Extending ODRL and XrML to Enable Bi-Directional Communication

Technical Report – CS04-28-00

Alapan Arnab*
Department of Computer Science
University of Cape Town
Cape Town, South Africa
aarnab@cs.uct.ac.za

Andrew CM Hutchison†
Department of Computer Science
University of Cape Town
Cape Town, South Africa
hutch@cs.uct.ac.za

Abstract

Current rights expression languages (RELs) only allow for rights holders to dictate terms to the end users. This limits their use as a means for negotiating electronic contracts and end users are not able to request for changes in their rights contracts. In this paper we propose extensions to two popular XML based RELs: ODRL and XrML that allow for end users to request changes as well as the rights holder to grant or deny these changes. These extensions allow the end user to request for changes to their current rights, and for the rights holder to grant or refuse the request. We also provide 2 examples for each REL to demonstrate possible uses of our extensions.

1 Introduction

Right Expression Languages or RELs are perhaps the most important part of DRM systems, since they allow rights holders to dictate the terms and conditions which need to be upheld by DRM systems. RELs have been designed to provide flexibility to DRM systems, allowing rights holders to offer a variety of terms and conditions without requiring a change in DRM systems. This feature is well demonstrated in Microsoft's Rights Management Services (RMS) system, which allow content authors to set various access control rights to the content using eXtensible rights Markup Language or XrML [4].

However, RELs have been criticised for being too restrictive and giving rights holders too much control. Mulligan et al. have argued that "rights" in DRM may possibly have no relationship with legal rights, but should rather be described as "permissions" given by the rights holders to the users [8]. This stems from the access control models used by most RELs – only rights expressed in the usage license, are granted to the user, and thus rights not mentioned are considered to be not granted. This is partly blamed on missing semantics in the RELs. For example, Open Digital Rights Language (ODRL) has been criticised by some for not having a "*not*" semantic [10].

Others, like Felten [7] have argued that expressing legal rights connected to copyright are too broad, and would require a highly sophisticated AI to interpret and implement, and current RELs are unsuitable for implementing legal rights.

Mulligan et al. [8] argued that contracts are negotiated between the involved parties, and true negotiations require parties to communicate. Referring to XrML, Mulligan et al. argued that "*the assumption of a one-way expression of rights has in part led to the current deficiencies in the REL*" [8]. To allow for bi-directional communication, it requires the REL to provide the vocabulary and syntax, and an extension to the *rights messaging protocol* (RMP). The work presented in this paper

*Alapan Arnab is a MSc candidate at the University of Cape Town

†Prof. Andrew CM Hutchison is an Adjunct Professor of Computer Science at the University of Cape Town

falls under a broader project *Distributed DRM System*, and a bi-directional RMP is also part of this project, but not in the scope of this particular paper.

The problem with the current system can be best represented using an example from the second scenario in Microsoft's overview of RMS [4]. Tom creates a document for Jill, and protects it using RMS. He specifies that the document can only be viewed and edited by Jill for one week. If Jill requires additional time, Tom is required to edit the rights to the document, extend the deadline and then redistribute the document to Jill. However, this solution is impractical when the data sizes become too large.

With a bi-directional REL, it should allow the user and rights holder to conduct negotiations on the rights the user is given. This process can take more than a single round of "requests" to the rights holder and "offers" to the user. Furthermore, a bi-directional REL should also allow a user to request changes to an existing use license. With bi-directional RELs it would thus be possible to cater for fair use at a general level – rights holders can issue use licenses with usage rules fair for majority of the users. If there are users who require additional privileges that fall under fair use (academics who would like to create extra copies for their lectures, journalists who would like to excerpt a quote for a review etc.), they can easily negotiate for these additional rules.

In this paper, we introduce vocabulary and syntax to facilitate bi-directional communication in two RELs; namely ODRL and XrML. With bi-directional communication the user can request a right and the rights holder can then create a license amendment or issue a new license with the additional rights.

2 Design

In 2000, Park et al. [9] discussed the different distribution architectures that could be implemented for secure content distribution. Park et al. distinguished various architectures with three criteria: the presence of a virtual machine, the type of control set and the distribution style. They concluded that a virtual machine is required for secure content distribution, while the type of control sets and distribution style dictate the amount of control the "owner" of the content has after distribution. In a DRM system, the virtual machine represents the DRM controller and the control set represents the REL and the usage licence mechanisms.

Park et al. categorised control sets into three types: fixed control sets, embedded control sets and external control sets [9]. In fixed control sets, the DRM system comes with a predefined set of controls, and thus the DRM enabled data does not have to have any additional controls. In embedded control sets, the DRM enabled data comes with a set of controls as a single secure package while in external control sets, the control set and the DRM enabled data come in separate packages. It is possible to combine multiple type of control sets, as long as the DRM controller can regulate which control sets should be implemented; e.g. if the fixed control set does not allow copying, but the embedded control set (issued after the fixed control set) does allow copying then the DRM controller should allow copying. This is important for bi-directional communication; since the control set issued by the rights holders is most likely to be an external control set, and the DRM enabled data might already have an embedded control set.

In our design we envisage a bi-directional system to be implemented as a web-service. Thus a user would *request* for changes to their current rights and can expect to receive three types of responses. Firstly, the rights holders can grant the request and issue a new license, which can be easily expressed with the current schemas for both ODRL and XrML. Alternatively, the rights holders can grant the request by creating a licence addendum (in a separate file) (*grant-request*). To handle this response, the DRM controller must be able to detect and use the extended license. Lastly, the rights holders can deny the request (*deny-request*). The user would need to be informed which requests

are being denied since it may happen that the user requested three changes, of which only one is granted. Thus, in both the *grant-request* and *deny-request* there would be a need to include the requests.

