CLARIN Metadata
Infrastructure

WP2 meeting
Oxford Feb. 26-27
Content

• Thursday
  – General description CLARIN MD Architecture
  – Components & Profiles & Schemas
  – Software component inventory
  – Work distribution (1), possible scenarios

• Friday
  – Work distribution (2)
User selects appropriate components to create a metadata description.

Semantic interoperability partly solved via references to ISOcat concept registry.

Selecting metadata components from the registry.
Perform search/browsing on the metadata catalog using the ISO DCR and other concept registries and CLARIN relation registry

Create metadata schema from selection of existing components. Allow creation of new components if they have references to ISOcat

Metadata harvesting by OAI protocol

Metadata descriptions created

Metadata component profile was selected from metadata component registry
Creating Metadata

ISOcat concept registry

CLARIN component registry

trusted concept registry

user area

component registration

myprofile

Metadata editor

metadata Descriptions

component editor

concept registration
CMD Description

Recursive structure of metadata:
An Actor component can contain a Language component, Contact component etc.

All Metadata elements consist from Name, Value, Scheme AND a concept reference

Possible relations & pointers to Journal files (special feature for workflow systems)
Metadata and Collections

Hierarchy of sub-collections

Easy extension with new collections
Components & Profiles & Schemas

- Component specification XML file.
- General XML Schema generating the components specifications.
- Profile specifying selected components and administrative information
- XML-Schema generating a specific profile
- Metadata descriptions based on a profile
- Template metadata: snippets of frequently used metadata specific for a project. Used by the metadata editor
<xml version="1.0" encoding="UTF-8"?

• <CMD_ComponentSpec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:/mnt/d/sync/doc/clarin/wp2/registry/cmd/cmd-schema.xsd">
  • <CMD_Component ConceptRef="http://iso.org/isocat/09238922" name="ActorLanguage" id="ActorLanguage">
    • <CMD_Element name="Name" ValueScheme="string"/>
    • <CMD_Element name="Id" ValueScheme="string"/>
  • </CMD_Component>
  • <CMD_Component ConceptRef="http://iso.org/isocat/09238923" name="Actor">
    <!-- inline element definitions -->
    • <CMD_Element name="ActorName" ValueScheme="string"
      ConceptRef="http://iso.org/isocat/09238924"/>
    • <CMD_Element name="ActorAge" ConceptRef="http://iso.org/isocat/09238925" CardinalityMin="0"
      CardinalityMax="1">
      • <ValueScheme>
        • <pattern>([0-9]+)*;([0-9]+)*;([0-9]+)</pattern>
      • </ValueScheme>
    • </CMD_Element>
    • <CMD_Element name="ActorSex" ConceptRef="http://iso.org/isocat/09238926" CardinalityMin="0"
      CardinalityMax="1">
      • <ValueScheme>
        • <enumeration>
          • <item ConceptRef="http://iso.org/isocat/09238945">male</item>
        • <item ConceptRef="http://iso.org/isocat/09238946">female</item>
      • </enumeration>
    • </CMD_Element>
    <!-- use element defined elsewhere -->
  • </CMD_Component>
• </CMD_ComponentSpec>
An example MD description

- `<xml version="1.0" encoding="UTF-8"?>`
- `<CMD SelfLink="hdl:1034/73277823" Creator="dvu" Profile="ClarinTestProfile" CreationDate="20090218" Domain="Antropology">`
- `<Actor ConceptRef="http://foo/bar" ComponentRegistryLink="http://bla/bla/Actor">`
- `<Firstname ConceptRef="http://isocat.org/712787812"> Foo </Firstname>`
- `</Actor>`
- `<TextTMD resource="resource1">`
  - `<Format ConceptRef="http://isocat.org/77712">text/plain</Format>`
  `</TextTMD>`
- `<PhotoTMD resource="resource2">`
  - `<Format ConceptRef="http://isocat.org/77712">image/jpeg</Format>`
  `</PhotoTMD>`
- `<ResourceProxy id="resource1">`
  - `<Type>Resource</Type>`
  - `<ResourceRef>http://www.clarin.eu/text.txt</ResourceRef>`
  `</ResourceProxy>`
- `<ResourceProxy id="resource2">`
  - `<Type>Resource</Type>`
  `</ResourceProxy>`

  `<ResourceRelation Type="annotates">`
  - `<res1 ref="resource1"/>`
  - `<res2 ref="resource2"/>`
  `</ResourceRelation>`
- `</CMD>`
Software Components
Schemas & Formats

- Component specification format + schema
- Profile format (not very different from comp spec.)
- Some XSLT style sheets to convert a component selection into a profile schema
- Usable already in the initial phase to populate the component registry
Initial MD Components

• Deconstruct existing metadata sets that CLARIN needs to be backward compatible with: IMDI, OLAC, TEI and describe the resulting components in the CLARIN Component specification formalism

• XSLT style sheets to transform the existing records into component based records
Component Registry

- Store components & profiles
- Offer one or more views that users can browse
  - Based on component metadata: domain, creator, ..
- Import/Export into component specification format
- Maintain administration
  - Every component and registry each have a unique id
  - Log information
Metadata Editor

- Select a metadata profile from the registry, fill in the metadata fields.
- Support for local work-spaces where users can store profiles and components that have been customized with prepared values and that act as templates.
- Offer opportunity to edit many metadata descriptions at a time.
  - show Identical components in table form allowing sorting on specific fields
  - Copy and paste to transfer information between descriptions
- Investigate if the metadata components and profiles should support special attributes to guide the editor GUI. (Help texts, ...)
- Preferably have both a web-based and a standalone editor
Editor should probably accept descriptions belonging to various profiles. But only values from the same component can be shown in a table.

