ODRL 2.0 Revisited

Stefan Becker, Benjamin Hück, Katharina Naujokat, Arne F. Schmeiser, Andreas Kasten

University of Koblenz-Landau
Universitätsstraße 1
56070 Koblenz
{stefanbecker, bhueck, knaujokat, afschmeiser, andreas.kasten}@uni-koblenz.de

Abstract: The Open Digital Rights Language (ODRL) is one major rights expression language which has been in development since 2001. ODRL allows to define policies which describe who is allowed to do what with a good. Despite its long history, ODRL 2.0 still has some drawbacks concerning the syntax and semantics of its entities and relations. This paper presents the results of an in-depth analysis of these shortcomings and outlines possible solutions. These include proposals to simplify the Core Model of ODRL to be better understood by new users as well as to prepare ODRL for the usage with encodings other than XML.

1 Introduction

The increasing usage of digital goods requires mechanisms for describing usage rights and access rights. Also a solution is needed to specify value chains to assign rights to third parties like producers, publishers, and consumers of such goods. Therefore, several rights expression languages (RELs) like MPEG-21 REL [Int04], ccREL [AALY08], ODRL [IGPK12] as well as different access control languages like XACML [Ris13] have been developed. A policy defines rights as actions which a user may or must not perform on a particular good. MPEG-21 REL is mostly used for describing usage rights for audio and video files. XACML allows the description of access rights for various assets so that only permitted parties can access goods. ODRL neither focuses on a particular media type nor a specific use case and can therefore be used for different scenarios.

The current version 2.0 of ODRL was published in 2012. The structure of an ODRL policy is defined in ODRL's Core Model [IGPK12] and a set of general terms for creating such policies is presented in the Common Vocabulary [IG12]. The language aims for a general description of usage rights of physical and digital goods. ODRL describes permissions and prohibitions that a party has with relation to such a good. Since ODRL does not focus on a particular use case, it can be used in many different scenarios. Although this results in a great flexibility and expressiveness, it also poses some drawbacks concerning the reusability and precision of the Core Model and the Common Vocabulary. This paper presents an in-depth analysis of the Core Model and the Common Vocabulary of ODRL 2.0. The analysis is based on other work such as [AH05] as well as on own experience from implementing an application using ODRL [BBH+13]. It presents identified

drawbacks and, where possible, outlines potential solutions. The paper is structured as follows: Section 2 outlines the basic concept of a rights expression language and Section 3 describes the current design of ODRL 2.0. Section 4 summarizes previous work discussing ODRL 2.0, analyzes the current ODRL design, and sketches possible improvements. The paper concludes with further suggestions of how ODRL may evolve in the future.

2 Rights Expression Languages

Rights expression languages (RELs) enable formal communication between rights holders and consumers. A rights holder describes which actions a consumer may perform on a physical or digital good. A REL allows for unambiguously [RTM02] describing usage policies which can directly be interpreted by an application program such as a *Digital Rights Management System* (DRMS). A REL has a syntactic and semantic structure [Bar06, Gut03]. The syntactic structure defines how its main entities are aligned and how they should be used. It covers parties which represent persons or organizations, allowed or prohibited actions, and the actual good which is often referred to as *asset*. The syntactic structure is often associated with a vocabulary which contains, e. g., the names and definitions of usable actions. Such a vocabulary is called a *Rights Data Dictionary* (RDD). Actions like "print" can be distinguished from constraints like "not more than three times". Conditions such as paying a fee must be fulfilled in order to acquire a right.

As RELs can be used in different domains, e. g., rights management of digital media or digital representation of contracts, the model and vocabulary of a REL might not fit all purposes. Therefore, many languages support extending or restricting their default REL and/or RDD for "a particular application" [Bar06]. These adjustments are defined as a *profile*. The concept of ODRL is presented in the next section. MPEG-21 REL [Int04] is another REL focusing on licenses for audio and video material. Creative Commons develop an own REL to express their licenses: ccREL [AALY08] consists of a small set of terms which can be embedded into a web page or a binary file. METSRights[1] defines a small set of metadata elements to extend the XML-based library standard METS [Cun04] with usage rights expressions. A different approach is taken by the access control language XACML [Ris13]. To decide whether to allow or deny access to a resource, XACML licenses are built upon boolean functions that are evaluated when a user requests access to that resource. General purpose languages follow a broader approach than RELs in order to cover several application scenarios such as access control or flow control. KAoS [UBJ+03], Rei [KFJ03], and Ponder [DDLS01] are examples for such languages.

3 ODRL 2.0 Summarized

ODRL 2.0 is specified in three documents: the Core Model [IGPK12], the Common Vocabulary [IG12], and the XML encoding [Ian12]. The Core Model and the Common Vo-

[1] `http://www.loc.gov/standards/rights/METSRights.xsd`

cabulary express the structure and semantics of ODRL and are its REL and RDD, respectively. The XML encoding defines a possible serialization for ODRL policies. However, the policies are not bound to a specific serialization and can be encoded in different formats. In April 2013, a draft for a JSON encoding was published [Öbe13]. This paper solely focuses on how to express rights using ODRL policies without covering their serialization. The following sections describe the usage of ODRL, compare it with other RELs and explain the Core Model and the Common Vocabulary.

