Introducing Policy Templates and Policy Variables in ODRL

Andrea Cimmino
Universidad Politécnica de Madrid
andreajesus.cimmino@upm.es

Nicoletta Fornara
Università della Svizzera italiana
nicoletta.fornara@usi.ch

Abstract

ODRL 2.2 provides a W3C Recommendation for expressing policies over digital assets, including permissions, prohibitions, duties, constraints, parties, and assets. However, several enforcement scenarios require policies whose concrete values are not fully known at design time. This position paper proposes introducing the concepts of `PolicyTemplate` and `PolicyVariable` to represent reusable ODRL-based policy structures that are instantiated into ODRL 2.2 compatible policies during enforcement. The proposal is immediately applicable to ODRL 2.2 as a lightweight profile or ontology extension, while its full value would emerge in a future version of ODRL where policy templates, variables, bindings, and template instantiation could be standardised as first-class modelling and enforcement concepts.

1 Position Statement

The Open Digital Rights Language (ODRL) is a W3C Recommendation for expressing policies over digital assets, including permissions, prohibitions, duties, constraints, parties, and assets [1, 2]. Its adoption in domains such as data spaces, access control, usage control, and knowledge graph governance shows the relevance of having a standard RDF-based policy language [3–5]. However, many practical enforcement scenarios require policies whose specific values are not fully known when the policy is authored. For instance, the assignee of an obligation or permission may be determined by the actor triggering an interaction, a deadline may be computed from the activation time of a policy, or the target of a rule may be obtained dynamically from the evaluation request or from the state of the world [4, 5]. In such cases, policy authors do not need to define a complete ODRL Policy directly, but rather a reusable policy structure that can be instantiated into a specific policy before enforcement.

This position paper proposes the introduction of two explicit concepts into the ODRL Information Model: `PolicyTemplate` and `PolicyVariable`. The former would represent a reusable ODRL-based policy structure containing one or more placeholders/variables, while the latter would represent the placeholders whose values are injected at run time. This proposal builds on previous work identifying policy templates and variables as a missing feature in ODRL, particularly for policies that contain values grounded during the evaluation of policy activation [4]. It also aligns with the broader need to bridge the gap between descriptive policy representation and enforceable policy behaviour, which has been highlighted in previous ODRL enforcement works [4, 5].

2 Policy Templates and Policy Variables

The core modelling proposal is to extend the ODRL ontology [1, 2] with the following OWL classes: `odrl:PolicyTemplate` and `odrl:PolicyVariable`. This proposal is compatible with ODRL 2.2 because it can be implemented as an ODRL profile or lightweight ontology extension. In such an implementation, a `PolicyTemplate` would be an ODRL-compatible RDF structure that contains variables, while `PolicyVariable` individuals would be used in those positions where concrete values must be provided during evaluation. The same variable can be used in different parts of the same policy, enabling fundamental links to be established between the various parts of a policy, for example, between a permission and the associated duties. For example, a newspaper's website policy, formalized in the following, may require users to click the "Accept cookies" link in order to be granted permission to view the homepage.

