A Proposal for the Evolution of the ODRL Information Model

Susanne Guth and Mark Strembeck
Department of Information Systems, New Media Lab
Vienna University of Economics and BA, Austria
{firstname.lastname}@wu-wien.ac.at

Abstract

In this paper, we discuss the information model of ODRL 1.1 with respect to the definition of rights and duties for contract parties. We identify a number of shortcomings, and propose an evolutionary advancement of the ODRL. In particular, we present a modified information model and corresponding XML schemas.

1 Introduction

A contract typically represents an agreement of two or more parties. The contract specifies rights and obligations of the involved stakeholders with respect to the subject matter of the respective contract. Digital contracts are most often defined via special purpose XML-based *rights expression languages* (REL), such as ODRL [13], XrML [6], or MPEG 21 REL [7].

On the one hand, a language for the definition of digital contracts should enable the automated processing of digital contracts via software programs. On the other hand, the resulting contracts should also be human-readable and also valid in law. In order to fulfill these requirements, languages for the definition of digital contracts must provide a straightforward grammar and a fixed (but extensible) and unambiguous vocabulary. Moreover, they should simultaneously be flexible enough to express a wide variety of different business cases.

In this paper, we discuss the information model of ODRL 1.1 with respect to the definition of rights and duties for different contract parties. We identify a number of shortcomings, and propose an evolutionary advancement of the ODRL. In particular, we present a modified information model and the corresponding XML schemas.

The remainder of this paper is structured as follows. In Section 2 we give an overview of ODRL 1.1 and identify a number of drawbacks in that specification. Section 3 then introduces a proposal for a future version of the ODRL information model. Subsequently, Section 4 briefly discusses our changes to the ODRL XML schemas. Moreover, we give an example to show how our approach helps to remove different disadvantages of ODRL 1.1. Section 5 provides an outline of (technical) issues which must be considered when mapping permissions and duties defined in digital contracts to enforceable policy rules in concrete software systems. Section 6 gives a brief overview of related work and Section 7 concludes the paper.

2 Definition of Digital Contracts with ODRL 1.1

In this section we introduce the language constructs offered by ODRL 1.1 to define the rights and duties of different contract parties. Section 2.1 presents the respective elements of the ODRL 1.1 information model.

Section 2.2 discusses drawbacks of the current ODRL information model with respect to the expressiveness, the understandability/comprehensibility for human users, and the automated processing of ODRL-based contracts via different software services, e.g. access control services (see also [11]).

2.1 Subset of the ODRL Information Model

Figure 1 shows a subset of the ODRL 1.1 information model [13]. *Party*, *Asset*, and *Permission* are the core elements of ODRL (these elements are subelements of the ODRL *Rights* element, the ODRL *Offer* element, or the ODRL *Agreement* element which are not shown in the figure). The three core elements allow for the definition of simple rights expressions, e.g. "*Ms. Guth* (party) has the right to *play* (permission) the movie *Run Lola Run* (asset)". Note that in ODRL a permission is an operation, such as *play, print,* or *copy*, whereas in the area of access control a "permission" is an $\langle operation, object \rangle$ pair, such as $\langle play, movie \rangle$ (see also [5, 8, 9]). Sometimes permissions need to be constrained, e.g. to a specific time-interval or by the maximum number of uses (see e.g. [1, 14, 16]). ODRL offers three different language elements to define constraints:

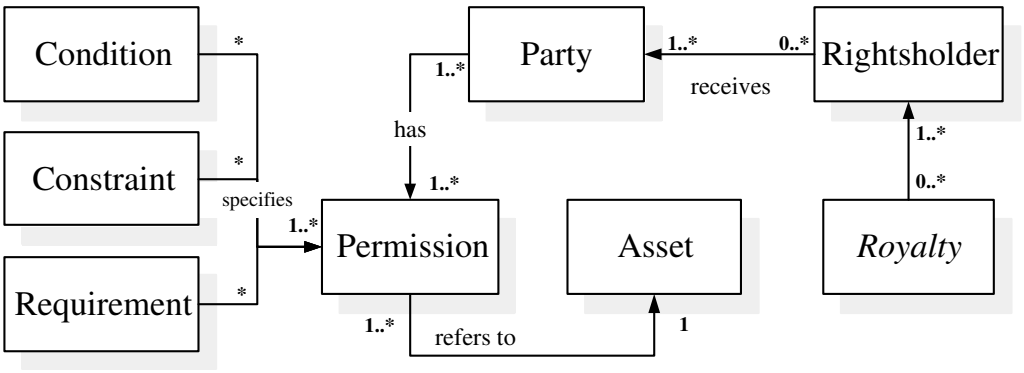

Figure 1: Excerpt of the ODRL 1.1 information model

- A *Requirement* element defines a specific type of precondition (for permission assignments). In particular, an ODRL requirement states that the permission it is related to may only be granted to the respective beneficiary if the corresponding requirement is fulfilled. In ODRL, monetary payments are the most common type of such “requirements”.
- The *Constraint* element of ODRL is intended to narrow ODRL permissions. For example, a “play” permission can be constrained to a maximum of five usages via a *count* constraint. ODRL provides a number of (predefined) constraints: user-, device- bound-, temporal-, aspect-, target-, and rights constraints (for details see [13]).
- An ODRL *Condition*, in essence, define constraints which restrict the validity of a permission. Once a condition is fulfilled, the condition renders the respective permission as no longer valid.

In contracts the party element usually occurs twice, once for the beneficiary (also: consumer or buyer) and once for the so called rightsholder (or seller). In ODRL a rightsholder is identified via a *Rightsholder* element nested in a party element. ODRL party elements that do not include a rightsholder element per definition

“automatically” reference a beneficiary. Additional information related to rightsholders can be specified via a *Royalty* element (see Figure 1). Concrete ODRL royalty elements are *Fixed Amount* and *Percentage* (for details see [13]). These constructs either define a fixed amount or a percentage of the revenues resulting from the corresponding business transaction, and can be assigned to rightsholder parties of a digital contract.