There are three actions that a user could request:

- Request to add one or more permissions, resources etc. that are either not currently present or to extend the current values e.g. add one more week to the deadline
- Request to remove one or more permissions, resources etc. that have been granted through an earlier license or license addendum. This feature is most probably not going to be in big demand, but is necessary for:
- Request to replace one or more permissions that have been granted through an earlier license or license addendum. The request to replace is essentially a combination of an add and a remove request, but it would be more useful for tracking purposes to utilise a replace request mechanism. There should not be any restriction on how the replace mechanism is used – for example a user might request a replacement of dis-similar permissions, e.g. replace his right to print 5 copies with the right to make a backup.

With a bi-directional system, it would require the rights holders to keep track of individual licenses, and how the licenses inter relate. The grant-request licenses should also be able to identify (possibly through the use of a URI) the original request as well as the original license. This would allow the DRM controller to keep track of the permissions, resources, etc. that have been removed or changed. For example, if the user originally had permission to print a document 2 times, printed it once, and then requested and received permission to print the document an additional 5 times, the DRM controller should allow the user to print 6 more times.

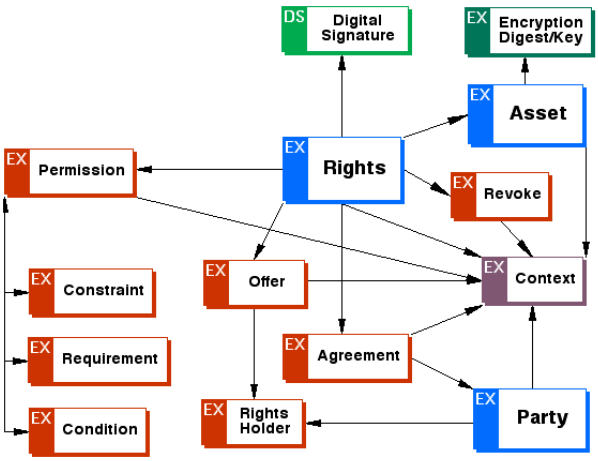

Figure 1: ODRL Foundation Model [2]

3 ODRL

3.1 Background

Open Digital Rights Language (ODRL) is an open rights expression language that is published and "available in the spirit of "open source" software" [2]. It is well supported by various corporations like Nokia and IBM; and has been accepted by the Open Mobile Alliance as the standard for rights expression in Mobile DRM systems.

Figure 1 shows the overall structure of ODRL. ODRL revolves around three core entities, Assets, Rights and Parties. *Assets* are physical or digital entities that can be uniquely identified. *Rights* comprise of permissions and their constraints and requirements that an end user is granted to use over the assets. Both the end user and the rights holders are *parties*.

Using ODRL, a rights holder can *offer* the end users terms for using a DRM protected asset. With this feature ODRL allows the rights holders to express their right policy; and this could be given to distributors for implementation and can include details of payments, restrictions, etc. When an end user accepts an offer, an *agreement* is created. The agreement captures the details of a completed transaction, and essentially creates an end user li-

cense with all the terms of the agreement. This model allows the rights holder to create a variety of offers for the end user's usage.

3.2 odrl-ext

Our extension adds three more entities – request, grant-request and deny-request – and are modelled on the agreement entity. We envisage its main use in a web-services environment. The end-user can request the rights holder for a set of rights on a set of assets. The rights holder can then deny or grant that request.

This model can be further extended where the rights holder can offer various rights at various prices. The prospective end user can then request for a combination of rights, pay for these rights and then receive an end user license. Thus in this manner the request entity can be used for electronic contract negotiation. The grant and deny request entities can be used to conditionally accept or reject requests during the contract negotiation.

3.2.1 Add, Remove and Replace

A request from a user can take the form of adding, removing or replacing a set of permissions, constraints, requirements, conditions, assets or even parties. Thus the remove and add request elements are simply instances of the offerAgreeType in the ODRL Expression Language Schema [2]. The *replace-request* element comprises of a set of remove requests followed by a set of add requests. Although a replace-request element is not necessary, we believe that this element would allow for better tracking and management by the rights holders. Figures 2,3 and 4 show the content model for the add, remove and replace elements.

3.2.2 Request

The *requestType* creates an envelope containing all the add, remove and replace requests from the user as well as the context of the request and information about the party making the request. The *context* element allows the rights holder to reconcile