A `PolicyVariable` should have a stable URI, a human-readable description, and a datatype specification through a property such as `odrl:variableType`, whose value refers to an XML Schema datatype, for example `xsd:anyURI`, `xsd:string`, `xsd:dateTime`, `xsd:duration`, `xsd:boolean`, or `xsd:decimal` [6]. Variables may also be classified according to the ODRL position in which they appear, such as assignee variables, target variables, right-operand variables, deadline variables, or contextual variables.

```
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex:   <http://example.org/policies/> .

ex:CookiePolicyTemplate
    a odrl:PolicyTemplate ;
    odrl:permission [
        odrl:target ex:newspaperHomePage ;
        odrl:action odrl:read ;
        odrl:assignee ex:readerVariable ;
        odrl:duty [
            odrl:action ex:acceptCookie ;
            odrl:assignee ex:readerVariable ;
            odrl:target ex:AcceptCookiesLink
        ] ;
        odrl:constraint [
          a odrl:Constraint ;
          odrl:leftOperand odrl:dateTime ;
          odrl:operator odrl:lteq ;
          odrl:rightOperand ex:SessionCookieExpirationTimeVariable
        ]
    ] .

ex:newspaperHomePage a odrl:Asset .
ex:AcceptCookiesLink a odrl:Asset .
ex:acceptCookies a odrl:Action .

ex:readerVariable
    a odrl:PolicyVariable ;
    odrl:variableType xsd:anyURI, odrl:Party  ;
    rdfs:comment "The identifier of the agent that accepted the cookie." .

ex:SessionCookieExpirationTimeVariable
    a odrl:PolicyVariable ;
    odrl:variableType xsd:dateTime ;
    rdfs:comment "The expiration time of the accepted session cookie." .
```

Following this approach, during the evaluation of a policy template, variables are substituted with concrete values. For example, when the user `ex:alice` clicks on the "Accept cookies" link, `ex:alice` gets the permission to read the web page for a certain period of time that is extracted from the accepted session cookie. The evaluation request or the state of the world may provide bindings between variable URIs and concrete values.

3 Compatibility with ODRL 2.2 and Future ODRL Versions

The proposed mechanism does not require waiting for a future version of ODRL to be tested in practice. It could already be implemented on top of ODRL 2.2 by defining the two classes and related properties in a separate namespace, for example `odrlx:PolicyTemplate`, `odrlx:PolicyVariable`, and `odrlx:variableType`. Existing ODRL engines should be able to evaluate policy templates using the same procedure as for evaluating `odrl:Policy` written in ODRL 2.2, with the addition of a step in which the variables contained in the policy template can be replaced by constant values.

The resulting behavioural pipeline can be described as follows: (1) a `PolicyTemplate` is authored and published; (2) an evaluation request or knowledge graph representing the state of the world provides bindings for the variables during the policy template evaluation; (3) the policy template is instantiated by replacing each `PolicyVariable` with its corresponding value; (4) the result is a concrete `odrl:Policy`; and (5) the generated policy is evaluated by an enforcement engine. Before specific values are assigned to the variables, an engine can check whether all required variables are bound and whether each value conforms to the datatype declared through `odrl:variableType`. This is particularly important for values such as deadlines, assignees, targets, spatial constraints, or numerical thresholds, where incorrect datatypes may lead to inconsistent enforcement results [4, 5].

In ODRL 2.2, mainly for compatibility reasons, the behaviour of a `PolicyTemplate` could be kept deliberately simple: given one policy template, the instantiation process may generate either one concrete `odrl:policy` or, in some cases, several concrete policies. However, if policy templates were designed from scratch as first-class citizens in a future version of ODRL, their behaviour could be considerably richer, in line with previous work on enforceable ODRL policies [5]. For example, a policy template could contain conditional fragments that are instantiated only when certain conditions hold, or it could define mutually

exclusive instantiation branches so that one policy or another is generated depending on the evaluation context. Therefore, although a lightweight interpretation of policy templates is sufficient for ODRL 2.2 compatibility, a more expressive account of their behaviour would require discussion and consensus in a future revision of ODRL.

Indeed, a future version of ODRL could standardise the vocabulary, expected behaviour, and validation rules for templates and variables. This would avoid incompatible ad hoc implementations and would allow different policy authoring tools, Policy Decision Points, and Policy Enforcement Points to share the same interpretation of variable binding and template instantiation. In particular, a future ODRL specification could define whether a template is a subclass of `odrl:Policy` or it is a separate class; how a `PolicyTemplate` is related to the generated `odrl:Policy`; whether variables are required or optional; how datatype conformance is checked; and how bindings are represented in an evaluation request. Thus, ODRL 2.2 is sufficient for prototyping and adoption, but a future version of ODRL would provide the right place to make this mechanism interoperable and normative.