The elements described above are used to express rights and duties in ODRL-based digital contracts. The ODRL example below expresses that: The two parties “Ms. Guth” and “Mr. Strembeck” have reached an agreement on the purchase of the right "display" to an asset identified as “ODRL workshop proceedings”. Mr. Strembeck is the consumer and Ms. Guth is the rightsholder of the workshop proceedings. The “display” right costs € 5.00 and is associated with a time constraint that expires on January 1st, 2011. 100 percent of the agreed upon royalties go to Ms. Guth.

```
<?xml version="1.0" encoding="UTF-8" ?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
             xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
  <o-ex:agreement>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>x500:c=AT;o=Registry;cn=sguth</o-dd:uid>
        <o-dd:name>Dr. Susanne Guth</o-dd:name>
      </o-ex:context>
      <o-ex:rightsholder>
         <o-dd:percentage>100</o-dd:percentage>
      </o-ex:rightsholder>
    </o-ex:party>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>x500:c=AT;o=Registry;cn=mstrembe</o-dd:uid>
        <o-dd:name>Dr. Mark Strembeck</o-dd:name>
      </o-ex:context>
    </o-ex:party>
    <o-ex:asset>
      <o-ex:context>
        <o-dd:uid>urn:wu-wien.ac.at#proc01</o-dd:uid>
        <o-dd:name>ODRL Intl. Workshop '04 Proceedings</o-dd:name>
      </o-ex:context>
    </o-ex:asset>
    <o-ex:permission>
     <o-dd:display>
        <o-ex:constraint>
          <o-dd:datetime>
                 <o-dd:end>2010-12-31</o-dd:end>
          </o-dd:datetime>
        </o-ex:constraint>
        <o-ex:requirement>
           <o-dd:peruse>
             <o-dd:payment>
               <o-dd:amount o-dd:currency="EUR">5.00</o-dd:amount>
             </o-dd:payment>
           </o-dd:peruse>
        </o-ex:requirement>
     </o-dd:display>
    </o-ex:permission>
  </o-ex:agreement>
</o-ex:rights>
```

2.2 Drawbacks of ODRL 1.1

We now discuss drawbacks that result from the ODRL 1.1 information model. The discussion focuses on the expressiveness, the understandability/comprehensibility for human users, and the automated processing of ODRL-based contracts via different software services.

1. **Expression of rights and duties.** Generally (and non-technically) speaking, contracts specify the rights and duties of contract parties. In the example from Section 2.1, a customer receives the *right* to display some resource after fulfilling his *duty* of paying a certain amount of money. The rights and duties of the rightsholder, however, are less explicit. With respect to the same example, the rightsholder receives a certain amount of money in return for the usage right of her resource. Here, the respective royalty element ("percentage") may also be interpreted as a right of the rightsholder (e.g. "the right to debit the consumers account"). But does the rightholder have duties, too? A human reader of the example contract shown in Section 2.1 may use his knowledge on the nature of contracts in general to interpret the contract and to identify the duty of the rightsholder that, in return to receiving a certain amount of money, she has to make the asset (the digital good) available to the consumer. Nevertheless, such implicit information can, in the general case, not easily be derived via an automated processing of ODRL-based contracts. This results from the fact that ODRL does not provide language elements that can explicitly express (arbitrary) duties of contract parties (aside from monetary payments, as mentioned above).

2. **Distinction between the rightsholder and consumer parties.** Let us assume that a certain agreement shall include duties of two contract parties, as it is common in barters for instance (a barter is an agreement on the exchange of one asset against another asset, in contrast to a monetary payment). For example, an Austrian university department offers its learning resources (presentations, papers, etc.) to a German university department. In return, the German department agrees to provide its learning resources to the Austrian department. In this example, two parties exchange usage rights for certain digital goods and no monetary payment is required of either department. Thus, both parties are rightsholder *and* beneficiary at the same time. If, however, both parties in an ODRL-based contract include the rightsholder element, the ODRL expression of granting mutual access rights gets ambiguous. Therefore, the simple example already indicates that ODRL 1.1. is not well-suited to model situations where different contract parties "act" in multiple contract roles. In particular, a rightsholder party cannot be treated as a consumer at the same time.

3. **The expression of constraints on requirements.** A common type of expression used in contracts is the following: "... the payment has to be made within four weeks after receipt of the shipment." With respect to ODRL 1.1 this expression is a requirement (the payment) that is narrowed by a constraint (within four weeks). However, ODRL 1.1 does not allow to constrain requirements.

4. **The term "permission".** In ODRL *permissions* refer to certain *operations*, for example "play" or "print". However, in the area of system security and access control a permission refers to an $\langle operation, object \rangle$ pair as $\langle play, movie \rangle$, for example. Moreover, as in the area of access control, ODRL provides *constraints* that can be used to define specific "side-conditions" on *permissions*. Nevertheless, due to the above mentioned difference the approaches significantly differ, since in ODRL, constraints are associated with operations rather than $\langle operation, object \rangle$ pairs (see also Figure 2).

   However, if we constrain operations rather than $\langle operation, object \rangle$ pairs, we need additional means to specify the asset(s) that a certain constraint applies to (at least if it should not apply to all possible assets

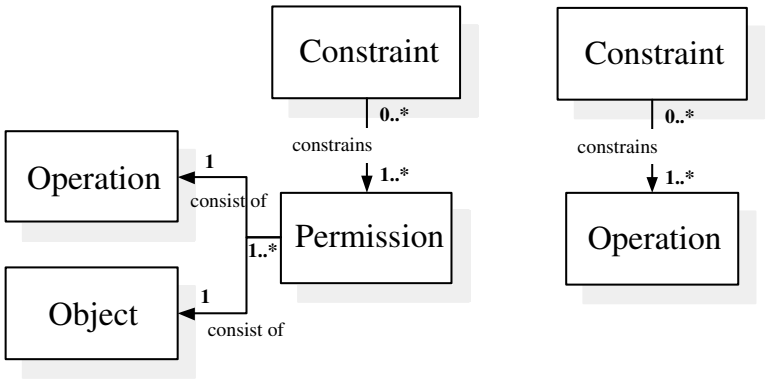

Figure 2: Constraints on Permissions vs. Constraints on Operations

the corresponding operation could be used on). In other words, if more than one asset (e.g. three PDF documents) is referenced in a contract, the operation (e.g. view or print) is constrained for each of these assets. This, however, causes obvious problems if a constraint should refer to one specific asset only (e.g. a specific PDF document, and thereby to a specific $\langle operation, object \rangle$ pair as $\langle print, lecturenotes \rangle$), while the same operation (e.g. print) should not be constraint for the other assets referenced in the contract. Therefore, in the general case, it is more flexible (and more expressive) to enable the definition of constraints on $\langle operation, object \rangle$ pairs than on operations only. In particular, this allows to unambiguously define constraints that clearly refer to a specific $\langle operation, object \rangle$ pair.

5. **Different types of "conditions" in ODRL.** ODRL version 1.1. distinguishes between constraints, requirements and conditions (see Section 2.1) which can be assigned to ODRL permissions (i.e. operations). As an example, let us consider an ODRL payment requirement, an ODRL time condition, and an ODRL count constraint (see also [13]). A permission that has these elements assigned to it, may then be granted *iff* the payment was settled, *iff* the time restriction is not met (i.e. the right is not yet expired), and *iff* the count limit is not exceeded. Each of these clauses defines a certain "side-condition" for a respective permission. With respect to our discussion points 1 and 3 mentioned above, we believe that it would be more reasonable to distinguish between ODRL conditions/constraints on the one hand and ODRL requirements (which in essence define duties) on the other. ODRL conditions and ODRL constraints directly relate to a permission (e.g. do not play *after 12/31/05*; or play at most *five times*) whereas requirements are more of a precondition that has to be fulfilled before a right may actually be assigned to the corresponding beneficiary. Therefore, we argue that ODRL requirements are classical duties, whereas ODRL conditions and ODRL constraints should be seen as constraints on $\langle operation, object \rangle$ pairs.

