Department of Information Systems
Vienna University of Economics and Business Administration

Rights Expression Languages:

Characteristics and Applications

*Susanne **Guth***

susanne.guth@wu-wien.ac.at

New Media Lab
http://nm.wu-wien.ac.at/

Agenda

Rights Expression Languages (REL):

- *Requirements*
- *Typical Components*
- *Standards and Initiatives*
- *Application Fields*
- *Market Situation and Trends*

A REL Application:

"Access Control Decisions based on ODRL Contracts"

REL Requirements:

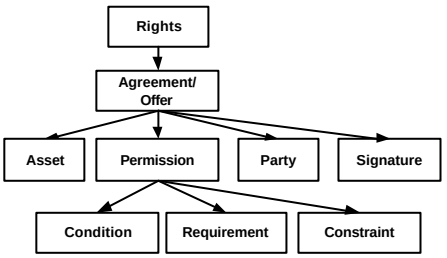
- *express rights of parties over assets/services*
- *being machine readable*
- *formulate business models*
- *articulation of roles*
- *identification mechanisms for resources/parties*
- *express permissions and constraints*
- *express royalties and payment details*
- *security information (digital signature),*
- *etc.*

Official Documents: MPEG 21 REL Requirements

Right Expression Languages: Characteristics and Application

REL Components:

Rights Language Concept:



Rights Data Dictionary:

Name	ID	Description	Comment
Play	play	The act of rendering the asset in audio/video form	...
Print	print	The act of rendering the asset on paper or hard copy form.	...
...	...		...

REL Standards and Initiatives:

- Open Digital Rights Language (**ODRL**) - IPR Systems
- eXtensible rights Markup Language (**XrML**) - Contentguard
- **MPEG 21 Part 5** (REL) & Part 6 (Data Dictionary)
- The **<indecs> 2rdd** Project - Rightscom (→ MPEG 21 Part 6)
- IEEE - **Intellectual Property Rights Framework**: Data Model for Reuse Library Interoperability
- Other initiatives:
  - *eXtensible Media Commerce Language (**XMCL**) - RealNetworks*
  - *DREL - Learning Technology Standards Committee (**LTSC**)*
  - *Digital Property Rights Language (**DPRL**) - Xerox (no further dev.)*
  - *eXtensible Access Control Markup Language (**XACML**) - OASIS*
  - ***"Rights Grammar"** – eBook*
  - *Custom Digital Rights Language (**CDRL**) - Octalis*
  - *Creative Commons*

Right Expression Languages: Characteristics and Application

Rights Expression based on Concepts of Logic Programming

- Nick Szabo, "A Formal Language for Analyzing Contracts", unpublished work, 2002.

  Szabo describes in his work an approach expressing contracts by a programming language rather than by a markup language. The formal language that he describes uses the concepts of logic programming, where licenses are represented by clauses and their processing is described by event-drawn, predefined functions or rules of the language.*

- Sandro & Co ;-) License Script

Differences between XML-based and „rules & facts“-based rights expression

XML-based

- generalizations
- ambiguous
- express contracts!
- need to be implemented
- exchangeable due to standardization (XML-feature) → readable by third party
- …

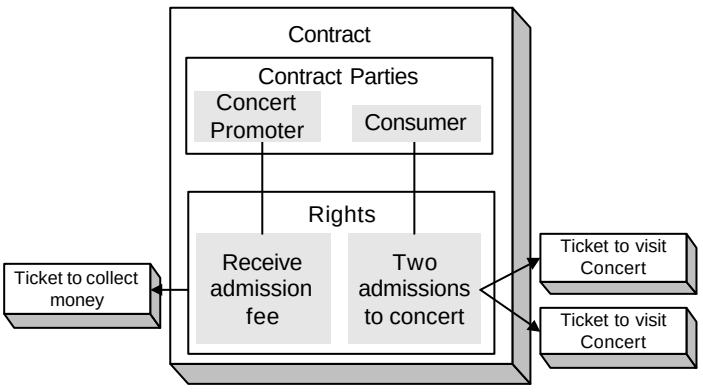
Rules & Facts - based

- fine grained
- unambiguous
- express licenses!?
- direct processing in software systems
- how to exchange rights information between different systems? and make them readable for lawyers or partners?
- …

Let‘s discuss this!