4 Relation to Ontology Template Instantiation

This proposal is related to existing ontology engineering practices based on template instantiation. In particular, Reasonable Ontology Templates (OTTR) provide a relevant precedent for representing ontology modelling patterns as parameterised templates that can be instantiated into RDF graphs and OWL ontologies [7–9]. ODRL should not necessarily adopt OTTR as a dependency, but OTTR demonstrates that template-based instantiation is already an established approach for RDF/OWL artefacts. Therefore, the relationship between a `PolicyTemplate` and the generated `Policy` should be understood as an RDF/OWL template-instantiation relation, rather than as an adaptation of the software engineering Prototype pattern. To the best of our knowledge, there is no widely established ontology design pattern corresponding directly to the Prototype pattern with the precise semantics required here.

5 Expected Benefits

The expected benefits are threefold. First, policy templates reduce duplication by allowing recurring policy structures to be reused across actors, targets, assets, and contexts. Second, policy variables make dynamic enforcement explicit and machine-processable, avoiding ad hoc placeholders encoded in strings, blank nodes, or implementation-specific conventions. Third, the approach makes it possible to create references between different rules of a policy.

The proposed extension is deliberately small. It requires, at minimum, two new OWL classes, `odrl:PolicyTemplate` and `odrl:PolicyVariable`, and one datatype-oriented property such as `odrl:variableType`. Further properties could be discussed by the ODRL community, for instance to indicate whether a variable is required, where it may occur in a policy, how it is bound, or whether its value is obtained from the request, from the state of the world, or from a computation. This provides a pragmatic path towards future ODRL evolution: supporting dynamic, reusable, and enforceable policy structures while keeping compatibility with existing ODRL policies and enforcement architectures.

References

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

[2] Renato Iannella, Michael Steidl, Stuart Myles, and Víctor Rodríguez-Doncel. ODRL Vocabulary & Expression 2.2. W3C Recommendation, 2018. URL https://www.w3.org/TR/odrl-vocab/.

[3] W3C. W3C Workshop on the Future of ODRL, 2026. URL https://www.w3.org/2026/ODRLWS/.

[4] Andrea Cimmino and Nicoletta Fornara. Improving ODRL 2.2: Current Limitations and Theoretical Solutions. In *ODRL and Beyond: Practical Applications and Challenges for Policy-based Access and Usage Control (OPAL 2025), co-located with ESWC 2025*, 2025.

[5] Andrea Cimmino, Jorge Cano-Benito, and Raúl García-Castro. Open Digital Rights Enforcement Framework (ODRE): From Descriptive to Enforceable Policies. *Computers & Security*, 150:104282, 2025. doi: 10.1016/j.cose.2024.104282.

[6] W3C. XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. W3C Recommendation, 2012. URL https://www.w3.org/TR/xmlschema11-2/.

[7] Martin G. Skjæveland, Henrik Forssell, Johan W. Klüwer, David Lupp, Evgenij Thorstensen, and Arild Waaler. Pattern-Based Ontology Design and Instantiation with Reasonable Ontology Templates. In *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017)*, volume 2043 of *CEUR Workshop Proceedings*, 2017. URL https://ceur-ws.org/Vol-2043/paper-09.pdf.

[8] Martin G Skjæveland, Daniel P Lupp, Leif Harald Karlsen, and Henrik Forssell. Practical Ontology Pattern Instantiation, Discovery, and Maintenance with Reasonable Ontology Templates. In *The Semantic Web – ISWC 2018*, volume 11136 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2018. doi: 10.1007/978-3-030-00668-6_29.

[9] Martin Georg Skjæveland and Leif Harald Karlsen. The Reasonable Ontology Templates Framework. *Transactions on Graph Data and Knowledge*, 2(2), 2024. doi: 10.4230/TGDK.2.2.5. URL https://drops.dagstuhl.de/entities/document/10.4230/TGDK.2.2.5.