3.1 Usage of ODRL and Comparison with Other RELs

The rights expression language ODRL [IGPK12] does not focus on a particular scenario and can be used within different use cases. For example, RightsML [IPT12] and OMA DRM [OMA08] provide ODRL profiles which adapt ODRL for their specific needs. RightsML is developed by the *International Press Telecommunications Council* (IPTC)[2] to express usage rights for the media industry. Newspaper publishing companies receive content from agencies which is subject to restrictions that have to be adhered to. RightsML provides means to express these usage rights in a machine-readable form to allow machine-based evaluation. The *Open Mobile Alliance* (OMA)[3] uses ODRL as a foundation for its own DRMS. OMA DRM focuses on mobile scenarios. It allows rights holders to express usage rights for their content, which can be enforced by the DRMS. OMA DRM is applicable, e. g., for ring tones, music, or streaming media.

ODRL was also used in the research project ROX [BBH+13] of the *University of Koblenz-Landau*. ROX provides a collection of tools to define and visualize ODRL policies for content on web pages like images or videos. The policies are created with an editing tool and visualized with a browser add-on.

The main advantage of ODRL is its expressiveness compared to other RELs. With ODRL rights holders can define which actions are allowed and which are prohibited related to a physical or digital good. This feature sets ODRL apart from languages like MPEG-21 REL [Int04] and METSRights [Cun04], which can only express allowed actions. Furthermore, ODRL allows to define multiple rights holders in the same policy whereas other languages such as MPEG-21 REL have only one rights holder in a policy. Moreover, ODRL allows to define a specific licensee. On the other hand, ccREL cannot express a licensee. Instead, each ccREL policy applies for everyone.

3.2 ODRL Core Model

The ODRL Core Model is depicted in Figure 1 and defines the basic structure of every ODRL policy. It defines the entities `Policy`, `Asset`, `Party`, `Permission`, `Duty`, `Prohibition`, `Action`, and `Constraint`. In the following, manifestations of an en-

[2] `www.iptc.org`
[3] `http://openmobilealliance.org`

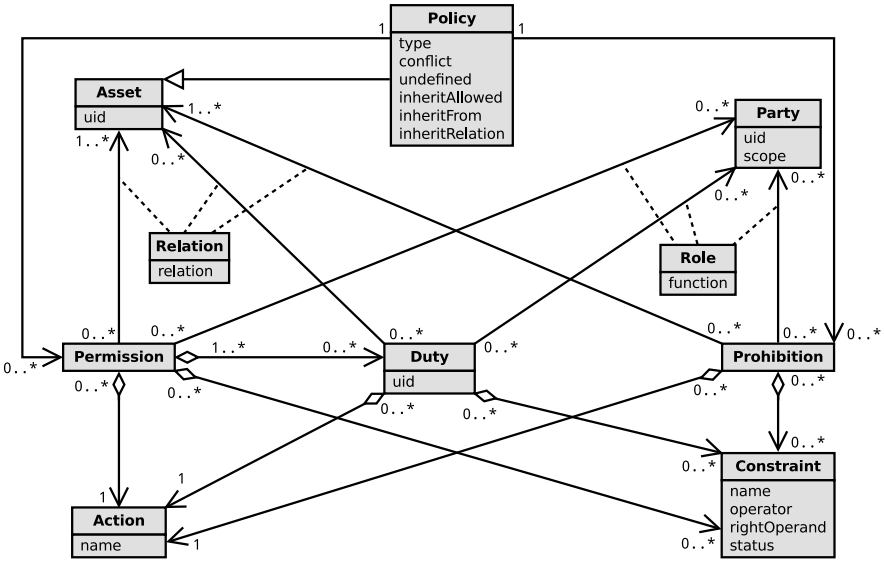

Figure 1: ODRL 2.0 Core Model [IGPK12]

tity are called instances. Most of these entities are further described by attributes. Mandatory attribute values, which must be supported by an ODRL-based DRMS, are defined in the Core Model. For example, each DRMS must support an attribute value to identify parties as rights holders. Optional attribute values are defined in the Common Vocabulary. For instance, a particular application might not require all of the actions defined in the Common Vocabulary. Thus, the actions can be omitted in a corresponding profile. An ODRL policy defines which actions a `Party` may, must, or must not perform on a good. Such actions are represented by the `Action` entity. The semantics of allowed and prohibited actions is expressed through the use of `Permission`s and `Prohibition`s, respectively. Besides actions, permissions and prohibitions are also linked to parties. A collection of permissions and/or prohibitions corresponds to a `Policy`. In the following, the entities of the ODRL Core Model are further described. The Core Model also includes experimental features which are not yet part of the specification. Some of these features are revisited in the further analysis of ODRL in Section 4.