Right Expression Languages: Characteristics and Application

Excurs: Tickets versus Contracts



Susanne Guth, REL Workshop in Twente/Netherlands, September 2003

REL Application Fields:

Access Control:

1. *Rights expressions in licenses of a secure container*
2. *Rights expressions separately from digital content*

other potential applications of digital contracts/rights expressions:

1. *Processing in book keeping/accounting software*
2. *Customer Relationship Management (CRM)*
3. *Disbursement to content providers (IPR)*
4. *...*

Prerequisite for XML-based REL Application:

1. Rights Expression Generator
2. REL Interpreter

Right Expression Languages: Characteristics and Application

REL Current Market Situation & Trends:

XrML

1. ... is used in Microsoft's Media Rights Manager. Secure Viewer: Windows Media Player , Format: *.wma-files
2. ... is used as basis REL for MPEG 21 part 5 (MPEG REL)
3. ... XrML does not experience further development

ODRL

1. ... is used in Nokia & IBM products (mobile services)
2. ... is used as basis REL in OMA (open mobile alliance)
3. ... Version 2.0 is in development and will be released in Summer 2004

Susanne Guth, REL Workshop in Twente/Netherlands, September 2003

A REL Application:

Access Control Decisions based on ODRL contracts.

- *Use Case Description*
- *Tailoring Contracts for Efficient Processing*
  1. General Contract Objects
  2. Required Contract Objects for Access Control
  3. Accordingly fill ODRL template with contract data
- *Required Software Components*
  1. REL Interpreter
  2. Access Control Mechanism
  3. Mediator
- *Contract Processing Steps*

Right Expression Languages: Characteristics and Application

A REL Application:

Use Case Description:

Customer Mary Smith (id ***msmith***) purchases at the e-commerce platform XY the ebook of her favorite author Emiliano Rossi with the id "***rossi-12345***" with the right to ***print*** and ***display***. The right print is restricted to ***2 times***, the right display is restricted to a certain ***cpu-id (Intel-12345).*** The value of receiving these rights is ***AUD 20.00*** with a ***10 %*** tax. The contract is legally concluded in ***Sydney/Australia.***

Mary Smith now would like to conclude a contract and access the ebook. How does this technically work?? ...

Tayloring Contracts for efficient processing:

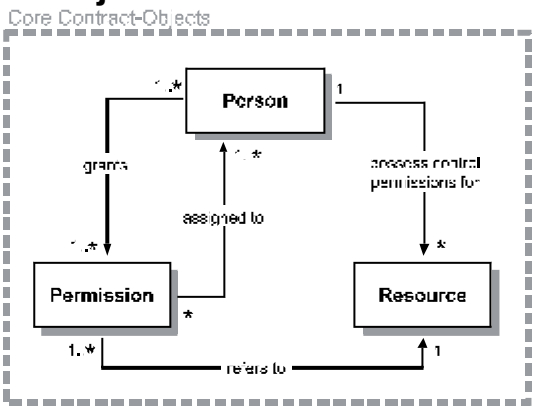
We can‘t use any ODRL contract in sophisticated access control!

- *What application has to be served? (here: Access Control)*
- *What information is needed for the application?*
- *Determine the required information objects and their attributes.*
- *Derive the respective ODRL contract template.*
- *Fill contract template with information.*

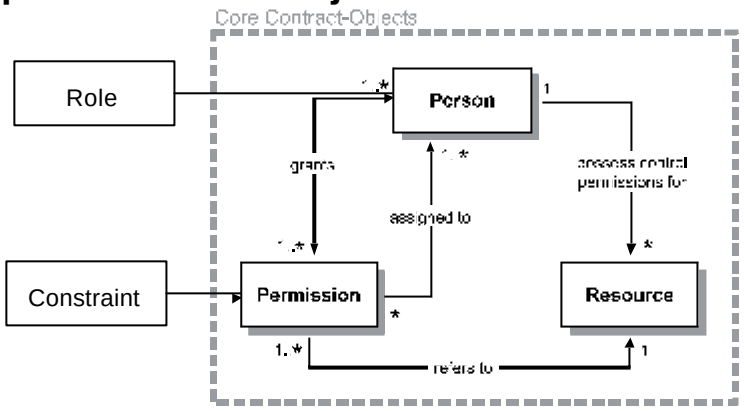
  → Taylored Contract!

