Extending ODRL to Enable Bi-Directional Communication

Alapan Arnab and Andrew Hutchison
Data Network Architectures Group
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
{aarnab, 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 changes in their rights contracts. In this paper we propose extensions to ODRL that allow end users to request changes and for the rights holder to grant or deny these changes. These extensions allow the end user to request changes to their current rights, and for the rights holder to grant or refuse the request. We also provide two examples to demonstrate possible uses of our extensions. The extensions we discuss can also be implemented in other RELs like XrML.**

I. Introduction

Rights Expression Languages (RELs), like Open Digital Rights Language (ODRL) and eXtensible rights Markup Language (XrML), form an integral part of a DRM system because they allow the rights holders to express the terms and conditions which need to be upheld by DRM systems. Most RELs have an extensive vocabulary, supporting syntactic rules that allow them to express a variety of different terms and conditions. Thus RELs allow for greater flexibility in the expression of rights from the view of the rights holders.

However, RELs have also been criticised for giving rights holders too much control, and thus the flexibility offered by RELs empower only the rights holders and not the end users. 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, ODRL has been criticised by some for the absence of a "*not*" semantic [7], which prevents rights holders from expressing a use license like "allow user B all rights except right A".

RELs allow for the expression of digital contracts, even though some, like Felten in [4], have argued that the RELs are unsuitable for expressing legal rights. However, contracts are usually negotiated between two parties, and true contracts require parties to communicate [5]. 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*" [5]. Mulligan et al. concluded that a REL allowing bi-directional communication as well as *rights messaging protocols (RMP)* that support contract negotiations are essential in future DRM systems.

The problem with the current system can be best represented using an example from the second scenario in Microsoft's overview of RMS [2]. 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 has some major drawbacks, like:

1) If the document in question is very big (presentation files for example can easily be over 50Mb in size), it may become impractical for Tom to redistribute the document every time rights need to be changed. Even with broadband Internet, many mail servers for example do not allow large attachments.
2) Tom could be out of the office, and thus may not necessarily be in a position to handle rights changes. If there are automated license servers, bi-directional RELs could allow end users to request for changes without the intervention of the rights holders.

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. Furthermore, a bi-directional REL potentially allows for upgrades to a use license after the initial issuing without the need to change the DRM controller or redistribute the protected data.

With bi-directional RELs it would also be possible to cater for fair use at a general level – rights holders can issue use licenses with usage rules fair for the 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.

Electronic negotiation can be represented in a layered model as shown in figure 1. The users are involved in a *transaction*,

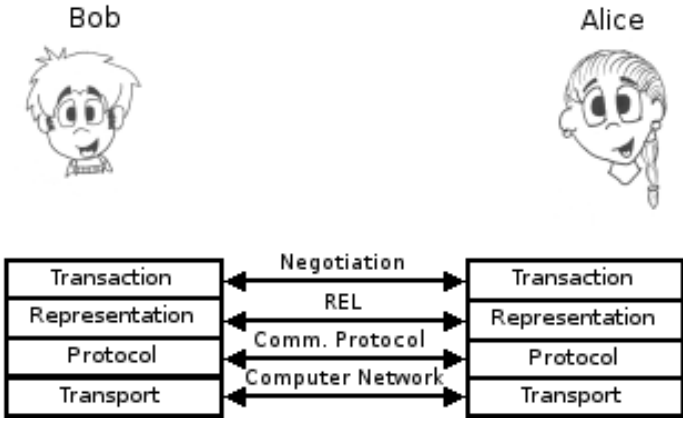

Fig. 1. Layered view of electronic negotiation

and in the case of contract negotiation, the contract will be a human readable contract. The contract is *represented* in a machine readable language, like ODRL. The negotiation takes place using a *communication protocol* over a *computer network*. Ideally, these layers should be independent, and thus the communication protocol should be separated from the REL. For this reason, this paper focuses on the ODRL extensions required to allow ODRL to express negotiations, and does not discuss the details of various negotiation protocols that could be used.

In this paper, we introduce vocabulary and syntax to facilitate bi-directional communication in ODRL. We motivate our design, detail individual elements and then provide two examples showing how a bi-directional ODRL can be used. We also detail a scenario, with examples, demonstrating the use of our extended language as a means for enabling fair use. Similar vocabulary and syntax can also apply to other RELs like XrML.

