Realistic targets in TEI to RDF

Sebastian Rahtz

TEI members meeting, Wurzburg, 2011-10-13

Context and questions

- What is the relationship of a TEI-encoded text to RDF?
- What is/are our target ontology or ontologies?
- How do get from TEI XML to RDF?
- Where do we maintain mapping information?
- How do we perform transformations?
- Does it work?

Anti-FAQ: questions I am not going to answer

- What is RDF?
- Isn't interchange of texts simply impossible?
- What RDF database and software shall I use?
- What is the *theoretical* background to your work?
- Why doesn't it work with my data?
- Cool people use RDFa, don't they?
- Cool people use microdata, don't they?

Why RDF?

Quite simply, to let us inject our data into the world of the semantic web in a standardized way, and let other people find or assert links.

Our TEI texts are a good archival form, not an interchange format (Discuss)

Background to my work

The CLAROS project based at Oxford aims to combine discrete databases of information about the ancient world using an RDF triplestore of assertions using CIDOC CRM.

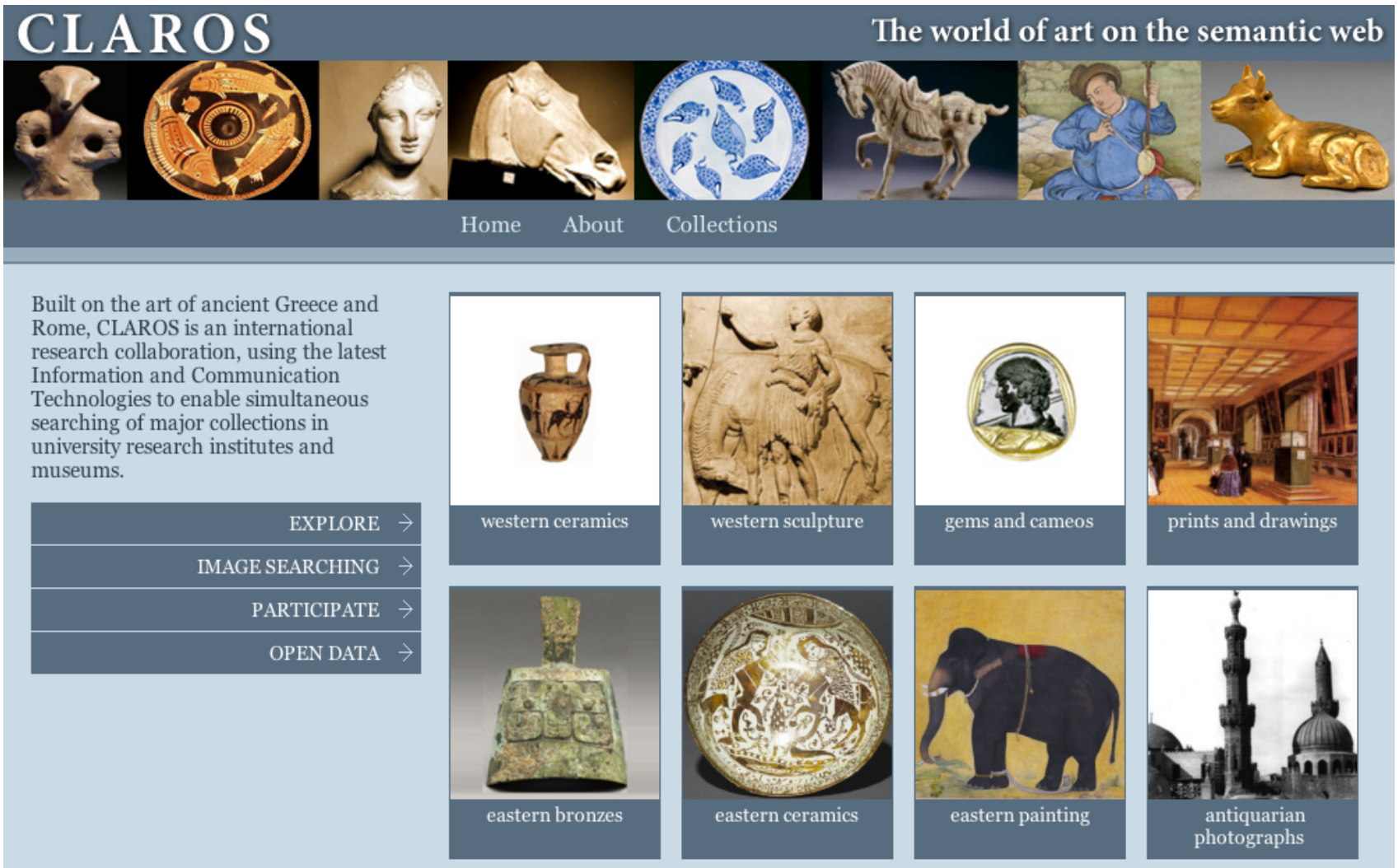
CLAROS currently includes art objects, archaeological site data, antiquarian photographs, and onomastics.