A `Policy` has several attributes and is identified by its unique `uid`. Its type is defined by the attribute `type` which determines the structure and the interpretation of a concrete policy instance. Conflicts arise if a new policy is created by merging two or more policies containing contradicting permissions and prohibitions. A contradiction occurs if one policy allows an actions whereas another policy prohibits this action. Such conflicts are solved by evaluating the attribute `conflict`. The attribute indicates if permissions take precedence over the conflicting prohibitions or vice versa or if the newly created policy is considered invalid. The attribute `undefined` indicates how to handle actions which are unknown to the interpreting DRMS. Actions are unknown if they are either not part of the Common Vocabulary or part of a profile unknown to the system. Inheritance be-

tween policies is possible by including all expressions of an existing policy into the inheriting one. Inheritance relations are expressed by the attributes `inheritAllowed`, `inheritFrom`, and `inheritRelation`. `inheritAllowed` defines if it is possible to inherit from the policy defining the attribute, `inheritFrom` expresses from which other policy to inherit, and `inheritRelation` defines the type of inheritance. As of April, 2013, neither the Core Model nor the Common Vocabulary define any particular values for `inheritRelation`.

An `Asset` is identified by its `uid` and represents either a physical good like a printed book or a digital good like an e-book. A `Relation` associates an asset with a `Permission`, `Prohibition`, or `Duty`. The semantics of this relation is defined by the attribute `relation`. In most cases, an asset is the good on which an action can or cannot be exercised. In other cases, it can be the result of a performed action.

A `Party` is identified by its `uid` and represents a person, an organization, or a group of such. A `Role` expresses the function of a party within a permission, prohibition, or duty. A typical scenario covers two different parties: The `assigner` defines the allowed and prohibited actions and the `assignee` is the party for which these permissions and prohibitions apply. The `scope` indicates how to interpret parties. The value `individual` expresses that a party is an individual person or an individual organization and `group` expresses that the party is a group of such.

`Permission`, `Prohibition`, and `Duty` define the three different types of modality ODRL is able to express. These entities define that an action may, must, or must not be performed. Permissions and prohibitions allow or prohibit actions whereas a duty is a precondition for a permission. All three entities have a similar structure. They define an `Asset`, an `Action`, and a `Party` and can be restricted with `Constraints`.

An `Action` is related to a permission, prohibition, or duty. Depending on the context, the action is allowed, prohibited, or required. The attribute `name` identifies a concrete action. Its particular semantics is defined in the Common Vocabulary or a profile.

A `Constraint` restricts a permission, prohibition, or duty. Constraints are structured like mathematical terms. The attributes `name`, `operator`, and `rightOperand` represent the left side, the operand, and the right side of a mathematical term. For example, a constraint can restrict how often an action can be performed. In this case, `name` has the value `count`, `operator` the value `lteq`, and `rightOperand` the number of possible uses, e. g., 5. This corresponds to the mathematical term $\texttt{count} \leq 5$. The attribute `status` saves the current state of the left side of the mathematical term. For instance, if the action was already performed three times, the value of `status` would be 3. In a proposed update of the Core Model from January 2013 [IGPK13], `dataType` and `unit` are suggested as new attributes for constraints. They define the type and unit of the right operand of a constraint, respectively.

3.3 ODRL Common Vocabulary

The Common Vocabulary is the rights data dictionary of ODRL. It defines a set of default terms which can be used for expressing particular ODRL policies. These terms correspond to possible values an attribute can have. A particular application may use all the terms specified in the Common Vocabulary or only a fragment of them. Furthermore, it may also use additional terms which are not part of the Common Vocabulary. The complete list of the terms used in a particular application corresponds to an ODRL profile [IGPK12]. The Common Vocabulary defines different policy types, action names, types and operators for constraints, scopes and functions for parties, and relations for assets.

4 Analyzing ODRL

This section presents the results of an in-depth analysis of the ODRL Core Model. It is structured as follows: First, a brief overview of previous work discussing potential improvements of ODRL is given. Second, the Core Model is analyzed along its entities. The analysis is based on the related work as well as on own experience of the authors with developing different ODRL applications. Finally, the main findings are summarized and a proposal for a different Core Model is presented.

A general disadvantage of the ODRL Core Model is that it uses modeling concepts that are bound to a later XML encoding. Attributes like `name` or `type` should not be used in a general model. For example, the attribute `type` as described in Section 3.2 is used to define the type of a policy. This attribute is misleading for a particular serialization of ODRL. It suggests that the type of an entity should always be expressed through attributes. However, different serializations of ODRL might use different approaches. Thus, the Core Model should not have such a restriction. The decision to model a manifestation of an entity with attributes or subclasses should be made for a concrete serialization. One example for a serialization which uses subclasses instead of `type` is the proposed RDF/XML serialization of ODRL 2.0 from McRoberts[4]. Generally, a clear separation [HL95] between the model and serialization of the ODRL Core Model should be made.