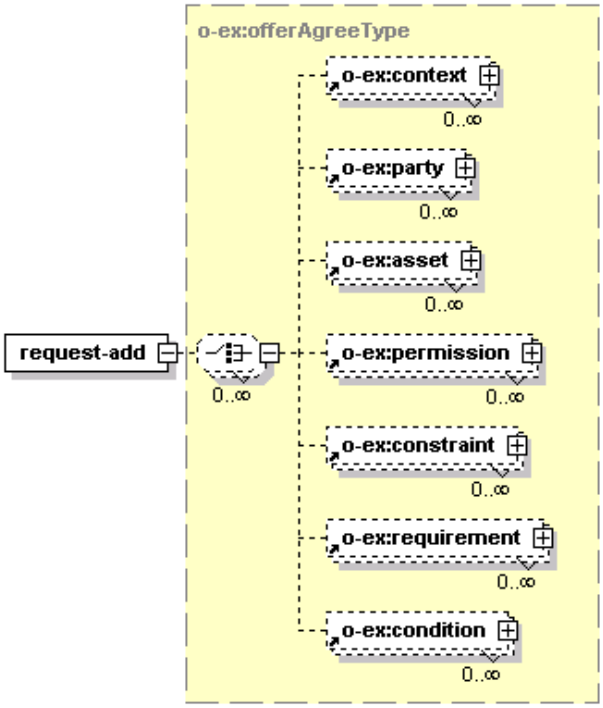

Figure 2: The Add Request Content Model

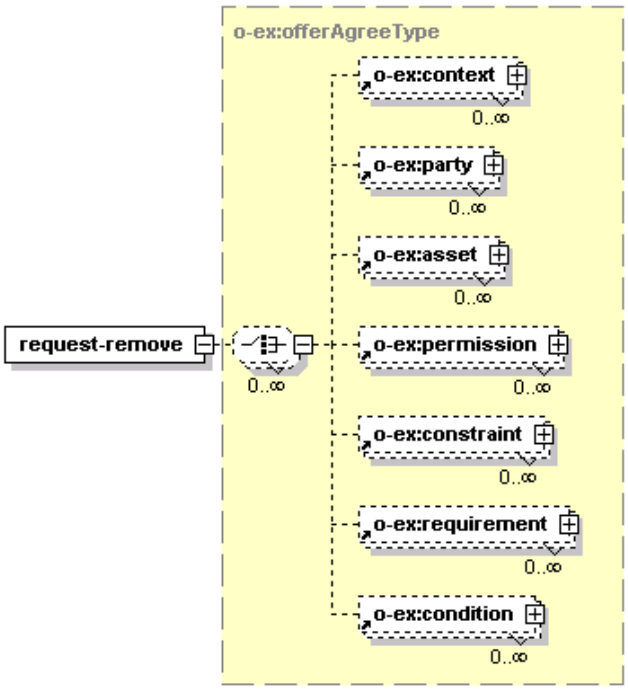

Figure 3: The Remove Request Content Model

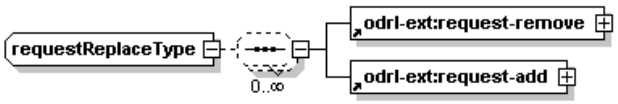

Figure 4: The Replace Request Content Model

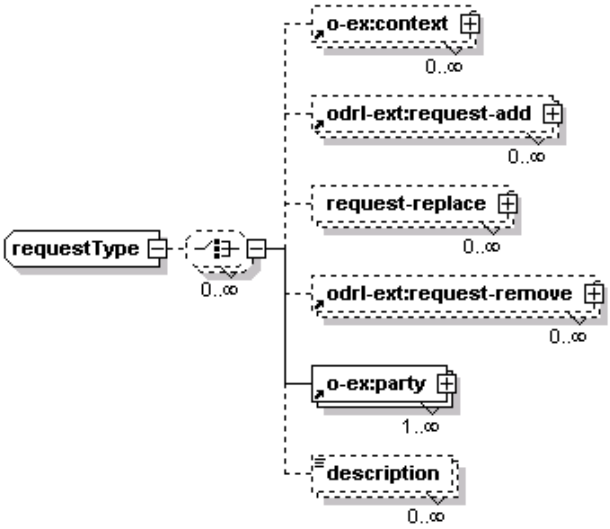

Figure 5: The Request Content Model

the request against an existing agreement or an offer. At least one party is required to identify the party making the request. The description element allows for the end user to write notes, and give more detailed information to the rights holder. If the request is processed manually, this feature can be very useful. Figure 5 shows the content model of the requestType. The request element is the only element of the requestType.

3.2.3 Request Response

The *requestResponseType* creates an envelope for the rights holder to respond back to the user making the request. There are two differences between the requestType and the requestResponseType. Firstly, the response from the rights holders must have a context, either of an earlier request or of the affected agreement. This will allow the DRM controller to keep track of the chain of agreements that it needs to manage and also allow the rights holders to track their responses to requests. Sec-

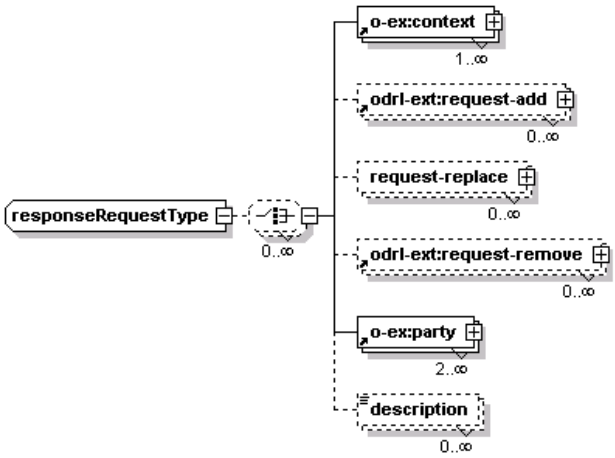

Figure 6: The Response-Request Content Model

ondly the response must have at least two parties - one identifying the user who made the request and another to identify the rights holder. Figure 6 shows the content model of the requestResponse type. Both the grant and deny request elements are of this type.

3.2.4 rightsType

In ODRL 1.1 the *rightsType* complex type encapsulates agreements and offers with a digital signature and a revoke mechanism. We extended this type to encapsulate the request, grant-request and deny-request elements.We have also redefined the rights element to be of this type. Figure 7 shows the content model of the rights type. The rightsType in ODRL 1.1 extends the offerAgreeType and this portion has been collapsed in the diagram.

3.2.5 Examples

In scenario 2 of the ODRL 1.1 specifications, a consumer (Mary Smith) purchases an ebook under conditions laid down in scenario 1 [2]. One of the conditions set in scenario 1 is that the consumer is only allowed to print a maximum of 2 copies of the ebook.