The Lexicon of Greek Personal Names contributes via representation in TEI XML.

plug: CLAROS

http://www.clarosnet.org

http://data.clarosnet.org



Four possible relationships of TEI to RDF

Discovery	Extract enough bibliographical information to make RDF triples for insertion into a resource finding aid
Extraction	Comb the text, using the TEI markup, looking for assertions about the 'real world' which we can represent in RDF
Mapping	Map everything found in the text into RDF
Container	Decorate our existing TEI markup with extra attributes which map to RDF

This talk is mainly about Extraction methods.

Target ontology

It depends on where we want to get:

- Dublin Core may be good for a general overview
- FRBR will make some library communities happy
- FOAF may be good for mapping persons and dates
- http://schema.org/Book might suit some TEI work

but it has long been a target of the Ontology SIG to align the TEI with ISO 21127:2006, the CIDOC conceptual reference model (CRM): see `http://www.cidoc-crm.org/` and `http://www.tei-c.org/SIG/Ontologies/guidelines/guidelinesTeiMappableCrm.xml`

Whatever the chosen ontology, the approach remains the same.

Where to store mapping?

It is easy enough to look at the TEI <person> element and say it corresponds to what the CRM calls *E21_Person*

> *This class comprises real persons who live or are assumed to have lived. Legendary figures that may have existed, such as Ulysses and King Arthur, fall into this class if the documentation refers to them as historical figures. In cases where doubt exists as to whether several persons are in fact identical, multiple instances can be created and linked to indicate their relationship. The CRM does not propose a specific form to support reasoning about possible identity. Examples: - Tut-Ankh-Amun - Nelson Mandela*

The <equiv> element allows us to provide a specification in ODD which points from a TEI element to an external identifier, and says how to get there.

Example of <equiv>

```
<elementSpec ident="geo" mode="change">
 <equiv
    filter="crm.xsl"
    mimeType="text/xsl"
    name="E47"
    uri="http://erlangen-crm.org/110404/E47_Place_Spatial_Coordinates"/>
</elementSpec>
```

name names the underlying concept of which the parent is a representation

uri references the underlying concept of which the parent is a representation by means of some external identifier

filter references an external script which contains a method to transform instances

mimeType gives the MIME media type of filter script

What is in crm.xsl?

Named XSL templates which do creation of RDF XML:

```
<xsl:template name="E47">
  <P87_is_identified_by>
    <E47_Place_Spatial_Coordinates>
      <value>
        <xsl:value-of select="."/>
      </value>
    </E47_Place_Spatial_Coordinates>
  </P87_is_identified_by>
</xsl:template>
<xsl:template name="E69">
  <P100i_died_in>
    <E69_Death>
      <P4_has_time-span>
        <E52_Time-Span>
          <P82_at_some_time_within>
            <E61_Time_Primitive>
              <xsl:call-template name="calc-date-value"/>
            </E61_Time_Primitive>
          </P82_at_some_time_within>
        </E52_Time-Span>
      </P4_has_time-span>
    </E69_Death>
  </P100i_died_in>
</xsl:template>
```

Combining the two

Read the ODD, extract filter information from each <equiv> and use it to generate a wrapper XSLT script:

```
<XSL:stylesheet version="2.0"
   xpath-default-namespace="http://www.tei-c.org/ns/1.0">
  <XSL:import href="crm.xsl"/>
  <XSL:template match="*">
    <XSL:apply-templates
       select="*|@*|processing-instruction()|comment()|text()"/>
  </XSL:template>
  <XSL:template
     match="text()|comment()|@*|processing-instruction()"/>
<!-- .... -->
  <XSL:template match="death">
    <XSL:call-template name="E69"/>
  </XSL:template>
  <XSL:template match="geo">
    <XSL:call-template name="E47"/>
  </XSL:template>
</XSL:stylesheet>
```

This is `rdf/make-acdc.xsl` in my stylesheet family.