II. Design Motivations

In 2000, Park et al. [6] 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 [6]. 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.

To fully exploit the power of a bi-directional REL, the DRM system must allow for changes to be made to the protected work after distribution has taken place. Thus the DRM controller must be able to enforce all three types of control sets, and be able to handle use licenses that allow for rights previously disallowed.

It is true that any number of mechanisms can be used to express communication from the user to the rights holders. However, if the expression is not made in the language used by the rights holders to express rights, there will be a need to translate from the users' needs to the appropriate REL. Translation can be an expensive process, and can lead to ambiguities and inaccuracies. Thus having bi-directional support in a REL allows for the possibility of a standardised mechanism to express the needs of the end users.

In our design we envisage a bi-directional system to be implemented as a web-service. Thus a user would *request* 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 any REL. 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. While this feature is most probably not going to be in big demand, it could be used to strip down undesired or unused permissions. The remove feature is also 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 dissimilar 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.

Lastly, we believe that the bi-directional extensions makes ODRL more *complete*. Current ODRL specifications allow for two types of licenses – an *offer* and an *agreement*. With an offer, the rights holders are allowed to express the rights that they are willing to offer to the end user. If the end user accepts, the rights holders can then create an agreement. With our extensions, it is now possible for the end user to have a more active part in generating the agreement, and thus allow for flexibility for the user.

In the following section, we discuss the details of our extensions.

III. ODRL-EXT: BI-DIRECTIONAL EXTENSIONS TO ODRL

Our extension adds three more entities – request, grant-request and deny-request – and are modelled on the agreement entity. We envisage its main use as being in a web-services environment and can be described in four easy steps. The end-user can request the rights holder for a set of rights on a set of assets. The rights holder can then evaluate the request, and then deny or grant that request. The user can accept the decision or carry on negotiating by refining his/her requests. This process is shown in figure 2

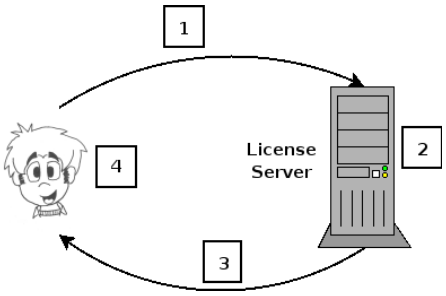

Fig. 2. Negotiating a use license

This model can be further extended where the rights holder can offer various rights at various prices. The prospective end user can then request 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.

A. Add, Remove and Replace

The add, remove and replace requests are the base elements of our extensions. A user can request a combination of these requests, and similarly the rights holders can grant or deny the

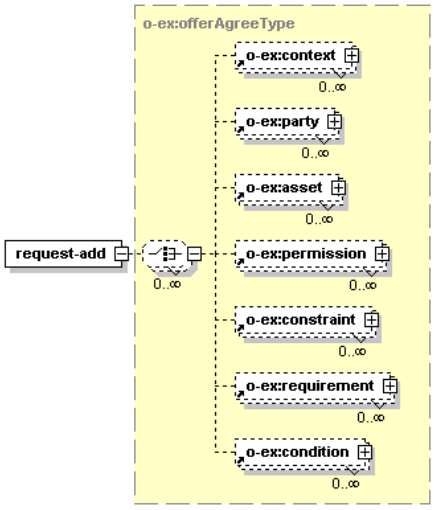

Fig. 3. The Add Request Content Model

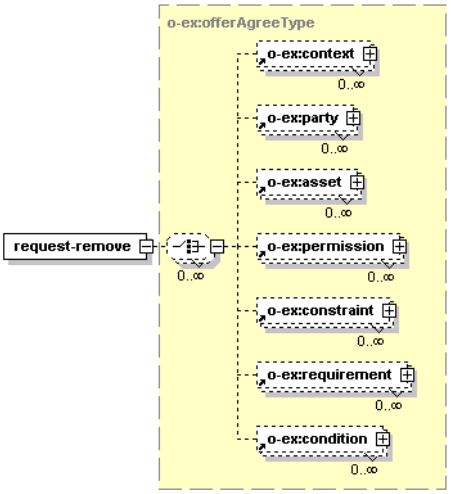

Fig. 4. The Remove Request Content Model

combination of the requests. For maximum flexibility, every element of a ODRL license agreement should be negotiable – permissions, constraints, requirements, conditions, assets and even the parties. For this reason, add, replace and remove elements are simply instances of the offerAgreeType in the ODRL Expression Language Schema [1]. Using the offerAgreeType also minimises ambiguity during negotiations, as the exact rights can be transfered to the "offer" license and eventually the "agreement" license.