In example 1, the consumer requests the rights holders to be allowed to print the ebook 5 more times. Note, that for the sake of clarity we have

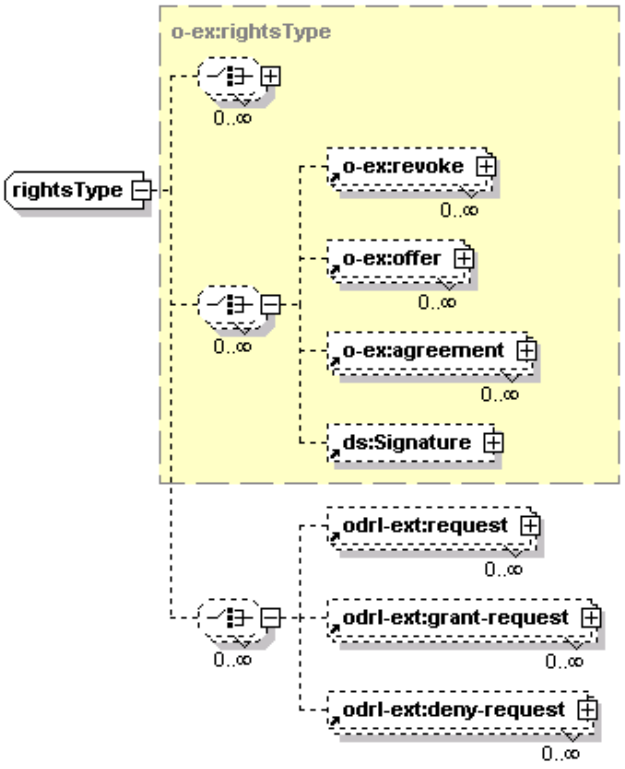

Figure 7: The rightsType Content Model

left the namespace definitions and schema locations out of the example. The descriptions of the namespaces are detailed below.

odrl-ext: The extended ODRL schema as discussed in this section.

o-ex: The *Expression Language Schema* of the ODRL 1.1 specifications.

o-dd: The *Data Dictionary Schema* of the ODRL 1.1 specifications.

Example 2 shows a grant request should the rights holders grant the user's request. A deny request would be the same except the *grant-request* elements will be replaced with the *deny-request* element.

3.2.6 Full Listing

A full listing of the schema definition is available in appendix A.

```
<odrl-ext:rights>
  <odrl-ext:request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/lic
ense/1234567890-ABCDEF</o-dd:uid>
    </o-ex:context>
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:print>
          <o-ex:constraint>
            <o-dd:count>5</o-dd:count>
          </o-ex:constraint>
        </o-dd:print>
      </o-ex:permission>
    </odrl-ext:request-add>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>
          urn:ebook.world/999999/users/msm
th-000111
        </o-dd:uid>
        <o-dd:name>Mary Smith</o-dd:name>
      </o-ex:context>
    </o-ex:party>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 1: ODRL Request

```
<odrl-ext:rights>
  <odrl-ext:grant-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/lic
ense/1234567890-GHIJKL</o-dd:uid>
    </o-ex:context>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/lic
ense/1234567890-ABCDEF</o-dd:uid>
    </o-ex:context>
```