3 Proposal for a Future ODRL Information Model

In this section, we propose an information model that can be considered in a future version of ODRL. In particular, we introduce *duty* as a new element that can be assigned to contract parties and we use *constraint* as the sole element to define "side-conditions" on duties or permissions (which are defined as $\langle operation, object \rangle$ pairs rather than operations in ODRL 1.1, see below). The rightsholder element and the related royalties have been removed. Figure 3 shows the a proposal for an information model which could serve as an replacement

for the model depicted in Figure 1. The elements of the proposed information model and their relations are described in this section. In subsequent paragraphes, contracts that are compliant to the proposed ODRL information model are referred to as *future* (ODRL) contracts.

Each future ODRL contract contains two or more contract parties. A *contract party* may be either a physical person (an individual), a legal person (an organization), or an abstract type of party (a role). Typical types of parties in a contract are buyer, seller, rightsholder, or beneficiary. A *permission* essentially consists of an $\langle operation, object \rangle$ pair and grants the right to perform the corresponding operation (e.g. play) on the respective object (e.g. a particular MPEG video file). Here, the object represents a reference to a certain asset, and an *asset* is defined as a good or service, be it digital or physical. Each permission in an future ODRL contract is assigned to at least one contract party, and each contract party may possess a number of permissions.

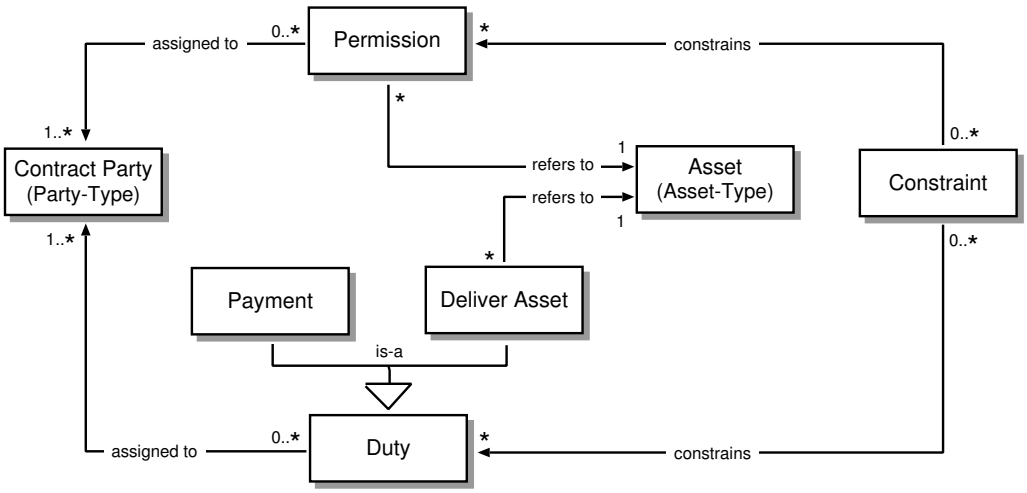

Figure 3: Proposal of a Future ODRL Information Model

A *duty* defines a reward for a certain permission. This reward may be, for example, a (monetary) *payment* or the duty to *deliver an asset*. Duties are optional elements in an electronic contract. Each duty is assigned to at least one contract party which is thereby (i.e. via the contract) obliged to fulfill this duty. On an abstract (business process) level, a duty can be seen as the "counterpart" of a permission, which means that a party receives one or more permissions in exchange for the fulfillment of one or more corresponding duties.

Permissions and duties may be associated with constraints. Here, a *constraint* is a predicate and defines a (side-)condition on the respective permission or duty. A typical example of a constraint may be a time constraint which checks if the current date (todays date) is previous to "December 31st, 2005" (see also [16]). Such a constraint could be associated with:

- a permission to define that this particular permission is valid while the corresponding constraint is fulfilled (i.e. while the respective predicate returns true), or
- a duty to define that the corresponding duty must be fulfilled before the respective constraint "expires" (i.e. before the respective predicate returns false). For example, such a constraint can indicate, that a certain duty, e.g. a payment, has to be made before "December 31st, 2005".

The information model proposed in this section thereby eliminates the drawbacks described in Section 2.2.

4 Proposed Changes for the ODRL Expression Language XML Schema

We now give an overview of proposed modifications of the ODRL XML schemas. These modifications should allow for the definition of ODRL-based digital contracts which adhere to the information model suggested in Section 3. Corresponding XML schemas can be found in Appendix A and B of this paper. These schemas include the following modifications:

- A new `duty` element. This element has the type *dutyType* and includes *dutyElements*. Concrete dutyElements, such as `prePayment`, `perusePayment`, or `deliverAsset` are defined in the corresponding ODRL data dictionary schema. The data dictionary currently does not include other duties that are available as requirements in ODRL 1.1, such as tracked, attribution, accept, and register (see also [13]). Those element have to be newly defined accordingly.

- The `permissionType` element has been revised and now includes the mandatory elements `operation` and `object` as well as the element `beneficiary` and the attribute `grantor`. The `hasConstraint` subelement is used to a link to constraint elements that are associated with the corresponding `permission` element.

- Conditions and requirements, as defined in ODRL 1.1, have been removed and are not part of our proposed ODRL schemas. Requirements are now represented as duties. All conditions that narrow permissions are modeled as constraints.

- The `constraintType` has been heavily modified and now includes `operand` and `operator` elements. Moreover, a corresponding `operatorType` and an `operandType` were added to the expression language. The type of an operator is determined via a corresponding `type` attribute that defines if an operator is a prefix, a postfix or an infix operator. Furthermore, the `operatorType` includes a `valence` attribute, defining if the operator is an unary, binary, ternary or n-ary operator. The `operandType` includes the `position` attribute which indicates whether the the operand is located e.g. to the left or to the right of a binary infix operator. An example of a constraint using the binary infix operator `<=` could be `date <= 2005-01-01`.

- The rightsholder element of ODRL 1.1 has been removed. Rights and duties of a rightsholder can now be represented by the permission and duty elements.