The *replace-request* element comprises of a set of remove requests followed by a set of add requests. Although a replace-

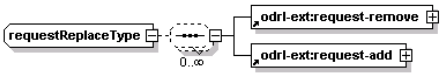

Fig. 5. The Replace Request Content Model

request element is not necessary, we believe that this element would allow for better tracking and management by the rights holders. This would also allow for automation of license servers, where the rights holders can write different rules on which combinations of replace requests they would allow. Figures 3,4 and 5 show the content model for the add, remove and replace elements.

B. Request

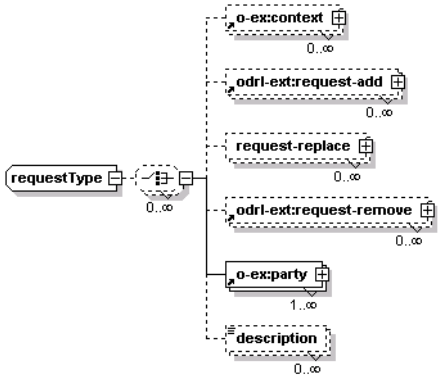

Fig. 6. The Request Content Model

The user communicates to the rights holders through a series of requests. The request element is the only element of the *requestType*. The requestType type, 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 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 6 shows the content model of the requestType.

C. Request Response

The *requestResponseType* creates an envelope for the rights holders to respond back to the user making the request.

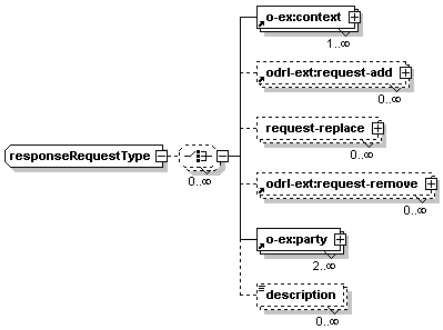

Fig. 7. The Response-Request Content Model

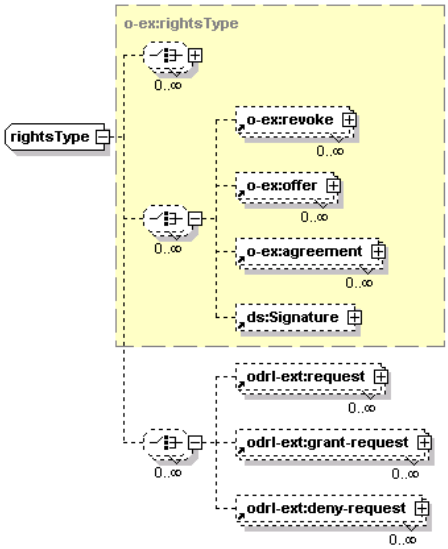

Fig. 8. The rightsType Content Model

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. Secondly the response must have at least two parties - one identifying the user who made the request and another to identify the rights holder. Figure 7 shows the content model of the requestResponse type. The rights holders can respond to a request from the end user in two ways – they can either grant or deny the requests, and thus the grant and deny request elements are of the requestResponseType.

D. rightsType

In ODRL 1.1 the *rightsType* complex type encapsulates agreements and offers with a digital signature and a revoke mechanism [1]. 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 8 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.

We recognise that these extensions could also be encapsulated in a new type (for example *negotiationType*) leaving the existing *rightsType* type alone. If this approach is taken, it would also need a digital signature and a revoke mechanism and we think that our current approach is more elegant as it avoids duplication of common functions.

E. Examples

In “Ebook Scenario #2” of the ODRL 1.1 specifications, a consumer (Mary Smith) purchases an ebook “Why Cats Sleep and We Don’t” [1]. The use license restricts consumers to a single CPU and allows them to print the book at most two times.

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 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.