```
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:print>
          <o-ex:constraint>
            <o-dd:count>5</o-dd:count>
          </o-ex:constraint>
        </o-dd:print>
      </o-ex:permission>
    </odrl-ext:request-add>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>urn:ebook.world/999999/
users/msmth-000111</o-dd:uid>
        <o-dd:name>Mary Smith</o-dd:name>
      </o-ex:context>
    </o-ex:party>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>x500:c=AU;o=RightsDir;cn
=AddisonRossi</o-dd:uid>
      </o-ex:context>
    </o-ex:party>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid>x500:c=AU;o=RightsDir;cn
=EBooksRUS
        </o-dd:uid>
      </o-ex:context>
    </o-ex:party>
  </odrl-ext:grant-request>
</odrl-ext:rights>
```

Example 2: ODRL Grant Request

4 XrML

4.1 Background

Unlike ODRL, XrML is a proprietary REL developed by ContentGuard. It is based on Xerox Parc's Digital Property Rights Language (DPRL), which was written in Lisp [1]. Xerox Parc still has an interest in ContentGuard, though the major shareholders are now Microsoft and AOL Time Warner. XrML currently forms the basis of the Motion Pictures Expert Group's (MPEG) MPEG-21 Rights Expression Language. XrML can also be used for XML-based security tokens in the webservices security (WS-Security) framework [3]. The WS-Security framework extends the SOAP specifications to allow for message level security, but it is still in a draft status.

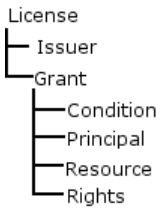

Figure 8: Simplified representation of the XrML hierarchy

[8]

Figure 8 shows a simplified representation of the XrML hierarchy. Unlike ODRL, in an XrML license, the issuer grants one or more users rights to resources under various conditions. To decrease repetitions, XrML uses an inventory to store all the principals, resources, conditions and rights. These are then referenced in the individual right expressions. This can be clearly seen in the XrML license model in figure 9. All the parties in the license are covered under *principals*, and they can include non human, non organisational entities such as network nodes or terminals [6]. *Resources* identify the object(s) that the license refers to and can include digital data like email. Rights and conditions place restrictions on the usage of the resource. Like ODRL, any right that is not specified in a license is not granted to the licensee. XrML also has a LicenseGroup structure that can be used to envelope multiple licenses in one file.

4.2 xrml-ext

As in our extension to ODRL, we add three more entities to XrML namely, request, grant-request and deny-request. These entities have been modelled slightly differently to the license model in order to make best use of XrML's inventory construct. Similar to our ODRL extension, we envisage the main use of these constructs in a web services environment. The end-user requests the rights holders for a set of grants, to which the rights

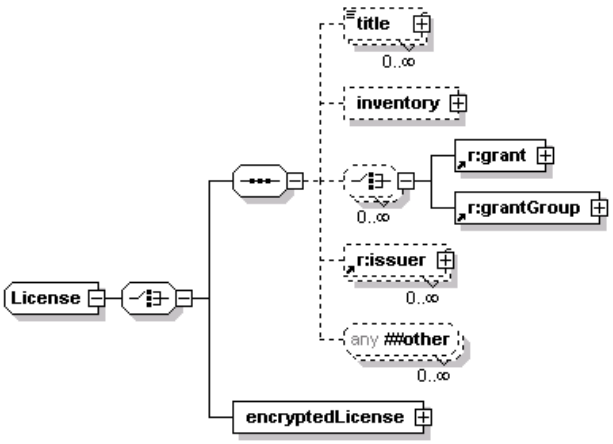

Figure 9: XrML License Model

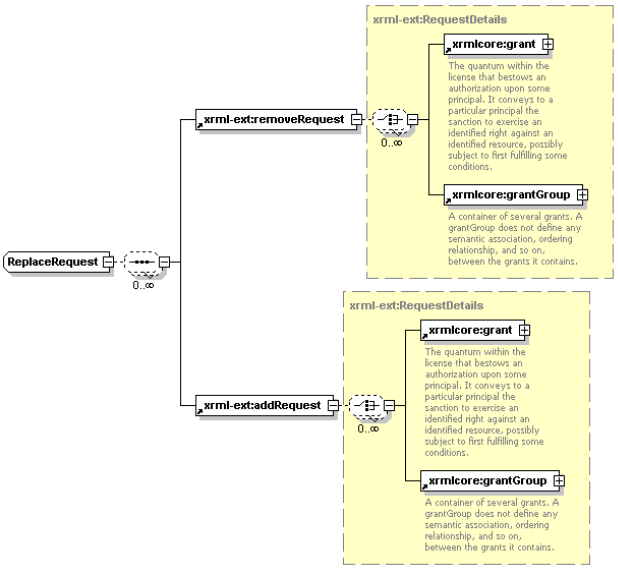

Figure 10: XrML Extension: Replace Request Element Model

holders can then agree or refuse. This model can also be used for contract negotiation, however the lack of an *offer* construct like ODRL makes this less appealing.

4.2.1 Add, Remove and Replace

As in our ODRL extension, the Replace element is a compound of the add and remove elements. The Add and Remove elements both extend the *RequestDetails* type, which comprise of a number of grants and grantGroups. These are essentially the same constructs that appear in the License element in XrML 2.0. The grant group is a container of several grants but it does not convey any associations between the grants it contains [1]. Figure 10 shows the model for the *ReplaceRequest* model, together with the remove and add components.

4.2.2 RequestBase

To reduce repetition, we first introduce the *RequestBase* type. This type is very similar to the *License* element. In addition to the add, remove and replace elements, this type contains the inventory of the grant elements, an optional title of the request, an optional description, as well as a mechanism to cater for other extensions. The *encryptedLicense* element has the same functionality as the *encryptedLicense* element in the License element in the XrML specifications. Figure 11 shows the content model for the RequestBase element.

4.2.3 Request

The *Request* type extends the RequestBase type. We have added the *requestor* principal, which would allow the rights holder to identify the principal making the request. In the XrML specifications, the *licenseID* was an attribute of the License element. However in a request mechanism, there is the possibility of catering for multiple IDs (e.g. requesting a change to a change granted through a grant-request). For this reason, we decided to incorporate *licenseIDs* as an element of the type. Figure 12 shows the content model of the request type. Please note that RequestBase has been compacted, and you can see figure 11 for the details of RequestBase. The *request* element is the only element of the Request type.

4.2.4 GrantDenyRequest

The *GrantDenyRequest* type creates an envelope for the rights holder to respond back to the user making the request. There are two differences between this type and the Request type. Firstly, a new element, issuer is added to enable the rights

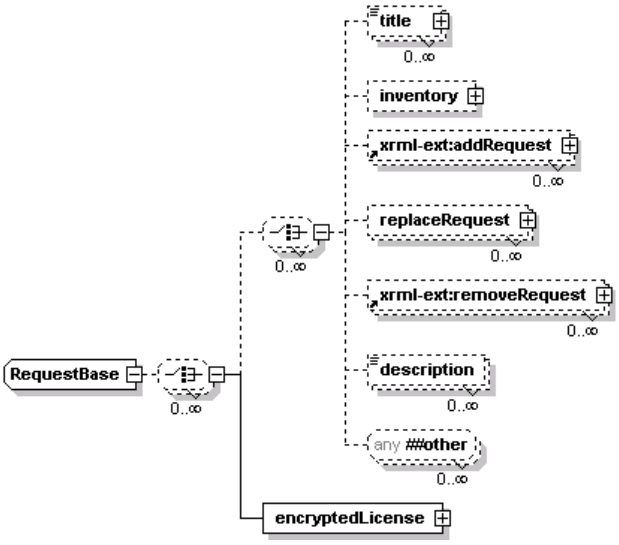

Figure 11: XrML Extension: RequestBase Type Model

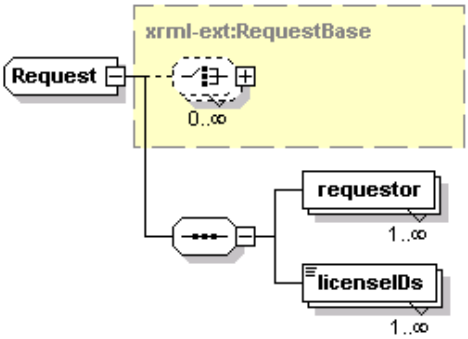

Figure 12: XrML Extension: Request Type Model

holders to positively identify themselves. Each response to a request must have at least one issuer. Secondly, the response must have an identity tag (*LicenseID*) to identify the response. Figure 13 shows the content model of the GrantDenyRequest type. Please note that RequestBase has been compacted, and you can see figure 11 for the details of RequestBase. The *grant-request* and *deny-request* elements are the only elements of the GrantDenyRequest type.

4.2.5 Examples

Alice Smith has bought an e-book "The Art of War" from the XYZ Books Company. The vendor allows Alice to lend the e-book to any number

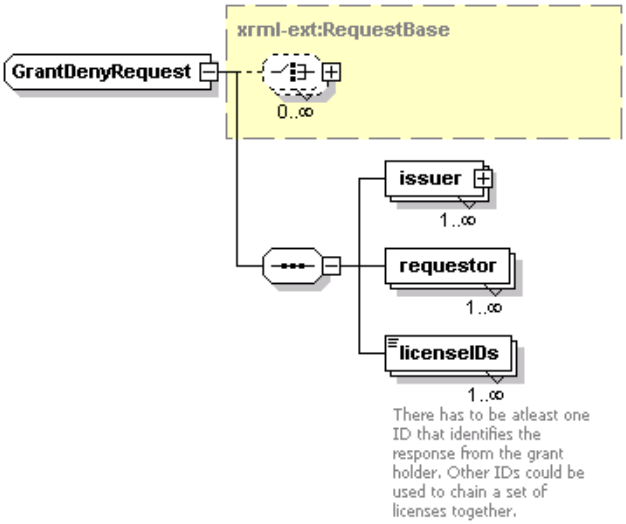

Figure 13: XrML Extension: GrantDenyRequest Type Model

of persons. However Alice does not make use of this option, and would rather have another permission in its place. XYZ Books does not allow Alice to print any copies of the book, and Alice would like to swap her "lend" permission for unlimited printing permission. Example 3 shows Alice's request. Example 4 shows the response from the rights holder denying the request. A grant-request response would have been identical except for the root element.

Note, that for the sake of clarity we have left the namespace definitions together with their respective schema locations and the contents of the encryption key fields (like Modulus of a RSA key) out of the example. The description of the namespaces are detailed below.

xrml-ext: The extended XrML schema as discussed in this section

xrmlcore: The *Core Schema* of the XrML 2.0 specification.

cx: The *Content Extension Schema* of the XrML 2.0 specification.

dsig: W3C schema for XML Schemas