The listing below depicts an XML instance of the proposed future ODRL XML schemas and shows how the identified shortcomings of ODRL 1.1 can be removed. With respect to its content, the example resembles the contract described in Section 2.1. To demonstrate the increased expressiveness the contract includes additional statements:

- The listing includes rights (or permissions) and duties. In particular, Mark Strembeck receives the permissions `display` and `print`, and in return, he accepts the duty of a prepayment over the amount of €5.00. For each permission a grantor and a beneficiary can be specified. In the same way a `bearer` can be specified for each duty. Therefore, permissions and duties can be assigned to each contract party, be it a seller or a buyer. This issue addresses drawback 1 discussed in Section 2.2.

- The example no longer distinguishes between a rightsholder party or a consumer party. This permits that each contract party may receive permissions, grant permissions, and/or be associated with duties. This issue addresses drawback 2 discussed in Section 2.2
- The new *duty* element helps eliminating drawback 3 explained in Section 2.2. In particular, the listing shows that constraints can be related to duties. In the example, a time constraint is related to the `prePayment-Duty`, expressing that the payment has to be settled until April 4th, 2004.
- The example also shows the new shape of the ppermission element that now includes operations and objects. Constraints are now related to the $\langle operation, object \rangle$ pair and no longer to the operation only. This method permits a clear translation from ODRL expressions to access control information, and thus addresses drawback 4, as discussed in Section 2.2. Additionally, the listing shows that each constraint (e.g. `constraint01`) can be assigned to various permissions.
- The rights expression in the listing below defines various constraints. ODRL 1.1 condition elements now are also expressed as constraints. This means that former elements ODRL *Constraint* and ODRL *Condition* are now expressed with one newly shaped element called `constraint`. For example, a condition in ODRL 1.1 states: "If the country where a usage right should be claimed is Australia, then the permission must not be granted." Constraint03 in the listing shows how this former *ODRL Condition* can be formulated as constraint. This change in the ODRL language addresses drawback 5. Addtionally, the constraint element now includes the elements `operator`, `operand`, and constraint `type`. Thus, an arbitrary number of constraints can be formulated without changing the data dictionary. Thereby we are also able to check the fulfillment of duties via constraints (see `constraint02`).

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex20:rights xmlns:o-ex20="http://odrl.net/2.0/ODRL-EX"
               xmlns:o-dd20="http://odrl.net/2.0/ODRL-DD">
   <o-ex20:agreement>

      <o-ex20:party partyID="party01">
         <o-ex20:context>
            <o-dd20:uid>x500:c=AT;o=Registry;cn=sguth</o-dd20:uid>
            <o-dd20:name>Dr. Susanne Guth</o-dd20:name>
         </o-ex20:context>
      </o-ex20:party>
      <o-ex20:party partyID="party02">
         <o-ex20:context>
            <o-dd20:uid>x500:c=AT;o=Registry;cn=mstrembe</o-dd20:uid>
            <o-dd20:name>Dr. Mark Strembeck</o-dd20:name>
         </o-ex20:context>
      </o-ex20:party>

      <o-ex20:asset assetID="asset01">
         <o-ex20:context>
            <o-dd20:uid>urn:wu-wien.ac.at#proc01</o-dd20:uid>
            <o-dd20:name>ODRL Intl. Workshop '04 Proceedings</o-dd20:name>
         </o-ex20:context>
      </o-ex20:asset>

      <o-ex20:permission grantor="party01">
         <o-ex20:operation>display</o-ex20:operation>
         <o-ex20:object>asset01</o-ex20:object>
         <o-ex20:beneficiary id="party02"/>
```

```
         <o-ex20:hasConstraint id="constraint01"/>
         <o-ex20:hasConstraint id="constraint02"/>
         <o-ex20:hasConstraint id="constraint03"/>
      </o-ex20:permission>
      <o-ex20:permission grantor="party01">
         <o-ex20:operation>print</o-ex20:operation>
         <o-ex20:object>asset01</o-ex20:object>
         <o-ex20:beneficiary id="party02"/>
         <o-ex20:hasConstraint id="constraint01"/>
      </o-ex20:permission>

      <o-ex20:constraint constraintID="constraint01">
         <o-ex20:type>datetime</o-ex20:type>
         <o-ex20:operator type="infix" valency="Binary"> LessThan </o-ex20:operator>
         <o-ex20:operand position="left">today</o-ex20:operand>
         <o-ex20:operand position="right">2011-01-01</o-ex20:operand>
      </o-ex20:constraint>
      <o-ex20:constraint constraintID="constraint02">
         <o-ex20:type>dutyfulfilled</o-ex20:type>
         <o-ex20:operator type="prefix" valency="Unary"> Fulfilled </o-ex20:operator>
         <o-ex20:operand>duty01</o-ex20:operand>
         </o-ex20:constraint>
      <o-ex20:constraint constraintID="constraint03">
         <o-ex20:type>spatial</o-ex20:type>
         <o-ex20:operator type="infix" valency="Binary"> NotEqual </o-ex20:operator>
         <o-ex20:operand position="left">Country</o-ex20:operand>
         <o-ex20:operand position="right">Australia</o-ex20:operand>
      </o-ex20:constraint>
      <o-ex20:constraint constraintID="constraint04">
         <o-ex20:type>datetime</o-ex20:type>
         <o-ex20:operator type="infix" valency="Binary"> LessThan </o-ex20:operator>
         <o-ex20:operand position="left">today</o-ex20:operand>
         <o-ex20:operand position="right">2004-04-23</o-ex20:operand>
      </o-ex20:constraint>