Tayloring Contracts:

General Contract Objects



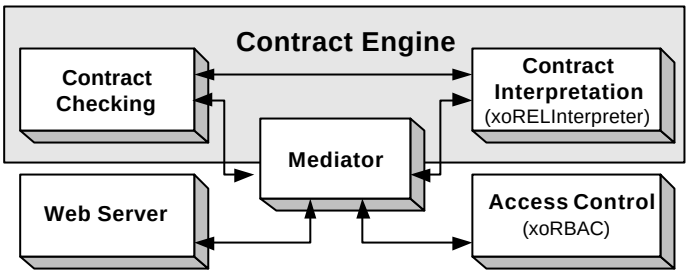
Tailoring Contracts:
Required Contract Objects for Access Control



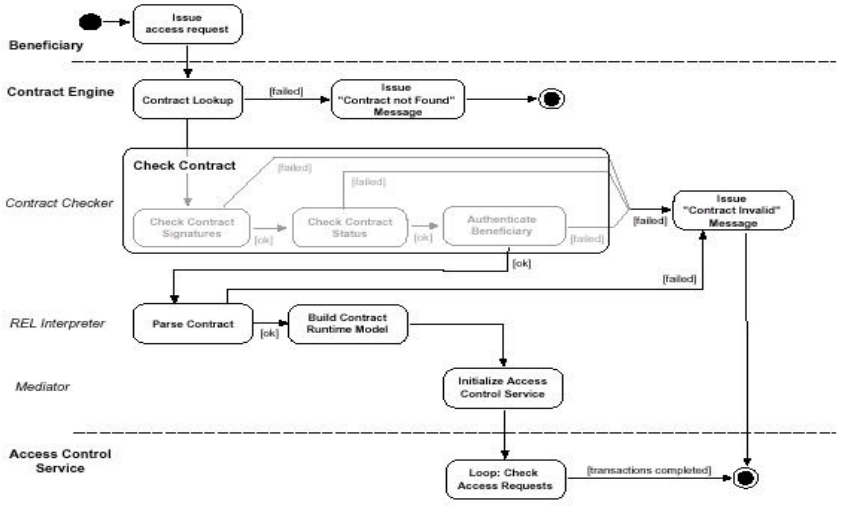
'Tailored ODRL instance

```
<o-ex:rights>
  <o-ex:agreement>
    <o-ex:context>
      <o-dd:uid> license-12345 </o-dd:uid>
      <o-dd:pLocation> Sydney, Australia </o-dd:pLocation>
      <o-dd:remark> Transacted by Example.Com </o-dd:remark>
    </o-ex:context>
    <o-ex:asset>
      <o-ex:context>
        <o-dd:uid> rossi-12345 </o-dd:uid>
      </o-ex:context>
    </o-ex:asset>
    <o-ex:permission>
      <o-dd:display>
        <o-ex:constraint>
          <o-dd:cpu>
            <o-ex:context>
              <o-dd:uid> Intel-12345 </o-dd:uid>
            </o-ex:context>
          </o-dd:cpu>
        </o-ex:constraint>
      </o-dd:display>
      <o-dd:print>
        <o-ex:constraint>
          <o-dd:count> 2 </o-dd:count>
        </o-ex:constraint>
      </o-dd:print>
      <o-ex:requirement>
        <o-dd:prepay>
          <o-dd:payment>
            <o-dd:amount o-dd:currency="AUD">20.00</o-dd:amount>
            <o-dd:taxpercent o-dd:code="GST">10.00</o-dd:taxpercent>
          </o-dd:payment>
        </o-dd:prepay>
      </o-ex:requirement>
    </o-ex:permission>
    <o-ex:party>
      <o-ex:context>
        <o-dd:uid> msmith </o-dd:uid>
        <o-dd:name> Mary Smith </o-dd:name>
      </o-ex:context>
    </o-ex:party>
  </o-ex:agreement>
</o-ex:rights>
```