```
<xrml-ext:request>
  <xrml-ext:title>This is a Request Exampl
e</xrml-ext:title>
  <xrml-ext:inventory>
<xrmlcore:keyHolder xrmlcore:licensePartId
="Alice">
      <xrmlcore:info>
        <dsig:KeyValue>
          <dsig:RSAKeyValue>
            <dsig:Modulus/>
            <dsig:Exponent/>
          </dsig:RSAKeyValue>
        </dsig:KeyValue>
      </xrmlcore:info>
    </xrmlcore:keyHolder>
    <cx:digitalWork xrmlcore:licensePartId
="book1">
      <cx:metadata><xml>
          <cx:title>The Art of War</cx:tit
le>
      </xml></cx:metadata>
    </cx:digitalWork>
  </xrml-ext:inventory>
  <xrml-ext:replaceRequest>
    <xrml-ext:removeRequest>
      <xrmlcore:grant>
        <cx:loan/>
      </xrmlcore:grant>
    </xrml-ext:removeRequest>
    <xrml-ext:addRequest>
      <xrmlcore:grant>
        <cx:print/>
      </xrmlcore:grant>
    </xrml-ext:addRequest>
  </xrml-ext:replaceRequest>
  <xrml-ext:requestor xrmlcore:licensePart
Id="Alice"/>
  <xrml-ext:licenseIDs>license://books.xyz
.issue.alice.smith.1</xrml-ext:licenseIDs>
</xrml-ext:request>
```

Example 3: XrML Request

```
<xrml-ext:deny-request
  <xrml-ext:title>This is a Deny Request
Example</xrml-ext:title>
  <xrml-ext:inventory>
    <xrmlcore:keyHolder xrmlcore:licensePa
rtId="Alice">
      <xrmlcore:info>
        <dsig:KeyValue>
          <dsig:RSAKeyValue>
            <dsig:Modulus/>
            <dsig:Exponent/>
          </dsig:RSAKeyValue>
        </dsig:KeyValue>
      </xrmlcore:info>
    </xrmlcore:keyHolder>
    <cx:digitalWork xrmlcore:licensePartId
="book1">
      <cx:metadata>
        <xml>
          <cx:title>The Art of War</cx:tit
le>
        </xml>
      </cx:metadata>
    </cx:digitalWork>
  </xrml-ext:inventory>
  <xrml-ext:replaceRequest>
    <xrml-ext:removeRequest>
      <xrmlcore:grant>
        <cx:loan/>
      </xrmlcore:grant>
    </xrml-ext:removeRequest>
    <xrml-ext:addRequest>
      <xrmlcore:grant>
        <cx:print/>
      </xrmlcore:grant>
    </xrml-ext:addRequest>
  </xrml-ext:replaceRequest>
  <xrml-ext:issuer>
    <dsig:Signature>
      <dsig:SignedInfo>
       <dsig:CanonicalizationMethod dsig:
Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
       <dsig:SignatureMethod dsig:Algorith
m="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
```

```
      <dsig:Reference>
        <dsig:Transforms>
          <dsig:Transform dsig:Algorithm=
"http://www.xrml.org/schema/2001/11/xrml2c
ore#license"/>
        </dsig:Transforms>
       <dsig:DigestMethod dsig:Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
        <dsig:DigestValue/>
      </dsig:Reference>
     </dsig:SignedInfo>
     <dsig:SignatureValue/>
     <dsig:KeyInfo>
       <dsig:KeyValue>
         <dsig:RSAKeyValue>
           <dsig:Modulus/>
           <dsig:Exponent/>
         </dsig:RSAKeyValue>
       </dsig:KeyValue>
     </dsig:KeyInfo>
   </dsig:Signature>
   <xrmlcore:details>
     <xrmlcore:timeOfIssue>
        2004-05-28T19:00:00</xrmlcore:tim
eOfIssue>
   </xrmlcore:details>
 </xrml-ext:issuer>
 <xrml-ext:requestor xrmlcore:licensePart
Id="Alice"/>