      <o-ex20:duty dutyID="duty01" bearer="party02">
         <o-dd20:prePayment>
            <o-dd20:amount currency="EUR"> 5.00 </o-dd20:amount>
         </o-dd20:prePayment>
         <o-ex20:hasConstraint>constraint04</o-ex20:hasConstraint>
      </o-ex20:duty>
   </o-ex20:agreement>
</o-ex20:rights>
```

The next listing shows an example of a barter contract. Here, one permission ⟨modify, ODRL_1.1⟩ is simply exchanged against a second permission ⟨print, ODRL_Workshop_proceedings⟩ between Renato Iannella and Susanne Guth. No monetary payment has to be made by either party; both permissions expire with the end of year 2010.

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex20:rights
        xmlns:o-ex20="http://odrl.net/2.0/ODRL-EX"
        xmlns:o-dd20="http://odrl.net/2.0/ODRL-DD">
    <o-ex20:agreement>
        <o-ex20:party o-ex20:partyID="party01">
            <o-ex20:context>
                    <o-dd20:uid>x500:c=AT;o=Registry;cn=sguth</o-dd20:uid>
                    <o-dd20:name>Susanne Guth</o-dd20:name>
            </o-ex20:context>
        </o-ex20:party>
        <o-ex20:party o-ex20:partyID="party02">
```

```
            <o-ex20:context>
               <o-dd20:uid>x500:c=AU;o=Registry;cn=riannel</o-dd20:uid>
               <o-dd20:name>Renato Iannella</o-dd20:name>
            </o-ex20:context>
        </o-ex20:party>

        <o-ex20:asset o-ex20:assetID="asset01">
            <o-ex20:context>
               <o-dd20:uid>urn:wu-wien.ac.at#proc01</o-dd20:uid>
               <o-dd20:name>ODRL Intl. Workshop '04 Proceedings</o-dd20:name>
            </o-ex20:context>
        </o-ex20:asset>
        <o-ex20:asset o-ex20:assetID="asset02">
            <o-ex20:context>
               <o-dd20:uid>urn:odrl.net#ODRLspec1.1</o-dd20:uid>
               <o-dd20:name>ODRL 1.1</o-dd20:name>
            </o-ex20:context>
        </o-ex20:asset>

        <o-ex20:permission o-ex20:grantor="party01">
            <o-ex20:operation>print</o-ex20:operation>
            <o-ex20:object>asset01</o-ex20:object>
            <o-ex20:beneficiary id="party02"/>
            <o-ex20:hasConstraint>constraint01</o-ex20:hasConstraint>
        </o-ex20:permission>
        <o-ex20:permission o-ex20:grantor="party02">
            <o-ex20:operation>modify</o-ex20:operation>
            <o-ex20:object>asset02</o-ex20:object>
            <o-ex20:beneficiary id="party01"/>
            <o-ex20:hasConstraint>constraint01</o-ex20:hasConstraint>
        </o-ex20:permission>

        <o-ex20:constraint o-ex20:constraintID="constraint01">
            <o-ex20:type>datetime</o-ex20:type>
            <o-ex20:operator o-ex20:type="infix" o-ex20:valency="Binary">LessThan</o-ex20:operator>
            <o-ex20:operand o-ex20:position="left">today</o-ex20:operand>
            <o-ex20:operand o-ex20:position="right">2011-01-01</o-ex20:operand>
        </o-ex20:constraint>
    </o-ex20:agreement>
</o-ex20:rights>
```

Note that the XML schemas proposed in this paper (listed in the Appendix) are an approach to a more flexible and expressive future version of ODRL and should be further discussed by the ODRL initiative.

5 Mapping to Enforceable Policies

Another essential aspect that was not yet discussed is the mapping of permissions and duties that are defined on the level of digital contracts to policy rules that can be enforced in an actual software system. Though this topic is by far to complex to be discussed in this paper, we would like to mention the corresponding problem domain and outline some issues arising in this context. Figure 4 shows a simplified information model for permissions and duties as they can be defined on a technical level, i.e. on the level of actual software systems that need to enforce the corresponding policy rules.

A particular difference between the information model for digital contracts introduced in Section 3 and the model shown in Figure 4 is the direct relation between duties and permissions. In a software system, a subject (a party) that must fulfill a certain duty inevitably needs a corresponding permission. Therefore, each duty must

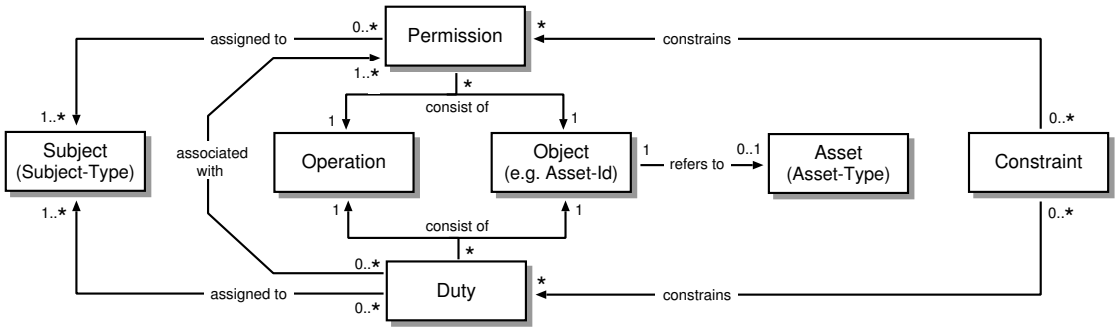

Figure 4: Simplified Information Model for System-level Permissions and Duties

be associated with (at least) one corresponding permission (note that while Figure 4 indicates that permissions and duties consist of $\langle operation, object \rangle$ pairs respectively, two related permission and duty objects do *not* necessarily refer to the same $\langle operation, object \rangle$ pair!). For example, a subject may only fulfill the duty "transfer money" iff the subject simultaneously possesses a corresponding permission which grants access to a specific banking account. Such information, however, is not (and typically should not be) modeled on the level of digital (business) contracts.

Nevertheless, in order to automatically process digital contracts and to enforce permissions and duties that are defined via ODRL contracts, one needs to specify a mapping of contract level elements to corresponding elements on the level of corresponding permissions and duties in concrete software systems. In the general case, each duty (or permission) defined on the contract level maps to one or more permission *and* duty objects on the "technical level" (as indicated by Figure 5). Moreover, constraints defined via a digital contract may (of course) only be enforced on a technical level if a respective (software) service exists which supports the corresponding type of constraints (see e.g. [16]). Another important issue that needs to be addressed is the definition and verification of "trust chains", for example to examine if a certain grantor (a party granting/assigning a permission to another party) actually is in possession of the respective control right, i.e. if the grantor was allowed/legitimated to pass the corresponding permission to another party (see e.g. [2, 4, 17, 18]).

6 Related Work

In the work of Keller et al. [15] a management architecture for specifying, deploying, monitoring, and enforcing service contracts is proposed to provide a basis for service level agreements. Contracts contain agreements about quality of service (QoS) attributes, they are concluded between service providers and a service integrator. This contract model is tailored to the needs of service level agreements, and thus contains different contract objects than the model discussed in this paper. However, their model also contains basic contract objects, such as provider, customer, and service, as well as objects that represent the guaranteed service parameters (rights). Keller et al., however, do not envision the exchange of contract information between the involved components in a standardized format, such as a rights expression language.

In [3], Beugnard et al. introduce a general model of software contracts that aims at increasing trust and reliability between software components. To conclude contracts between components, every component publishes a feature set to describe its services in a specific language (e.g. CORBA IDL). Contracts are established between a client and server component in a negotiation phase where the contract parties agree on certain services.