4.1 Critics of ODRL and Their Critiques

Beck et al. [BBH+13] developed a set of tools to embed and visualize ODRL 2.0 policies in web pages. These tools are based on an OWL ontology[5] of the ODRL 2.0 Core Model. Working with ODRL 2.0 and creating the ontology revealed several weaknesses in the Core Model. The experiences from this project laid the foundation of the analysis of ODRL presented in this paper. Arnab and Hutchinson [AH05] proposed an alternative

[4] `http://www.w3.org/community/odrl/wiki/SemanticWeb`

[5] `http://userpages.uni-koblenz.de/~aggrimm/rox/downloads/ontologies/ODRL2.0.zip`

Core Model for ODRL 2.0. They noted that the Core Model draft of 2005 [IG05] has a lack of expressiveness. Policies cannot be considered as equivalent to legal contracts as they lack some information like date and location of the agreement and are thus not legally binding. Furthermore, the authors criticized that parts of the model are ambivalent, especially those covering `Permission`, `Prohibition`, and `Duty`. These limitations apply to the latest version of the Core Model as well. Finally, the ODRL Initiative defines a list of requirements [GI05] as basis to develop the ODRL 2.0 Core Model. Due to various reasons, not all of these requirements are implemented in the version 2.0 of April 2012 [IGPK12]. The particular findings and suggestions of these works are examined more thoroughly in the following subsections.

4.2 Analyzing the ODRL Policy

This subsection discusses minor flaws of the `Policy` entity like unclear attribute names but also inconsistent modeling and weaknesses in its expressiveness. One of the minor flaws are the names of the attributes `undefined` and `conflict` and their respective values. As described in Section 3.2, `undefined` covers the handling of unknown actions. However, the current name suggests a broader semantics such as handling unknown policy types or unknown constraints. To clearly express that the attribute only applies to actions, it should be renamed to `handleUndefinedActions`. Additional attributes for handling other terms could also be added such as `handleUndefinedPolicyTypes`. In general, a consistent naming scheme within the ODRL specifications for attributes and their values would achieve coherent names. Values could begin with a verb, followed by an optional adjective, and end with an object. The values of `undefined` could then be changed to `supportUndefinedActions`, `ignoreUndefinedActions`, and `invalidatePolicy`. As described in Section 3.2, the attribute `conflict` is evaluated if two conflicting policies are merged. Therefore, a name like `handleConflictsBetweenPolicies` would better convey the intended semantics. The same shortcoming exists for the attribute's values which are `perm`, `prohibit`, or `invalid`. The former two define that either permissions or prohibitions take precedence whereas the latter defines that the new policy shall be made invalid. These values could be changed to `preferPermissions`, `preferProhibitions`, `invalidatePolicy`. More importantly, the attribute `conflict` cannot fulfill its intended semantics. It can only solve conflicts between two or more policies if these policies have the same value of the attribute. Assume that two policies are merged which have different values of the attribute. Since ODRL does not support priorities between policies, both values of `conflict` must be equally evaluated. However, this is not possible because the attribute values contradict with each other. To solve this issue, the handling of conflicts between different policies could be shifted to a shared meta policy. A meta policy can be considered a collection of policies which specifies how to solve conflicts between them. The concept of a meta policy is, e. g., supported by the access control language XACML [Ris13].

The policy type `privacy` expresses ODRL policies describing the usage of personal information [IG12]. This information is identified as the policy's asset. Thus, `privacy` pri-

marily defines this asset's content. Therefore, a privacy policy may also be of type `offer`, `request`, or `agreement`. However, the Core Model only supports one type at a time for each policy. To enhance the expressiveness of ODRL policy types, they could be modeled as an inheritance hierarchy as depicted in Figure 2. Alternatively, the Core Model could be expanded with an asset type which further describes the asset's content. This approach would allow to express that an asset contains personal information and enables a clearer distinction between policy types and asset types. The relations between the `Policy` entity and other entities from the Core Model are modeled inconsistently. A `Permission` is related to an `Asset` via the association class `Relation`.Likewise, the association class `Role` expresses the function of a `Party` associated with a `Permission`. On the other hand, the inheritance relation between two policies is expressed through the attributes `inheritRelation` and `inheritFrom`. However, these attributes are semantically equivalent to an association class which connects the entity `Policy` with itself. Thus, inheritance between two policies should be modeled as such a class as depicted in Figure 3. Policy inheritance can also be utilized to reuse existing policies as templates. An additional system would be useful to create template policies which are reusable for different assets. A template policy is a collection of permissions and prohibitions without concrete assets and/or parties. A particular policy is created from a template policy by adding the omitted parts. Such a template system allows to predefine policies which can be later used for different assets. This can be considered as an alternative to the policy type `set`. Creative Commons follows a similar approach by using predefined policies which can be applied to arbitrary goods.