Input

```
<person xml:id="ArnMag01" sex="1" role="scholar">
  <persName xml:lang="is">Árni Magnússon</persName>
  <persName xml:lang="la">Arnas Magnæus</persName>
  <persName xml:lang="da">Arne Magnusson</persName>
  <birth when="1663-11-13">13 November 1663</birth>
  <death when="1730-01-07">7 January 1730</death>
  <residence>
    <date from="1663" to="1680">1663-1680</date>
    <placeName>
      <settlement type="farm">Hvammur</settlement>
      <region type="county">Dalasýsla</region>
      <region type="compass">Western</region>
      <country key="IS">Iceland</country>
    </placeName>
  </residence>
  <residence>
    <date from="1680" to="1683">1680-1683</date>
    <placeName>
      <settlement type="institution">Skálholt</settlement>
      <region type="county">Árnessýsla</region>
      <region type="compass">Southern</region>
      <country key="IS">Iceland</country>
    </placeName>
  </residence>
</person>
```

Result

```
<RDF>
  <E21_Person
      rdf:about="http://www.example.com/idArnMag01">
    <P131_is_identified_by  xml:lang="is">
      <E82_Actor_Appellation
          rdf:about="http://www.example.com/persname/ArnMag01">
        <value>Árni Magnússon</value>
      </E82_Actor_Appellation>
    </P131_is_identified_by>
    <P98i_was_born>
      <E67_Birth>
        <P4_has_time-span>
          <E52_Time-Span>
            <P82_at_some_time_within>
              <E61_Time_Primitive>
                <value>1663-11-13</value>
              </E61_Time_Primitive>
            </P82_at_some_time_within>
          </E52_Time-Span>
        </P4_has_time-span>
      </E67_Birth>
    </P98i_was_born>
  </E21_Person>
</RDF>
```

Result (continued)

```
<RDF>
  <E21_Person
      rdf:about="http://www.example.com/person/ArnMag01">
    <P74_has_current_or_former_residence>
      <E53_Place
          rdf:about="http://www.example.com/place/hvammur">
        <P2_has_type
            rdf:resource="http://www.tei-c.org/type/place/settlement"/>
        <P87_is_identified_by>
          <E48_Place_Name
              rdf:about="http://www.example.com/placename/hvammur">
            <value>Hvammur</value>
          </E48_Place_Name>
        </P87_is_identified_by>
        <P89_falls_within
            rdf:resource="http://www.example.com/place/dalassla"/>
      </E53_Place>
    </P74_has_current_or_former_residence>
  </E21_Person>
  <E53_Place
      rdf:about="http://www.example.com/place/dalassla">
    <P2_has_type
        rdf:resource="http://www.tei-c.org/type/place/region"/>
    <P87_is_identified_by>
      <E48_Place_Name
          rdf:about="http://www.example.com/placename/dalassla">
        <value>Dalasýsla</value>
      </E48_Place_Name>
    </P87_is_identified_by>
    <P89_falls_within
        rdf:resource="file:/Users/rahtz/TEI/tei.oucs.ox.ac.uk/Talks/2011-10-teimm/test.xml#IS"/>
  </E53_Place>
```

or if you prefer N3 triples format

```
<http://www.example.com/persname/ArnMag01> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value>
"\u00C1rni Magn\u00FAsson"@is .
<http://www.example.com/persname/ArnMag01> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/NET/crm-owl#E82_Actor_Appellation> .
<http://www.example.com/place/rnesssla> <http://purl.org/NET/crm-owl#P87_is_identified_by>
<http://www.example.com/placename/rnesssla> .
<http://www.example.com/place/rnesssla> <http://purl.org/NET/crm-owl#P89_falls_within>
<file:///Users/rahtz/TEI/tei.oucs.ox.ac.uk/Talks/2011-10-teimm/test.xml#IS> .
<http://www.example.com/place/rnesssla> <http://purl.org/NET/crm-owl#P2_has_type>
<http://www.tei-c.org/type/place/region> .
<http://www.example.com/place/rnesssla> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/NET/crm-owl#E53_Place> .
<http://www.example.com/placename/denmark> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value>
"Denmark" .
<http://www.example.com/placename/denmark> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/NET/crm-owl#E48_Place_Name> .
<http://www.example.com/placename/germany> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value>
"Germany" .
<http://www.example.com/placename/germany> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/NET/crm-owl#E48_Place_Name> .
<http://www.tei-c.org/type/place/settlement> <http://purl.org/NET/crm-owl#P127_has_broader_term>
<http://www.tei-c.org/type/place> .
<http://www.tei-c.org/type/place/settlement> <http://www.w3.org/2000/01/rdf-schema#label>
"contains the name of a settlement such as a city, town, or village identified as a single
geo-political or administrative unit." .
```

Cleaning up

In practice, a second pass is needed to

- remove repeated assertions of same relationship (names)
- move embedded extra places to the right level

So how do I use this transformation?