F. Full Listing

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

```
<odrl-ext:rights>
  <odrl-ext:request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/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:na
me>
      </o-ex:context>
    </o-ex:party>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 1: Simple ODRL Request

```
<odrl-ext:rights>
  <odrl-ext:grant-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-GHIJKL</o-dd:uid>
    </o-ex:context>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/99999/
license/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/99999
9/users/msmth-000111</o-dd:uid>
        <o-dd:name>Mary Smith</o-dd:na
me>
      </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

IV. Extended Example

Examples 1 and 2 used a simple scenario to demonstrate the use of our proposed extensions. In this section, we detail a more complicated scenario (based once again on "Ebook Scenario #2" in [1]) that also demonstrates how our extensions could be used as a means to enable fair use.

In the existing scenario, Mary Smith purchases an ebook "Why Cats Sleep and We Don't" [1]. Users are restricted to a single CPU and print the book at most 2 times (which we extended by another 5 copies in examples 1 and 2). Suppose, Mary Smith is a journalist and wishes to write a thorough review of the ebook and would like to excerpt some of the pictures for this purpose (excerption for the purpose of review is normally considered a fair use right). In this section, we detail the interactions between Mary Smith and the license server for this purpose.

Note, that for the sake of clarity we have 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.

o-dd-ext: An extension of the Data Dictionary Scheme of ODRL 1.1 to allow representation of credentials (discussed in sections IV-B and V).

A. Initial Request

Mary Smith wishes to excerpt 3 pictures from different pages in the ebook, the first picture in page 3 while the last picture is in page 56 (about half way through the book).

```
<odrl-ext:rights>
  <odrl-ext:request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF</o-dd:uid>
    </o-ex:context>
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:excerpt>
          <o-ex:constraint>
            <o-dd:range>
              <o-dd:min>3</o-dd:min>
              <o-dd:max>56</o-dd:max>
            </o-dd:range>
          </o-ex:constraint>
        </o-dd:excerpt>
      </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>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 3: Extended Example – Request 1

B. Initial Rejection and Counter Offer

Excerption is a fair use, but is usually limited to a percentage of a work. The license server rejects Mary Smith's request with an explanation, but also offers a counter offer that could be used by Mary Smith. This counter offer makes use of a *credential* constraint not present in the standard ODRL data dictionary. The counter offer is given as a *grant-request*, although it could also be expressed as an *offer*.

```
<odrl-ext:rights>
  <odrl-ext:deny-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/TRANS-0101</o-dd:uid>
    </o-ex:context>
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:excerpt>
          <o-ex:constraint>
            <o-dd:range>
              <o-dd:min>3</o-dd:min>
              <o-dd:max>56</o-dd:max>
            </o-dd:range>
          </o-ex:constraint>
        </o-dd:excerpt>
      </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>
    <odrl-ext:description>
Excerption is only available with an
academic, scholar or journalist
credential. Furthermore, a maximum of
10% of the total protected work can be
excerpted
    </odrl-ext:description>
</odrl-ext:deny-request>
</odrl-ext:rights>
```

Example 4: Extended Example – Response 1, the denial of request

```
<odrl-ext:rights>
  <odrl-ext:grant-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF</o-dd:uid>
    </o-ex:context>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF-01</o-dd:uid>
    </o-ex:context>
```

Example continued over the page

```
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:excerpt>
          <o-ex:constraint>
            <o-dd:range>
              <o-dd:min>3</o-dd:min>
              <o-dd:max>13</o-dd:max>
            </o-dd:range>
            <o-dd-ext:credential>
              <o-dd-ext:OrList>
            <o-dd-ext:CredentialsType>
                  Journalist
            </o-dd-ext:CredentialsType>
            <o-dd-ext:CredentialsType>
                  Academic
            </o-dd-ext:CredentialsType>
            <o-dd-ext:CredentialsType>
                  Scholar
            </o-dd-ext:CredentialsType>
              </o-dd-ext:OrList>
            </o-dd-ext:credential>
          </o-ex:constraint>
        </o-dd:excerpt>
      </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>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 5: Extended Example – Response 2, A counter offer