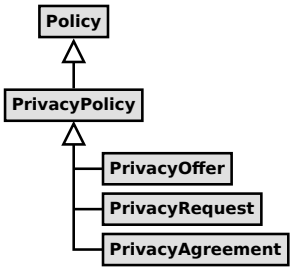

Figure 2: Hierarchy of privacy policies

Arnab and Hutchison [AH05] show a new approach to the expression of contracts using ODRL. They state that the process of contract negotiations and the respective status of policies cannot be expressed. For example, a contract could be a draft or an agreement already accepted by all parties. The authors propose to record such statuses in the policy type or an additional policy attribute. Supporting contract negotiation is also part of the ODRL Initiative's list of requirements [GI05, Req. 1.4]. Arnab and Hutchison also remark that ODRL lacks the possibility to express further metadata of the contract's formation. This is needed in some countries such as South Africa to make the formation legally valid [AH05]. They suggest the usage of further attributes to represent this metadata, e. g., place and time of the contract formation.

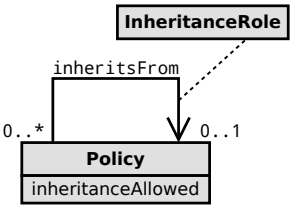

Figure 3: Policy inheritance

Furthermore, Arnab and Hutchison [AH05] point out that ODRL policies do not have a default modality. It is unclear whether those actions not explicitly listed in a policy are permitted or prohibited by default. Without a such default modality, some scenarios such as *Creative Commons* licenses [AALY08] cannot be expressed. According to such

licenses, every action is forbidden which is not explicitly allowed. The general purpose language KAoS [UBJ+03] supports a default modality and distinguishes between *default permit* and *default prohibit*. If each ODRL policy contained its own default modality, an approach like that of KAoS would be sufficient. Using `prohibit` as default modality, permissions express the exceptions from this default setting, i. e., they define those actions permitted by the policy. If the default modality is `permit`, every action is permitted unless it is explicitly prohibited by a prohibition. Consider the example of an Internet video stream which shall only be available after 10 pm to protect minors from harmful content. Depending on the default modality, this could be modeled in two different ways: If `prohibit` is used as default modality, the action `play` is permitted with the constraint "after 10 pm". If all actions are allowed by default, the execution of the action `play` is prohibited before 10 pm. Given that in ODRL the asset is not directly related to the policy but rather to a permission or prohibition, a simple attribute is not sufficient to describe a policy's default modality. It would not be clear for which assets this modality would be applicable. One approach could be a default rule that defines the modality effective for all assets related to that rule within the policy. The general purpose language Rei [KFJ03] supports default modalities for different parties and actions. Alternatively Rei allows rules to be explicitly prioritized.

4.3 Analyzing the ODRL Asset

ODRL identifies assets by their `uid`. A further specification of an asset including additional attributes is not supported. However, some ODRL terms like the action `play` rely on a further description of an asset. The semantics of `play` refers to the rendering of audio assets and video assets. Thus, the action requires an asset to be "playable". In order to explicitly state this requirement, an asset could be associated with an abstract type such as time-based media. Such types can be modeled as subclasses of asset as it is modeled in the general purpose language Rei [KFJ03].

Aggregated assets are not supported by the Core Model, i. e., it is impossible to express that an asset consists of multiple parts. The Common Vocabulary contains several actions that represent the integration or extraction of parts of assets. An example is the action `aggregate` which corresponds to creating a new asset as a composite collection of several other assets [IG12]. In the ODRL 2.0 Core Model the connection between the former asset and the result of such an operation is only expressed through their relation to the same permission or prohibition. There is no relation between the parts and the whole. A possible solution could be a `hasPart` relation between the involved assets which has also been a requirement for ODRL 2.0 [GI05, Req. 1.2].

4.4 Analyzing the ODRL Party

As described in Section 3.2, a `Party` can represent a person, an organization, or a group of such. To indicate whether the party is a group or an individual, the `scope` attribute is used. However, the Core Model and the Common Vocabulary are ambiguous at this point. The Common Vocabulary describes `scope` as attribute of the entity `Role`. On the contrary, the Core Model defines it as part of the entity `Party` as well as in the association class `Role`. Figure 1 presents a visualization of the Core Model where the scope is an attribute of `Party`. In this case, it is only possible to associate the same party with different instances of `Permission`, `Prohibition`, or `Duty` in which its `scope` is alike. To express a different `scope` of the same party, multiple instances of this party must be defined. Apparently, this is a modeling error. The same party can take part in different policies or have different roles even in the same policy with variable scopes. For example, the *University of Koblenz-Landau* may pay 500,00 Euro in order to allow each of its members to use a certain software. This is expressed by defining the university as the assignee of a permission covering the `use` action. Since the payment is carried out only once, the university's scope within the duty to pay the fee is set to `individual`. On the other hand, the usage of the software affects all members of the university. In this case, the scope should be set to `group`. In this example, it would be necessary to define two different instances of `Party` which both represent the University of Koblenz-Landau. The attribute `scope` should therefore better be defined in the association class `Role`. This supports different `scope`s for the same party without the need for multiple instances. In the example, the scope of the `assignee` within the `Permission` is set to `group` and within the `Duty` to `individual`.