 <xrml-ext:licenseIDs>
   license://books.xyz.issue.alice.smith.
1
 </xrml-ext:licenseIDs>
 <xrml-ext:licenseIDs>
   license://books.xyz.denyrequest.alice.
smith.1
 </xrml-ext:licenseIDs>
</xrml-ext:deny-request>
```

Example 4: XrML Deny Request

4.2.6 Full Listing

A full listing of the schema definition is available in appendix B.

5 Future Work

As pointed out by Mulligan et al. [8], bi-directional communication does not depend on REL support only. The protocols used by the DRM systems and the DRM controllers need to be modified. License servers could also be setup to grant or deny certain requests automatically. In the broader scheme, bi-directional REL forms a core part of our proposal to create an open right management services framework [5], and will hopefully overcome many of the current obstacles in DRM systems.

6 Conclusions

In this paper we discussed extensions to XrML and ODRL that creates a request mechanism. It allows users to request for changes in rights, permissions or even resources (and their combinations) to their existing license. The extensions also allow the rights holders to respond to these requests through a grant and deny request mechanisms. By extending the XML schema, we have not broken the existing standard; and thus allows for full backward compatibility. We believe that the request feedback mechanism would allow for easier rights management through better contract negotiation, and would also allow for users to request (and be subsequently granted) fair use rights that might not necessarily hold for everyone[1].

7 Acknowledgements

This work is supported through grants from the UCT Council, the KW Johnstone and Daimler-Chrysler scholarships. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of UCT or the trustees

[1] For example, it is generally considered fair use to reproduce a section of text for academic purposes as long as there is no monetary gain. In current DRM systems, this right is hard to express unless it is known that the user is an academic before the sale. This would be easy to implement using a request mechanism where an academic could buy the text as normal, and then request to reproduce a section; and prove to the rights holders that he/she is an academic.

of the UCT Council, KW Johnstone and Daimler-Chrysler scholarships.

XML Schema content model diagrams were generated using XMLspy.

This report is based on a paper that was originally submitted to the ACM Workshop on Digital Rights Management. While the paper was not accepted, the reviewers gave a lot of constructive criticisms, and many of the recomendations and criticisms have been incorporated and addressed in this report. We would like the thank the reviewers on their detailed reviews.

References

[1] *eXtensible rights Markup Language (XrML) 2.0 Specification*, 2001.

[2] *Open Digital Rights Language (ODRL) 1.1*, 2002.
URL: http://odrl.net/1.1/ODRL-11.pdf.

[3] *Specification: WS-Security Profile for XML-based Tokens*, 2002.
URL: http://www-106.ibm.com/developerworks/webservices/library/ws-sectoken.html.

[4] Technical overview of windows rights management services for windows server 2003. White paper, Microsoft, 2003.

[5] Arnab, A., and Hutchison, A. Distributed drm system. 2004.

[6] Coyle, K. Right Expression Languages, A report for the Library of Congress. Tech. rep., Library of Congress, USA, 2004.

[7] Felten, E. Skeptical view of DRM and Fair Use. *Communications of the ACM 46*, 4 (2003), 57–59.

[8] Mulligan, D., and Burstein, A. Implementing Copyright Limitations in Right Expression Languages. In *Proceedings of the 2002 ACM workshop on Digital Rights Management* (2002), ACM.

[9] Park, J., Sandhu, R., and Schifalacqua, J. Security architectures for controlled digital information dissemination. In *Proceedings of the 16th Annual Computer Security Applications Conference* (2000).

[10] Wenning, R. DRM and the Web. In *ODRL International Workshop 2004, Vienna Austria* (2004).
URL: http://www.w3.org/Talks/2004/04-odrl/.

8 Appendix

In the following two appendices, we provide a full source listing of the two extended schemas. Due to space constraints, indentation has been reduced.

A Full Listing – Extended ODRL Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://people.
cs.uct.ac.za/~aarnab-ODRL"
elementFormDefault="qualified"
attributeFormDefault="qualified" version="
0.1" xmlns:odrl-ext="http://people.cs.uct.
ac.za/~aarnab-ODRL"
xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
>

<xs:import namespace="http://odrl.net/1.1/
ODRL-EX" schemaLocation="http://www.odrl.n
et/1.1/ODRL-EX-11.xsd"/>

<xs:annotation>
 <xs:documentation>
   XML Schema extends ODRL Expression Lang
   uage Schema by allowing users/distribut
   ers to requ-est rights from theright ho
   lder.

   Alapan Arnab
   Validated with XMLSpy 2004
```

```
 </xs:documentation>
</xs:annotation>


<xs:element name="rights" type="odrl-ext:
rightsType"/>

<!-- Add the query element to the language
-->
<xs:element name="request" type="odrl-ext:
requestType"/>
<xs:element name="grant-request" type="odr
l-ext:responseRequestType"/>
<xs:element name="deny-request" type="odrl
-ext:responseRequestType"/>


<!-- The request type comprises of a numbe
r of addition, replace and remove requests
. These requests themselves are of the off
erAgreeType.
-->

<xs:complexType name="requestType">
 <xs:choice minOccurs="0" maxOccurs="unbou
nded">
  <xs:element ref="o-ex:context" minOccurs
="0"
   maxOccurs="unbounded"/>
  <xs:element ref="odrl-ext:request-add"
   minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="request-replace"
   type="odrl-ext:requestReplaceType"
   minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="odrl-ext:request-remove
"
   minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="o-ex:party"
   maxOccurs="unbounded"/>
  <xs:element name="description" type="xs:
string"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:choice>
</xs:complexType>

<!-- A grant/deny request should have the
information about the request its granting
, the license number/context information
of the original request and license.contex
t information about the new license.-->

<xs:complexType name="responseRequestType"
>
 <xs:complexContent>
  <xs:restriction base="odrl-ext:requestTy
pe">
   <xs:choice minOccurs="0" maxOccurs=
    "unbounded">
    <xs:element ref="o-ex:context"
     maxOccurs="unbounded"/>
    <xs:element ref="odrl-ext:request-add"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="request-replace"
     type="odrl-ext:requestReplaceType"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="odrl-ext:request-remo
ve"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="o-ex:party" minOccurs
="2"
     maxOccurs="unbounded"/>
    <xs:element name="description"
     type="xs:string" minOccurs="0"
     maxOccurs="unbounded"/>
   </xs:choice>
  </xs:restriction>
 </xs:complexContent>
</xs:complexType>

<!-- Allows for a multiple number of tuppl
es for replacement.-->

<xs:complexType name="requestReplaceType">
 <xs:sequence minOccurs="0"
  maxOccurs="unbounded">
  <xs:element ref="odrl-ext:request-remove
"/>
  <xs:element ref="odrl-ext:request-add"/>
 </xs:sequence>
</xs:complexType>
```

```
<xs:element name="request-add"
 type="o-ex:offerAgreeType"/>

<xs:element name="request-remove"
 type="o-ex:offerAgreeType"/>

<!-- The rightType container. Added the re
quest container. -->

<xs:complexType name="rightsType">
 <xs:complexContent>
  <xs:extension base="o-ex:rightsType">
   <xs:choice minOccurs="0"
    maxOccurs="unbounded">
    <xs:element ref="odrl-ext:request"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="odrl-ext:grant-reques
t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="odrl-ext:deny-request
" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
</xs:schema>
```