If you follow the world of my XSL stylesheets (`http://tei.svn.sourceforge.net/viewvc/tei/trunk/Stylesheets/`):

- you can run `profiles/default/rdf/to.xsl` by hand on your file
- for command-line users, there is a script `teitordf`
- the web site/REST server OxGarage (`http://oxgarage.oucs.ox.ac.uk:8000/ege-webclient`) supports TEI P5 XML to RDF
- oXygen users can set up an Ant-based transformation (`rdf/build-to.xml`)

Demo time

.....

The problem of identifying things

RDF objects need to be *identified* to be at all useable, preferably with a real, stable, URI.

Among the ways we can generate an identifier:

- if a TEI element has a *@ref* attribute, that is perfect
- if we have an *@xml:id* and meaningful *@xml:base*, we can generate a good URL
- we may be follow a private *@key* if we know the scheme for the document
- we might generate new identifiers based on using <xsl:number> and an *@xml:base*

remembering that really *everything* needs an identifier. When CRM distinguishes Place from Place_Name, they both have to be identified.

The problem of ambiguity of context

Contrast

```
<place>
  <placeName>Bristol</placeName>
</place>
```

with

```
<p>He was born in <placeName>Bristol</placeName>
</p>
```

Problem of ambiguity: <name>

```
<name type="place">Zadar</name>
<name type="person">Oyvind</name>
<rs type="person">Piotr</rs>
```

A pre-processing stage may be order to resolve all such cases to a canonical format

The usual problems

- how to record location in TEI text of source claim
- date of claim
- how to actually express dates
- representing uncertainty and precision
- chronological periods
- how to actually express spatial coordinates

Taxonomies, categories, @type etc

Do we

- understand how to use E55_Type
- convert existing taxonomies and categories to SKOS notation, and refer to it
- enhance existing taxonomies with links to SKOS
- use informal @type from TEI documents to generate SKOS on the fly?
- something else?

What more could we do with the ODD?

Use it to create a schema to check whether a document can be mapped

- remove elements which cannot be matched to CRM
- add constraints to check situations which stop meaningful mapping
- eg check whether a <placeName> has a *@ref*
- eg check whether a <placeName> has a *@xml:id*

Another use of ODD

<district>, <settlement>, <region>, <country> and <bloc> are all members of *model.placeNamePart*. We can derive some types automatically:

```
<E55_Type rdf:about="http://www.tei-c.org/type/place">
  <rdfs:label>place</rdfs:label>
</E55_Type>
<xsl:for-each
   select="key('MEMBERS','model.placeNamePart')">
  <E55_Type
     rdf:about="http://www.tei-c.org/type/place/{@ident}">
    <rdfs:label>
      <xsl:call-template name="makeDescription"/>
    </rdfs:label>
    <P127_has_broader_term
       rdf:resource="http://www.tei-c.org/type/place"/>
  </E55_Type>
</xsl:for-each>
```

Result of that typology creation

```
<E55_Type
   rdf:about="http://www.tei-c.org/type/place/bloc">
  <rdfs:label>(bloc) contains the name of a geo-political unit consisting of
two or more nation states or
    countries.</rdfs:label>
  <P127_has_broader_term
    rdf:resource="http://www.tei-c.org/type/place"/>
</E55_Type>
<E55_Type
   rdf:about="http://www.tei-c.org/type/place/country">
  <rdfs:label>(country) contains the name of a geo-political unit, such as a
nation, country, colony, or
    commonwealth, larger than or administratively superior to a region and
smaller than a bloc.</rdfs:label>
  <P127_has_broader_term
    rdf:resource="http://www.tei-c.org/type/place"/>
</E55_Type>
```

etc

Alternatives to extraction

We *could* just abandon all the markup, and revert to linguistic analysis of text; this would let us try extraction of assertions from plain text using NLP. cf
`http://hypermedia.research.glam.ac.uk/kos/STELLAR/`

We could also add microdata attributes to our TEI markup and extract from there.

Conclusions

So was that realistic?

- So far as it goes, it performs *extraction* of viable RDF/XML from TEI P5 documents
- The coverage of TEI elements is not great, but relatively easily extended
- The biggest problem is working out the context of an assertion, because of the descriptive nature of TEI markup
- Much more work is needed to sort out all the possible ways of identifying objects