4.5 Analyzing the ODRL Permission, Prohibition, and Duty

`Permission`, `Duty`, and `Prohibition` have a similar structure, but are modeled as separate entities. All three entities describe under which constraints what actions a party may, must, or must not perform on an asset. The sole difference is that a duty can only be assigned to a permission whereas permissions and prohibitions can be directly assigned to a policy. To increase the clarity of the Core Model, it would be more elegant to model all three classes as specializations of a common superclass like `Rule` as depicted in Figure 4 and already proposed in the experimental features of the Core Model. Such a superclass also allows for defining additional rule types in a profile [IGPK12]. For example, the general purpose language Rei [KFJ03] defines the rule type *dispensation* for suspending an existing duty. Similarly, KAoS [UBJ+03] defines *negative duties*, i. e. actions which are not required to be performed. In Figure 4 `Duty` is modeled as subclass of `Permission` since each required action implies a permitted one. The relation between `Permission` and `Duty` would not

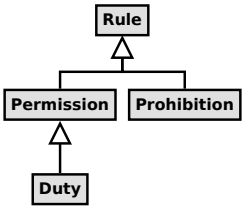

Figure 4: Rule hierarchy

be influenced by this modification. A policy would then contain at least one `Rule` instead of a `Permission` or `Prohibition` as shown in Figure 5 at the end of this section.

Arnab and Hutchinson [AH05] identify the impossibility of describing that an asset can be used without any limitation if a user pays a certain fee. Such a scenario could be realized through a policy's default modality (see Section 4.2) and the previously proposed superclass `Rule`. The use of `Rule` would also allow a policy to contain only a duty. A single duty requiring a certain fee combined with the default modality `permit` can be used, e. g., to express that an asset can be used without any limitations after paying the fee.

Furthermore, it should be reexamined whether it would be useful to allow a connection between two duties. This could be used to express that a certain requirement has to be fulfilled before another one. Similarly, it is currently not possible to define a duty as post-condition of a permission. According to the Core Model, a duty is always a precondition. The ODRL Reqirements [GI05, Req. 1.10] give an example of a payment that must be carried out within four weeks after the receiving the ordered good. Following this requirement, it should also be possible to model a duty as a postcondition of a permission.

4.6 Analyzing the ODRL Action

The Common Vocabulary lists various actions to be used in ODRL policies. However, the semantics of some actions is unclear due to an imprecise definition in the vocabulary. According to the Common Vocabulary, the actions `copy` and `reproduce`, `derive` and `modify`, as well as `display` and `present` share the same semantics. However, the vocabulary does not clearly define what this means. It is unclear whether two actions are completely equivalent to each other or if there is a inheritance relation between them. For example, `present` and `display` as well as `present` and `execute` share the same semantics. However, these actions are clearly not equivalent to each other since they have a different semantics. `display` covers the rendering of visual assets whereas `execute` is used for describing the usage of software assets [IG12]. In order to avoid such a semantic confusion and to create unambiguous actions, Barlas [Bar06] suggest that actions must be defined precisely as possible. Furthermore, Kasten and Scherp [KS12] point out that abstract and therefore ambiguous actions are typical for RELs and that this complicates their enforcement by a DRMS. The authors give `anonymize` as an example. Although this action covers the anonymization of the asset, it is not define how much and which personal data must actually be removed. In order to create unambiguous policies, each action semantics should only be defined using one single action and equivalent actions should be removed. To keep acceptance and usability for communities which have their own vocabulary, these members could use profiles to include special actions.

Value chains of digital goods represent the path of a good from its producer to a consumer. Therefore, usage rights granted to a party can be passed on by a vendor. For example, a producer grants usage rights to a distributor who in turn may only grant certain rights to consumers. This can be represented by attaching further policies to a policy. The Common Vocabulary suggests the action `nextPolicy` for this case. The producer would add the

policy that she suggests to be used between vendor and customer as an asset to a duty of the action `nextPolicy`. This approach uses an action to express the role of an asset. The function of an asset should better be modeled through a role entity which is also depicted in Figure 5. The action `attachPolicy` is another example that should be replaced by a role. This approach refines the effort already made by the ODRL 2.0 entity `relation`.

The actions listed in the Common Vocabulary can generally be distinguished between *atomic actions* and *complex actions*. Complex actions such as `display`, `execute`, and `play` are such actions which can be expressed by combining one or more other actions and by using constraints. Many complex actions like `adhocShare` or `extractChar` are primarily designed for special use cases and can thus hardly be used in other scenarios. Actions like these lead to a large list of actions in the Common Vocabulary which makes it difficult to distinguish them from the more important ones. It would be more effective to define a basic vocabulary with a few atomic actions like `present`, `delete`, `give` and so on. Complex actions could be built on top of such atomic actions. `display`, for example, is used for visual assets and is a special form of `present`. Therefore, it can be expressed by using the atomic action `present` and an asset-constraint to qualify assets like audio or video. `execute` and `play` could also be described that way. `extractChar` can be expressed by the atomic action `extract` and a constraint which describes the element to be extracted. The definition of a basic vocabulary eases the process of comparing complex actions built on top of them. Even newly defined actions can be easily compared to already defined actions as long as they are based on the same basic terms. Using the same basic vocabulary for different ODRL policies also increases the interoperability, compatibility, and reusability between them and their respective applications.