Broeder, 24/02/2009
Metadata Editor

Component & Profile registry
- Comp 1
- Comp n
- Profile A

MD editor
- new descr
- descr 1
- descr 2
- descr 3

Existing descriptions

Actor A
Actor B

Project 1
Project 2
Templates in private workspace
Component Editor

- Creation of new metadata components and profiles
  - based on the component registry
  - and concept registries like the DCR
- New components may be based on (combination of) existing ones or completely new.
- Only requirement is that the metadata elements in the components have references to concepts in an accepted concept registry: DCR, DC, TEI.
- Prevent proliferation of components
  - Registration process before users are allowed to edit components or profiles.
- The DCR can also contain possible relationships between complex data categories and a simple data categories expressing that the one can be the value of the other. However the CLARIN component specification allows the component creator to specify a vocabulary himself since
  - People want to use their own terminology
  - Not all concept registries offer this.
Component Editor

Component & Profile registry

Comp 1

Comp 2

new comp.

Profile A

Profile A

new profile

DCR

genre

DC

title

TEI

reusing old components

use references to concepts

Component Editor

add new elements

user
Metadata harvesting

- OAI PMH (protocol for metadata harvesting) is well established and is considered standard.
- Software is available both for metadata providers service providers (metadata consumers)
- Need a registry that lists all CLARIN metadata providers
  - The registry might be structured (2-tiers) according to the metadata provider’s country
  - That would make it easier to have some national center be responsible for maintaining the list.
  - Such a center could make itself responsible for interacting with a metadata provider if its metadata format is not conforming to the standard.
CLARIN Metadata Repository

- Store harvested metadata descriptions
- Can expect large number of schemas
  - Every profile results in a schema

- Organization; how many repositories?
  - Specialization by domain/topic
  - Organization by nationality
  - Need one that harvests all

- Still open questions:
  - Normalization before storage?
  - Store in native format

- Possible technologies:
  - XML DB
  - RDBMS using object relational mapping

- Provide API for access by other software components
Metadata Services

- Builds on the services provided by the metadata repository
- Metadata queries.
  - What terminology is allowed?
    - From DCR and other concept registries
    - From legacy metadata sets DC, IMDI, OLAC, TEI.
    - From every registered & harvested component
- Makes use of the component, concept and relation registries
Semantic Mapping

• **Purpose**
  – Allow metadata queries to be mapped on metadata components using different terminology.

• **Depending on**
  – the granularity of concepts in the CRs
  – Required return precision
  – Need also relation registry
    • Query: Actor.name = “xxx” will find the records with Actor.fullname = “xxx” and
    • Participant.name = “xxx” by specifying the relations:
      • Participant isA Actor and fullname isA name
Metadata Catalog App.

- Builds on metadata services
- Web application allowing
  - Specify and execute metadata search queries
  - Browsing over all the metadata records harvested from the CLARIN metadata providers.
  - *Function to create views & hierarchies based on user specified selections and metadata queries*
Virtual Collections Registry

• Results from metadata queries or browsing results in a (distributed) virtual collection of resources

• Necessity to make these collections persistent
  – Create metadata record describing the collection
  – Register this record
Portals

• Construct one or several portals that offer the CLARIN metadata services accompanied by suitable documentation.

• Integrated offering of one or more of:
  – Metadata creation
  – Metadata component & profile creation
  – Browsing & Searching harvested CLARIN metadata
  – Virtual collection creation.
  – CLARIN LRT World.
<table>
<thead>
<tr>
<th></th>
<th>Task Description</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overall CLARIN MD Schema</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Component schemas from legacy MD</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Component Registry &amp; Browser</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Concept Relation Registry</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Metadata Editing for &quot;legacy&quot; components</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Component editor</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Metadata Harvesting</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Metadata Repository</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Metadata Query Service</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Metadata Query GUI</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>Semantic mapping metadata queries</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>MD search engine</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>Search results &amp; virtual collection specification</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>Specify &amp; implement APIs</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>Portal solution</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total** 50
Work division

• Only limited resources available
  – To be divided over: MD infra, workflow, AAI
  – Luckily there is also national funding.

• Need to optimize wrt. communication, specification, meetings etc. So small interfaces between the sw components.

• Where to draw boundaries?
  – Natural division between metadata consumer side and the production side

• Identification of advanced functions that can be allotted to other parties: Virtual collections, portal creation, virtual hierarchy creation.
The End

Thank you for your attention