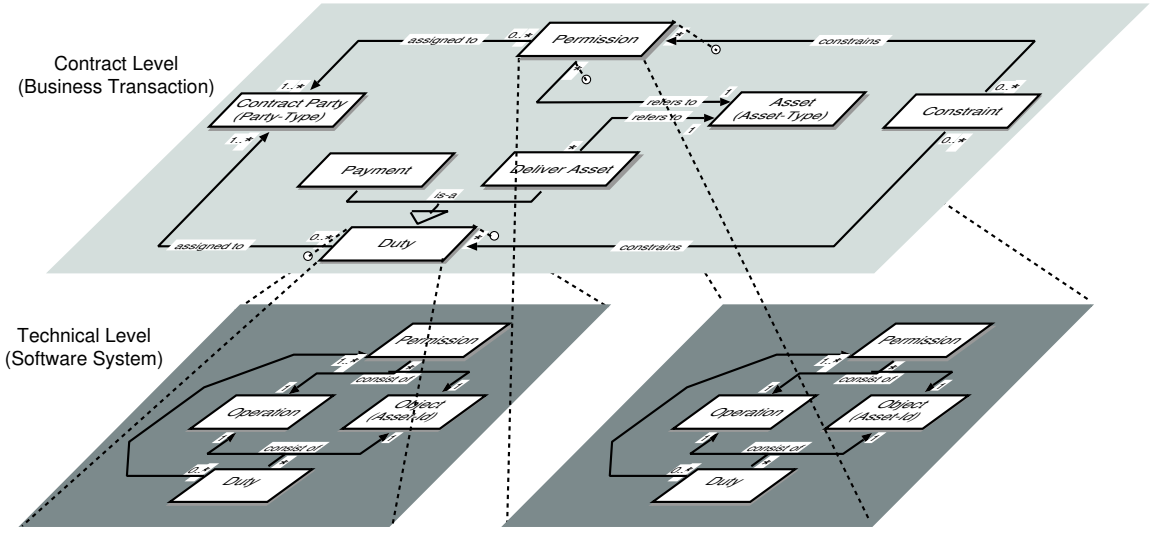

Figure 5: Mapping of Contract Level Elements to Enforceable Policies on the Software System Level

The work provides a basic interface description for the negotiation phase. Beugnard et al. suggest an "XML-formatted description of the contracts" that is applied for negotiation purposes. This is somewhat similar to the approach presented in this paper.

The eXtensible Access Control Markup Language (XACML) [10] is a standard adopted by the Organization for the Advancement of Structured Information Standards (OASIS). XACML provides an XML-based language for the definition of access control policies. The language syntax is formalized via an XML schema. Beside standard access control policies including subjects, operations, and objects, it also allows for the definition of obligations and conditions. Conditions are boolean functions over attributes associated with a subject, an operation, an object, or the system environment. XACML environment attributes are attributes which are relevant to an authorization decision but are independent of a particular subject, operation, or object (see [10]).

The Security Assertion Markup Language (SAML) [12] is an other standard adopted by OASIS. SAML defines an XML-based framework for exchanging security information via computer networks. It is based on the SAML protocol which consists of XML-based request and response messages. By this protocol, clients can request assertions from so-called "SAML authorities" (trusted servers). SAML authorities can make three different kinds of assertion statements: authentications, authorization decisions, and attributes. An authentication assertion confirms that a specific subject has been authenticated by a particular means at a particular time. An authorization decision assertion states that a particular access request consisting of a $\langle subject, operation, object \rangle$ triple has been granted by the corresponding SAML authority. Finally, an attribute assertion confirms that a specific subject is associated with a certain set of attributes.

7 Conclusion and Future Work

In this paper we discussed the information model of ODRL version 1.1. In particular, we focussed on the ODRL *Rightsholder* element, the ODRL *Permission* element, as well as the elements that define further "characteristics" of ODRL Permissions, such as ODRL *Condition*, ODRL *Constraint*, and ODRL *Requirement*. When

investigating the current ODRL information model we identified several drawbacks in terms of expressiveness and with respect to the mapping of ODRL rights information to enforceable (access control) policies on the level of actual software systems. For example, with the ODRL 1.1 information model, rights and duties can only be expressed for the purchasing contract party but not for the seller. In this paper, we explained the identified shortcomings in detail, and proposed an information model for ODRL which aims to eliminate these drawbacks. This information model is translated to corresponding XML schemas which serve as a proposal for future ODRL specifications. The resulting XML schemas can be found in the appendix of this paper. The future work in this field is clearly to embed the improvements into an official, future ODRL version in order to ensure (and further advance) the usability and timeliness of the ODRL.

A Proposed XML Schema for the ODRL Expression Language

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://odrl.net/2.0/ODRL-EX"
      xmlns:o-ex20="http://odrl.net/2.0/ODRL-EX"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
      elementFormDefault="qualified" attributeFormDefault="qualified" version="2.0">
<xsd:import
      namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
<!-- NOTE: The W3C Encryption Namespace URI will be updated as the specification is advanced -->
<xsd:import
      namespace="http://www.w3.org/2001/04/xmlenc#"
      schemaLocation="http://www.w3.org/Encryption/2001/Drafts/xmlenc-core/xenc-schema.xsd"/>
      <xsd:element name="rights" type="o-ex20:rightsType"/>
      <xsd:element name="offer" type="o-ex20:offerAgreeType"/>
      <xsd:element name="agreement" type="o-ex20:offerAgreeType"/>
      <xsd:complexType name="offerAgreeType">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element ref="o-ex20:party" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element ref="o-ex20:asset" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element ref="o-ex20:permission" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element ref="o-ex20:duty" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element ref="o-ex20:constraint" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:choice>
      </xsd:complexType>
      <xsd:complexType name="rightsType">
            <xsd:complexContent>
                  <xsd:extension base="o-ex20:offerAgreeType">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                              <xsd:element ref="o-ex20:revoke" minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="o-ex20:offer" minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="o-ex20:agreement" minOccurs="0" maxOccurs="unbounded"/>
                              <xsd:element ref="ds:Signature" minOccurs="0"/>
                        </xsd:choice>
                        <xsd:attributeGroup ref="o-ex20:IDGroup"/>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:element name="context" type="o-ex20:contextType"/>
      <xsd:element name="contextElement" abstract="true"/>
      <xsd:complexType name="contextType">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xsd:element ref="o-ex20:contextElement" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attributeGroup ref="o-ex20:IDGroup"/>
</xsd:complexType>
<xsd:element name="duty" type="o-ex20:dutyType"/>
<xsd:element name="dutyElement" abstract="true"/>
<xsd:complexType name="dutyType">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex20:dutyElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:hasConstraint" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attribute name="dutyID" type="xsd:string" use="required"/>
      <xsd:attribute name="bearer" type="xsd:string" use="required"/>
      <xsd:attributeGroup ref="o-ex20:IDGroup"/>
</xsd:complexType>
<xsd:complexType name="partyType">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex20:context" minOccurs="0"/>
            <xsd:element ref="o-ex20:party" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:container" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:asset" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attributeGroup ref="o-ex20:IDGroup"/>
      <xsd:attribute name="partyID" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="party" type="o-ex20:partyType"/>
<xsd:complexType name="assetType">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex20:context"/>
            <xsd:element ref="o-ex20:inherit"/>
            <xsd:element name="digest">
                  <xsd:complexType>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                          <xsd:element ref="ds:DigestMethod"/>
                          <xsd:element ref="ds:DigestValue"/>
                        </xsd:choice>
                  </xsd:complexType>
            </xsd:element>
            <xsd:element ref="ds:KeyInfo"/>
      </xsd:choice>
      <xsd:attributeGroup ref="o-ex20:IDGroup"/>
      <xsd:attribute name="assetID" type="xsd:string" use="required"/>
      <xsd:attribute name="type">
            <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="work"/>
                        <xsd:enumeration value="expression"/>
                        <xsd:enumeration value="manifestation"/>
                        <xsd:enumeration value="item"/>
                  </xsd:restriction>
            </xsd:simpleType>
      </xsd:attribute>
</xsd:complexType>
<xsd:element name="asset" type="o-ex20:assetType"/>
<xsd:complexType name="inheritType">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attribute name="override" type="xsd:boolean" default="false"/>
      <xsd:attribute name="default" type="xsd:boolean" default="false"/>
</xsd:complexType>
```