B Full Listing – Extended XRML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://people.
cs.uct.ac.za/~aarnab-XRML"
 elementFormDefault="qualified"
 attributeFormDefault="qualified"
 version="0.1"
 xmlns:xrml-ext="http://people.cs.uct.ac.
za/~aarnab-XRML"
 xmlns:xs="http://www.w3.org/2001/XMLSchem
a" xmlns:xrmlcore="http://www.xrml.org/sch
ema/2001/11/xrml2core" xmlns:ns1="http://w
ww.w3.org/XML/1998/namespace">

<xs:import namespace="http://www.w3.org/XM
L/1998/namespace" schemaLocation="http://w
ww.w3.org/2001/xml.xsd"/>

<xs:import namespace="http://www.xrml.org/
schema/2001/11/xrml2core" schemaLocation="
xrml2core.xsd"/>

<xs:annotation>
 <xs:documentation>
  XML Schema extends XRML Expression Langu
  age Schema by allowing users/distributer
  s to request rights from the right holde
  r.

  Alapan Arnab
  Validated with XMLSpy 2004
 </xs:documentation>
</xs:annotation>

<xs:element name="request" type="xrml-ext:
Request"/>
<xs:element name="grant-request"
 type="xrml-ext:GrantDenyRequest"/>
<xs:element name="deny-request"
 type="xrml-ext:GrantDenyRequest"/>

<xs:complexType name="Request">
 <xs:complexContent>
  <xs:extension base="xrml-ext:RequestBase
">
   <xs:sequence>
    <xs:element name="requestor"
     type="xrmlcore:Principal"
     maxOccurs="unbounded"/>
    <xs:element name="licenseIDs"
     type="xs:anyURI" maxOccurs="unbounded
"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>

<xs:complexType name="GrantDenyRequest">
 <xs:complexContent>
  <xs:extension base="xrml-ext:RequestBase
">
   <xs:sequence>
    <xs:element name="issuer"
     type="xrmlcore:Issuer"
```

```
     maxOccurs="unbounded"/>
    <xs:element name="requestor"
     type="xrmlcore:Principal"
     maxOccurs="unbounded"/>
    <xs:element name="licenseIDs"
     type="xs:anyURI" maxOccurs="unbounded
">
     <xs:annotation>
      <xs:documentation>
       There has to be atleast one ID that
       identifies the response from the gr
       ant holder. Other IDs could be used
       to chain a set of licenses together
       .
      </xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>

<!--Definition of the Request Base Type-->

<xs:complexType name="RequestBase">
 <xs:choice minOccurs="0" maxOccurs="unbou
nded">
  <xs:choice minOccurs="0" maxOccurs="unbo
unded">
   <xs:element name="title"
    type="xrmlcore:LinguisticString"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="inventory"
    type="xrmlcore:Inventory" minOccurs="0
"/>
   <xs:element ref="xrml-ext:addRequest"
minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="replaceRequest"
    type="xrml-ext:ReplaceRequest" minOccu
rs="0" maxOccurs="unbounded"/>
   <xs:element ref="xrml-ext:removeRequest
" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="description"
    type="xs:string" minOccurs="0"
    maxOccurs="unbounded"/>
   <xs:any namespace="##other"
    processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:choice>
  <xs:element name="encryptedLicense"
   type="xrmlcore:EncryptedContent"/>
 </xs:choice>
</xs:complexType>

<!-- Definitions of the addRequest and
 removeRequest elements -->

<xs:element name="addRequest"
 type="xrml-ext:RequestDetails"/>
<xs:element name="removeRequest"
 type="xrml-ext:RequestDetails"/>

<!-- Definition of the ReplaceRequest comp
lex type -->

<xs:complexType name="ReplaceRequest">
 <xs:sequence minOccurs="0"
  maxOccurs="unbounded">
  <xs:element ref="xrml-ext:removeRequest"
/>
  <xs:element ref="xrml-ext:addRequest"/>
 </xs:sequence>
</xs:complexType>

<!-- Definition of the RequestDetails comp
lex type-->

<xs:complexType name="RequestDetails">
 <xs:annotation>
  <xs:documentation>
   A request detail contains the details a
   s specified by the Grant and GrantGroup
   types in the XrML core schema. The inve
   ntory, issuer details are global.
  </xs:documentation>
 </xs:annotation>
 <xs:choice minOccurs="0" maxOccurs="unbou
nded">
  <xs:element ref="xrmlcore:grant"/>
  <xs:element ref="xrmlcore:grantGroup"/>
 </xs:choice>
</xs:complexType>
```

```
</xs:schema>
```