W3C Recommendation 1 September 2009
[26]https://www.w3.org/TR/powder-test/

certificates from other providers), (ii) *Playground*, a section that enables users to create and sign any type of Verifiable Credential using provided JSON-LD examples, starting from scratch, or importing their own signed Verifiable Credentials, (iii) *Onboarding*, which guides users through the creation of specific Verifiable Credentials such as Legal Participant, Legal Registration Number, and Terms and Conditions, requiring the completion of forms or agreement to terms, (iv) *Stepper*, a helpful step-by-step form for creating Verifiable Credentials, (v) *Legal Registration Number*, a feature that allows users to input their number, which is then added to the "Holder" section and the Local Wallet, (vi) *Local Wallet*, a tool for easily storing signed Verifiable Credentials, and (vii) *Gaia-X Compliance API*, which enables users with the required Verifiable Credentials in their "Holder" section to obtain verified Verifiable Credentials.

5. Conclusions

It is fair to say that ODRL2.2 was created to satisfy the needs of a very wide community –*20 people* contributed use cases. The derived requirements did not include authorisation issues and the current specifications of ODRL have served well. However, there is a new breed of ODRL users who are demanding higher levels of interoperability. This paper has outlined how the conformance checks in these advanced scenarios should be.

Conformance is particularly important for ODRL 3.0, and with some open challenges in decentralised, asynchronous, environments. There is an opportunity for a healthy ODRL-based software ecosystem, where implementations take advantage of each other. But for this ecosystem to exist, there must be a conformance guarantee that ensures that policy engines, access control systems, and reasoning frameworks correctly interpret rights, permissions, and obligations. Without robust conformance mechanisms, discrepancies in policy enforcement may lead to interoperability issues, misinterpretation of usage rights, or even regulatory non-compliance –the ecosystem would miss the potential for software reuse. Therefore, the final message of this work is that the future ODRL 3.0 spec should be grounded in use cases, leading to clear requirements (every requirement must originate in one or more use cases). Conformance tests should be directly related to the requirements, and verifiable through test cases.

Acknowledgments

This work has been partially funded by the project HARNESS, which has received funding from the EU's Horizon 2020 research and innovation programme under grant agreement no. 101169409, see https://www.harness-network.eu

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to do: Grammar and spelling check.

References

[1] Renato Iannella and Serena Villata, ODRL Information Model 2.2, World Wide Web Consortium (W3C) Recommendation, 2018. URL: https://www.w3.org/TR/odrl-model/.

[2] Renato Ianella, Michael Steidl, Stuart Myles and Victor Rodriguez-Doncel, ODRL Vocabulary and Expression 2.2, World Wide Web Consortium (W3C) Recommendation, 2018. URL: https://www.w3.org/TR/odrl-vocab/.

[3] IEEE, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, Institute of Electrical and Electronics Engineers, New York, NY, USA, 1990. IEEE Std 610.

[4] ISO/IEC, Information technology – Vocabulary – Part 1: Fundamental terms, International Standard ISO/IEC 2382-1:1993, International Organization for Standardization, Geneva, Switzerland, 1993. Withdrawn standard.

[5] A. Cockburn, Writing Effective Use Cases, Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. URL: http://alistair.cockburn.us/get/2465.

[6] V. Rodríguez-Doncel, P. Labropoulou, K. Gkirtzou, Rights management for language resources iii workshop, in: INTELE, Madrid, 2022. URL: https://doi.org/10.5281/zenodo.7140773.

[7] S. Villata, F. Gandon, Licenses compatibility and composition in the web of data, in: Third International Workshop on Consuming Linked Data (COLD2012), 2012.

[8] M. Zichichi, V. Rodriguez-Doncel, Encoding of media value chain processes through blockchains and mpeg-21 smart contracts for media, IEEE MultiMedia 22 (2023) 1–8.

[9] A. Cimmino, J. Cano-Benito, R. García-Castro, Open digital rights enforcement framework (odre): from descriptive to enforceable policies, Computers & Security 150 (2025) 104282.

[10] Wout Slabbinck, Julián Andrés Rojas, Beatriz Esteves, Pieter Colpaert, Ruben Verborgh, Interoperable Interpretation and Evaluation of ODRL Policies, in: Accepted for Semantic Web - 22nd International Conference, ESWC 2025, 2025.