```
<xsd:element name="inherit" type="o-ex20:inheritType"/>
<xsd:element name="permission" type="o-ex20:permissionType"/>
<xsd:element name="operation" type="operationType"/>
<xsd:element name="beneficiary" type="o-ex20:linkType"/>
<xsd:element name="object" type="o-ex20:assetType"/>
<xsd:complexType name="permissionType">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:operation" minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:object" minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:hasConstraint" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:beneficiary" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:attribute name="exclusive" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="grantor" type="xsd:string" use="required"/>
      <xsd:attributeGroup ref="o-ex20:IDGroup"/>
</xsd:complexType>
<xsd:complexType name="operandType">
      <xsd:attribute name="position" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="type" type="o-ex20:cType"/>
<xsd:element name="operand" type="o-ex20:operandType"/>
<xsd:complexType name="operatorType">
      <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                  <xsd:attribute name="type" use="required">
                        <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                <xsd:enumeration value="prefix"/>
                                <xsd:enumeration value="infix"/>
                                <xsd:enumeration value="postfix"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                  </xsd:attribute>
                  <xsd:attribute name="valence" use="required">
                        <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                    <xsd:enumeration value="unary"/>
                                    <xsd:enumeration value="binary"/>
                                    <xsd:enumeration value="ternary"/>
                                    <xsd:enumeration value="n-ary"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                  </xsd:attribute>
            </xsd:extension>
      </xsd:simpleContent>
</xsd:complexType>
<xsd:element name="operator" type="o-ex20:operatorType"/>
<xsd:complexType name="constraintType">
      <xsd:choice>
            <xsd:element ref="o-ex20:container" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:type" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="o-ex20:operand" minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex20:operator" minOccurs="1" maxOccurs="1"/>
      </xsd:choice>
      <xsd:attribute name="constraintID" type="xsd:string" use="required"/>
      <xsd:attribute name="type" type="xsd:NMTOKEN" use="required"/>
</xsd:complexType>
<xsd:element name="hasConstraint" type="linkType/">
<xsd:complexType name="linkType">
      <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
```