C. Refined Request

Mary Smith decides to refine her request to suit the terms of the license server. She chooses to make a request to excerpt from three different parts of the book but with much smaller page ranges. She also decides to get the license specified for a “Journalist” credential only. The credential would form part of the protocol and not part of the negotiation message, and thus would be represented separately.

```
<odrl-ext:rights>
  <odrl-ext:grant-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF-01</o-dd:uid>
    </o-ex:context>
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:excerpt>
          <o-ex:constraint>
            <o-dd:range>
              <o-dd:min>3</o-dd:min>
              <o-dd:max>4</o-dd:max>
            </o-dd:range>
            <o-dd:range>
              <o-dd:min>16</o-dd:min>
              <o-dd:max>18</o-dd:max>
            </o-dd:range>
            <o-dd:range>
              <o-dd:min>56</o-dd:min>
              <o-dd:max>57</o-dd:max>
            </o-dd:range>
            <o-dd-ext:credential>
              <o-dd-ext:CredentialsTy
pe>
                  Journalist
              </o-dd-ext:CredentialsTy
pe>
            </o-dd-ext:credential>
          </o-ex:constraint>
        </o-dd:excerpt>
      </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>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 6: Extended Example – Request 2, A refined request

D. Accepted Response

The license server accepts Mary Smith’s request and issues a grant request use license.

```
<odrl-ext:rights>
  <odrl-ext:grant-request>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF</o-dd:uid>
    </o-ex:context>
    <o-ex:context>
      <o-dd:uid>urn:ebook.world/999999/
license/1234567890-ABCDEF-01</o-dd:uid>
    </o-ex:context>
    <odrl-ext:request-add>
      <o-ex:permission>
        <o-dd:excerpt>
          <o-ex:constraint>
            <o-dd:range>
              <o-dd:min>3</o-dd:min>
              <o-dd:max>4</o-dd:max>
            </o-dd:range>
            <o-dd:range>
              <o-dd:min>16</o-dd:min>
              <o-dd:max>18</o-dd:max>
            </o-dd:range>
            <o-dd:range>
              <o-dd:min>56</o-dd:min>
              <o-dd:max>57</o-dd:max>
            </o-dd:range>
            <o-dd-ext:credential>
              <o-dd-ext:CredentialsTy
pe>
                  Journalist
              </o-dd-ext:CredentialsTy
pe>
            </o-dd-ext:credential>
          </o-ex:constraint>
        </o-dd:excerpt>
      </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>
  </odrl-ext:request>
</odrl-ext:rights>
```

Example 7: Extended Example – Response 3

V. Future Work

As pointed out by Mulligan et al. [5], 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 to allow for bi-directional communication and for better management of multiple use licenses for the same digital object.

License servers could also be setup to grant or deny certain requests automatically, and thus algorithms are needed to automatically evaluate license templates (ODRL offers) against user requests.

We are currently investigating the use of credentials in DRM, particularly as a mechanism in allowing for fair use (as shown in section IV). Together with a bi-directional REL, we believe that most of the common fair uses can be accommodated in a DRM system.

In the broader scheme, bi-directional REL forms a core part of our proposal to create an open right management services framework [3], and will hopefully overcome many of the current obstacles in DRM systems. A smaller sub-project is currently implementing some of the extensions for DRM controllers mentioned above.

VI. Conclusions

In this paper we discussed extensions to ODRL to allow for bi-directional communication. We discussed our motivation, the concept model, the syntax and semantics of the extensions. Furthermore, we presented examples using existing ODRL scenarios that make use of our extensions. Finally, we discussed how the extensions could be used to allow for fair use with examples drawn from an existing ODRL scenario.

The extensions allow end users to specify any part of a use license including rights, constraints and resources, they would like to have in a use license and rights holders to respond to these requests, thus allowing for negotiations of rights. These extensions complement the existing “offer” and “agreement” license types, and make ODRL more complete.

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. The extensions we have presented can also be implemented in other RELs such as XrML.

Acknowledgements

This work is partially supported through grants from the UCT Council and the National Research Foundation (NRF) of South Africa. Any opinions, findings, and conclusions or recommendations expressed in this paper/report are those of the author(s) and do not necessarily reflect the views of UCT, the NRF or the trustees of the UCT Council.

XML Schema content model diagrams were generated using XMLspy.

Alice and Bob created by Nicholas Hall.

REFERENCES

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

[2] "Technical overview of windows rights management services for windows server 2003," Microsoft," White Paper, 2003.

[3] A. Arnab and A. Hutchison, "Distributed DRM System," University of Cape Town," Departmental Technical Report, No. CS04-27-00, 2004.

[4] E. Felten, "Skeptical view of DRM and Fair Use," *Communications of the ACM*, vol. 46, no. 4, pp. 57–59, 2003.

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

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

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

APPENDIX

In this section, we provide a full source listing of the extended ODRL schema. Due to space constraints, indentation has been reduced.

```
<?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>
```