4.7 Analyzing the ODRL Constraint

In ODRL 2.0 constraints express mathematical terms by using the attributes `name`, `operator`, and `rightOperand`. Constraints are associated with a permission, prohibition, or duty and can limit different entities like assets or actions. An example for a constraint is `play < 5`, which means that an asset can be executed up to four times. Although a mathematical foundation seems appropriate for numerical constraints, it does not fit for all constraints defined in the Common Vocabulary. For certain constraints like `industry`, `event`, or `product` this mathematical approach seems not intuitive. To get a constraint model which fits for all constraints defined in the Common Vocabulary and to simplify the usage of constraints, a distinction between different types of constraints should be adopted. This approach is also used by other policy languages such as Ponder [DDLS01] which distinguishes between time constraints, action/event parameters and subject/target state. The type of constraint defines the entity it restraints. Therefore, the constraint model should provide different types like `quantity-constraints`, `time-constraints` as well as `asset-constraints` for various use cases. A quantity-constraint like `count` or `percentage` defines how often an action can be exercised on an asset and limits thereby permissions and prohibitions. On the other hand, asset-constraints such as `fileFormat` restrict the type of an asset. These different constraint types are defined in a basic set of

common constraints. In addition, combining one or more constraints of different types related to the same permission can be used for defining more extensive constraints. A party who can only watch a movie up to three times between 8 pm and 11 pm would be an example for such a combined constraint. In this case, the constraint is composed by a quantity constraint and a time constraint.

4.8 Summary of the Analysis

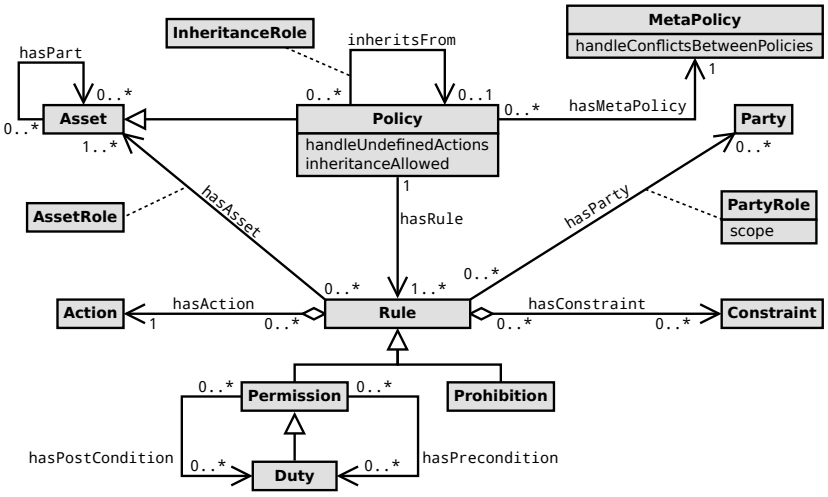

Figure 5: Adapted ODRL 2.0 Core Model

Figure 5 presents a new version for an ODRL 2.0 Core Model incorporating the main suggestions from above. However, the model does not solve all of the identified shortcomings. Thus, further improvements are necessary. The main entity of the new Core Model proposal is the new class `Rule` introduced as superclass of `Permission`, `Duty`, and `Prohibition`. It links assets, parties, and constraints to an action. Aggregated assets can be expressed by the relation `hasPart`. Specific instances of `Action`, `Policy`, `Constraint` and their attributes are modeled as subclasses of the corresponding entity (not shown in Figure 5). To improve the readability of the model, the association class `Role` has been renamed to `PartyRole` and `Relation` has been renamed to `AssetRole`. A `MetaPolicy` is used to express conflict management between policies by evaluating the attribute `handleConflictBetweenPolicies`. Therefore, the `conflict` attribute of `Policy` was removed. Also `type` was replaced by corresponding subclasses of `Policy`, which are not shown in Figure 5. `undefined` has been renamed to `handleUndefinedActions`. `InheritanceRole` defines the inheritance relation between policies. Furthermore, aspects such as the default modality as well as attributes of `Constraint` are not modeled in Figure 5 yet.

5 Conclusion and Future Work

Based on own experience with ODRL [BBH+13] and related work [AH05, Bar06], this paper discussed several weaknesses of the current ODRL specification and outlined possible improvements. The identified weaknesses include inconsistent and misleading terms defined in the Common Vocabulary and attributes given in the Core Model, incomplete concepts such as the solution of conflicting rules, and even modeling errors such as the function of a party within a rule. Future work includes a further development of the suggested Core Model as a semantic web ontology which provides more expressiveness and background information than an XML schema. As of April, 2013, the XML encoding is the only available serialization format for ODRL policies. First attempts at expressing the ODRL Common Vocabulary and the Core Model as RDF are already done by McRoberts[6]. Furthermore, the Common Vocabulary should be revised and separated into different models for the entities `Policy`, `Party`, `Asset` and `Constraint`. The Policy Model and the Party Model define the policy types and party types, respectively. The Asset Model covers the newly introduced entities for describing assets. The Constraint Model consists of all constraints, which are distinguished between the entities they apply to. The proposed concepts aim at clarifying the meaning of ODRL and its policies by separating the REL and RDD into smaller models which are easier to manage and comprehend.