```
	<xsd:element name="constraint" type="o-ex20:constraintType"/>
	<xsd:complexType name="revokeType">
		<xsd:sequence minOccurs="0" maxOccurs="unbounded">
			<xsd:element ref="o-ex20:context" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attributeGroup ref="o-ex20:IDGroup"/>
	</xsd:complexType>
	<xsd:element name="revoke" type="o-ex20:revokeType"/>
	<xsd:complexType name="sequenceType">
		<xsd:sequence>
			<xsd:element ref="o-ex20:seq-item" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="order" default="total">
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="total"/>
					<xsd:enumeration value="partial"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
	</xsd:complexType>
	<xsd:element name="sequence" type="o-ex20:sequenceType"/>
	<xsd:complexType name="containerType">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:element ref="o-ex20:container" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:permission" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:constraint" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:sequence" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:party" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:choice>
		<xsd:attribute name="type" default="and">
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="and"/>
					<xsd:enumeration value="in-or"/>
					<xsd:enumeration value="ex-or"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attributeGroup ref="o-ex20:IDGroup"/>
	</xsd:complexType>
	<xsd:element name="container" type="o-ex20:containerType"/>
	<xsd:complexType name="seqItemType">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:element ref="o-ex20:container" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:permission" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:constraint" minOccurs="0" maxOccurs="unbounded"/>
			<xsd:element ref="o-ex20:sequence" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:choice>
		<xsd:attribute name="number" type="xsd:integer" use="required"/>
	</xsd:complexType>
	<xsd:element name="seq-item" type="o-ex20:seqItemType"/>
	<xsd:attributeGroup name="IDGroup">
		<xsd:attribute name="id" type="xsd:ID"/>
		<xsd:attribute name="idref" type="xsd:IDREF"/>
	</xsd:attributeGroup>
</xsd:schema>
```

B Proposed XML Schema for the ODRL Data Dictionary

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://odrl.net/2.0/ODRL-DD"
      xmlns:o-ex20="http://odrl.net/2.0/ODRL-EX"
      xmlns:o-dd20="http://odrl.net/2.0/ODRL-DD"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified" attributeFormDefault="qualified" version="2.0">
<xsd:import namespace="http://odrl.net/2.0/ODRL-EX" schemaLocation="http://odrl.net/2.0/ODRL-EX-20.xsd"/>
      <!-- Declare the operation Vocabulary -->
      <xsd:simpleType name="operationType">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="display"/>
                  <xsd:enumeration value="print"/>
                    <xsd:enumeration value="play"/>
                  <xsd:enumeration value="execute"/>
                    <xsd:enumeration value="sell"/>
                    <xsd:enumeration value="lend"/>
                    <xsd:enumeration value="give"/>
                    <xsd:enumeration value="lease"/>
                    <xsd:enumeration value="modify"/>
                    <xsd:enumeration value="excerpt"/>
                    <xsd:enumeration value="aggregate"/>
                    <xsd:enumeration value="annotate"/>
                    <xsd:enumeration value="move"/>
                    <xsd:enumeration value="duplicate"/>
                    <xsd:enumeration value="delete"/>
                    <xsd:enumeration value="verify"/>
                    <xsd:enumeration value="backup"/>
                    <xsd:enumeration value="restore"/>
                    <xsd:enumeration value="install"/>
                    <xsd:enumeration value="uninstall"/>
                    <xsd:enumeration value="save"/>
            </xsd:restriction>
      </xsd:simpleType>
      <!-- Declare the Payment Elements -->
      <xsd:element name="payment">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="amount">
                              <xsd:complexType>
                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:decimal">
                                      <xsd:attribute name="currency" type="xsd:NMTOKEN" use="required"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                              </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="taxpercent" minOccurs="0">
                              <xsd:complexType>
                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:decimal">
                                      <xsd:attribute name="code" type="xsd:NMTOKEN" use="required"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                              </xsd:complexType>
                        </xsd:element>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
      <!-- Declare all the Duty Elements -->
      <xsd:element name="perusePayment" substitutionGroup="o-ex20:dutyElement">
```

```
      <xsd:complexType>
            <xsd:complexContent>
                  <xsd:extension base="o-ex20:dutyType">
                        <xsd:sequence>
                          <xsd:element ref="o-dd20:payment"/>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:element>
<xsd:element name="postPayment" substitutionGroup="o-ex20:dutyElement">
      <xsd:complexType>
            <xsd:complexContent>
                  <xsd:extension base="o-ex20:dutyType">
                        <xsd:sequence>
                          <xsd:element ref="o-dd20:payment"/>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:element>
<xsd:element name="prePayment" substitutionGroup="o-ex20:dutyElement">
      <xsd:complexType>
            <xsd:complexContent>
                  <xsd:extension base="o-ex20:dutyType">
                        <xsd:sequence>
                                    <xsd:element ref="o-dd20:payment"/>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:element>
<xsd:element name="deliverAsset" substitutionGroup="o-ex20:dutyElement">
      <xsd:complexType>
            <xsd:complexContent>
                  <xsd:extension base="o-ex20:dutyType">
                        <xsd:sequence>
                          <xsd:element ref="o-ex20:asset"/>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:element>
<!--Duty elements for accept, register, attribution, tracked have to be formulated accordingly
    Declare all the Context Elements -->
<xsd:simpleType name="uriAndOrString">
      <xsd:union memberTypes="xsd:anyURI xsd:string"/>
</xsd:simpleType>
<xsd:element name="uid" type="o-dd20:uriAndOrString" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="role" type="xsd:anyURI" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="name" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="remark" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="event" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="pLocation" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="dLocation" type="xsd:anyURI" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="reference" type="xsd:anyURI" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="version" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="transaction" type="xsd:string" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="service" type="xsd:anyURI" substitutionGroup="o-ex20:contextElement"/>
<xsd:element name="date" type="o-dd20:dateType" substitutionGroup="o-ex20:contextElement"/>
<!-- Declare all the Constraint Elements -->
<xsd:simpleType name="cType">
```

```
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="dutyfulfilled">
                  <xsd:enumeration value="individual"/>
                    <xsd:enumeration value="group"/>
                    <xsd:enumeration value="cpu"/>
                    <xsd:enumeration value="network"/>
                    <xsd:enumeration value="screen"/>
                    <xsd:enumeration value="storage"/>
                    <xsd:enumeration value="memory"/>
                    <xsd:enumeration value="printer"/>
                    <xsd:enumeration value="software"/>
                    <xsd:enumeration value="hardware"/>
                    <xsd:enumeration value="spatial"/>
                    <xsd:enumeration value="quality"/>
                    <xsd:enumeration value="format"/>
                    <xsd:enumeration value="unit"/>
                    <xsd:enumeration value="watermark"/>
                    <xsd:enumeration value="purpose"/>
                    <xsd:enumeration value="industry"/>
                    <xsd:enumeration value="count"/>
                    <xsd:enumeration value="minimum"/>
                    <xsd:enumeration value="maximum"/>
                    <xsd:enumeration value="datetime"/>
                    <xsd:enumeration value="accumulated"/>
                    <xsd:enumeration value="interval"/>
                    <xsd:enumeration value="recontext"/>
            </xsd:restriction>
      </xsd:simpleType>
      <!-- Transfer Permission is defined as a ContainerType to enable complete expression of
       rights in the Constraint  -->
      <xsd:element name="transferPerm" substitutionGroup="o-ex20:container">
            <xsd:complexType>
                  <xsd:complexContent>
                        <xsd:extension base="o-ex20:containerType">
                              <xsd:attribute name="downstream" default="equal">
                                    <xsd:simpleType>
                                          <xsd:restriction base="xsd:NMTOKEN">
                                            <xsd:enumeration value="equal"/>
                                            <xsd:enumeration value="less"/>
                                            <xsd:enumeration value="notgreater"/>
                                          </xsd:restriction>
                                    </xsd:simpleType>
                              </xsd:attribute>
                        </xsd:extension>
                  </xsd:complexContent>
            </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

References

[1] G.J. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4), November 2000.

[2] J. Bacon and K. Moody. Toward Open, Secure, Widely Distributed Services. *Communications of the ACM*, 45(6), June 2002.

[3] A. Beugnard, J.-M. Jezequel, N. Plouzeau, and D. Watkins. Making Components Contract Aware. *IEEE Computer Magazine*, 32(7), July 1999.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of the IEEE Conference on Security and Privacy*, May 1996.

[5] National Computer Security Center. A Guide to Understanding Discretionary Access Control in Trusted Systems, September 1987. NCSC-TG-003-87.

[6] ContentGuard Inc. eXtensible rights Markup Language (XrML), Version 2.0. http://www.xrml.org/, November 2001.

[7] T. DeMartini, X. Wang, and B. Wragg. MPEG-21 Working Documents - Part 5 & Part 6, MPEG-21 Rights Expression Language. http://www.chiariglione.org/mpeg/working_documents.htm, March 2003.

[8] D.E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5), May 1976.

[9] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3), August 2001.

[10] S. Godik and T. Moses (eds.). eXtensible Access Control Markup Language (XACML) Version 1.0. http://www.oasis-open.org, February 2003. OASIS Standard.

[11] S. Guth, G. Neumann, and M. Strembeck. Experiences with the Enforcement of Access Rights Extracted from ODRL-based Digital Contracts. In *Proc. of the 3rd ACM Workshop on Digital Rights Management (DRM)*, October 2003.

[12] P. Hallam-Baker and E. Maler (eds.). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). http://www.oasis-open.org, November 2002. OASIS Standard.

[13] R. Iannella. Open Digital Rights Language (ODRL), Version 1.1. http://odrl.net, August 2002.

[14] T. Jaeger. On the Increasing Importance of Constraints. In *Proc. of the ACM Workshop on Role-Based Access Control*, 1999.

[15] A. Keller, G. Kar, H. Ludwig, A. Dan, and J.-L. Hellerstein. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. In *Proc. of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2002.

[16] G. Neumann and M. Strembeck. An Approach to Engineer and Enforce Context Constraints in an RBAC Environment. In *Proc. of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT)*, June 2003.

[17] K.E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for Policy Languages for Trust Negotiation. In *Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY)*, June 2002.

[18] W.H. Winsborough and N. Li. Towards Practical Automated Trust Negotiation. In *Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY)*, June 2002.