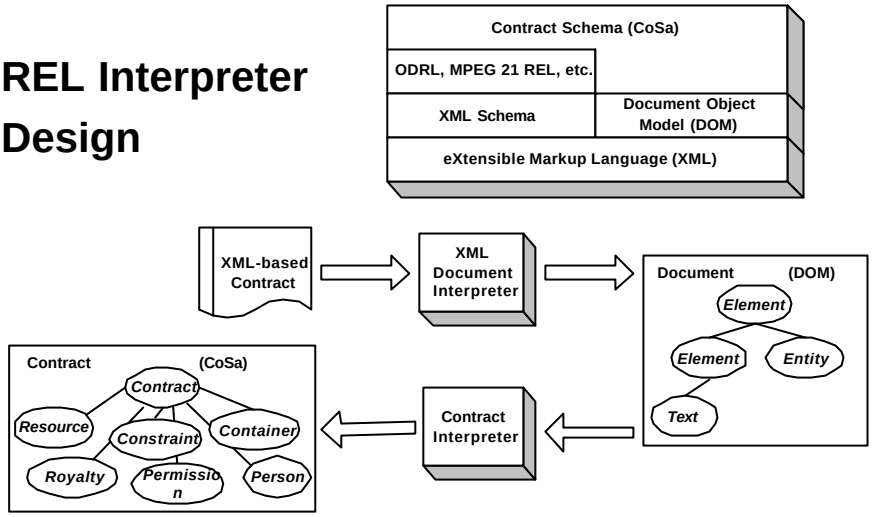
Required Software Components



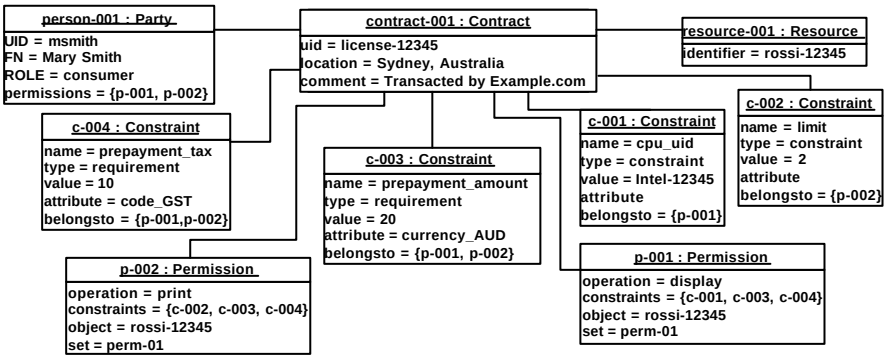
The steps of contract processing:



REL Interpreter Design



CoSa Objects - Runtime Model of the ODLR Contract



CoSa Application Programming Interface

`getContracts():`

Returns a list of all Contract objects registered for the current ODRL instance (an ODRLinterpreter instance can contain more than one contract).

`getAssets(contract):`

Returns a list of all Asset objects,included in a special contract.

`getConsumers(contract):`

Return a list of all Consumer objects,included in a special contract.

`getPermission(consumer):`

Returns a list of all Permission objects,assigned to the respective consumer.

`getUniqueID(object):`

Return the value of the id attribute of the respective object.The respective object could be of any valid CoSa type (e.g.Party, Resource, or Permission).

Mediator:

Combining REL Interpreter and Access Control Mechanism Features

```
requestAccess (subject, operation, object, contract-id) {
  #contract lookup
  set contractInstance [contractLookup(contract-id)]
  #check contract
  set contractOK [checkContract(contract-id]
  #interpret contract and initialize xoRBAC instance "rm"
  ODRLContract cl $contractInstance
  RightsManager rm
  if {$contractOK == true} {
     set contracts [cl getContracts]
     foreach c $contracts {
        set assets [cl getAssets $c]
        set consumers [cl getConsumers $c]
        foreach asset $assets {
           set assetId [cl getUniqueID $asset]
           foreach con $consumers {
              set conID [cl getUniqueID $con]
              rm createSubject $conID
              set perms [cl getPermission $con]
              foreach p $perms {
                 set perm [cl getName $p]
                 rm createPermission $perm
                 rm subjectPermAssign $conID $perm
              }
           }
        }
     }
  } else {
    #contract checks not successfull, access denied.
    return false
  }
  #checkAccess - performed by access control service
  set result [rm checkAccess $subject $operation $object]
  if {$result == 1} {
    return true  #access granted
  } else {
    return false #access denied
  }
}
```

Finish

..... Puh, that was hard stuff!

..... Thanks for listening!

..... Questions?