References

[AALY08] Hal Abelson, Ben Adida, Mike Linksvayer, and Nathan Yergler. ccREL: The Creative Commons Rights Expression Language, 2008. `http://www.w3.org/Submission/ccREL/` (accessed 17/04/13).

[AH05] Alapan Arnab and Andrew Hutchison. A New Approach to ODRL V2.0. 2005.

[Bar06] Chris Barlas. Digital Rights Expression Languages (DRELs). Technical report, JISC Technology and Standards Watch, 2006.

[BBH+13] Nicolas Beck, Stefan Becker, Marcel Heinz, Niklas Hessel, Alexander Klein, Katharina Naujokat, Alexander Philipp, Matthias Querbach, Torsten Schäfer, Arne Fritjof Schmeiser, Irina Schmidt, Alexander Steinmetz, and Martin Westermayer. ROX 2 RDFa-based ODRL licenses in XHTML web pages. Technical report, Universität Koblenz-Landau, 2013.

[Cun04] Morgan V Cundiff. An introduction to the metadata encoding and transmission standard (METS). *Library Hi Tech*, 22(1):52–64, 2004.

[DDLS01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder policy specification language. In *POLICY 2012*, pages 18–38. Springer, 2001.

[GI05] Susanne Guth and Renato Iannella. Open Digital Rights Language (ODRL) Version 2 Requirements, 2005. `http://odrl.net/2.0/v2req.html` (accessed 24/03/13).

[Gut03] Susanne Guth. Rights expression languages. In *Digital Rights Management*, pages 101–112. Springer, 2003.

[6] `http://www.w3.org/community/odrl/wiki/SemanticWeb`

[HL95] Walter L. Hürsch and Cristina Videira Lopes. Separation of Concerns. Technical report, 1995.

[Ian12] Renato Iannella. ODRL V2.0 XML Encoding, 2012. http://www.w3.org/community/odrl/two/xml/ (accessed: 25/03/13).

[IG05] Renato Iannella and Susanne Guth. ODRL V2.0 - Model Semantics: Working Draft: 16th May 2005, 2005. http://odrl.net/2.0/WD-ODRL-Model-20050516.html (accessed 04/12/13).

[IG12] Renato Iannella and Susanne Guth. ODRL V2.0 Common Vocabulary, 2012. http://www.w3.org/community/odrl/two/vocab/ (accessed: 25/03/12).

[IGPK12] Renato Iannella, Susanne Guth, Daniel Pähler, and Andreas Kasten. ODRL V2.0 Core Model, 2012. http://www.w3.org/community/odrl/two/model/ (accessed: 25/03/13).

[IGPK13] Renato Iannella, Susanne Guth, Daniel Pähler, and Andreas Kasten. ODRL V2.0 Core Model – Constraint Draft Changes, 2013. http://www.w3.org/community/odrl/work/2-0-core-model-constraint-draft-changes/ (accessed: 20/04/13).

[Int04] International Organization for Standardization. Information technology – Multimedia framework (MPEG-21) – Part 5: Rights Expression Language, 2004.

[IPT12] IPTC. RightsML V1.0, 2012. http://www.iptc.org/std-dev/RightsML/1.0EP/specification/RightsML_1.0EP1-spec_2.pdf (accessed: 25/03/13).

[KFJ03] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *POLICY 2003*, pages 63–74. IEEE, 2003.

[KS12] Andreas Kasten and Ansgar Scherp. A pattern system for information flow control on the Internet. 2012. under work, publication prepared.

[OMA08] OMA. DRM Rights Expression Language, 2008. http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DRM&file=V2_1-20081106-A/OMA-TS-DRM_REL-V2_1-20081014-A.pdf (accessed: 02/04/13).

[Ris13] Erik Rissanen. eXtensible Access Control Markup Language (XACML) 3.0, 2013. OASIS Standard.

[RTM02] Bill Rosenblatt, Bill Trippe, and Stephen Mooney. *Digital Rights Management*. M&T Books, 2002.

[UBJ+03] Andrzej Uszok, Jeffrey Bradshaw, Renia Jeffers, Niranjan Suri, Patrick Hayes, Maggie Breedy, Larry Bunch, Matt Johnson, Shriniwas Kulkarni, and James Lott. KAoS policy and domain services. In *POLICY 2003*, pages 93–96. IEEE, 2003.

[Öbe13] Jonas Öberg. ODRL 2 JSON Encoding. W3C ODRL Community Group, 2013. http://www.w3.org/community/odrl/two/xml/ (accessed: 11/04/13).