Connecting TEI content into an ontology of the editorial domain

Peter Boot[1], Marijn Koolen[2]
[1]Huygens ING, The Netherlands – peter.boot@huygens.knaw.nl
[2]Royal Netherlands Academy of Arts and Sciences - Humanities Cluster, The Netherlands – marijn.koolen@di.huc.knaw.nl

INTRODUCTION

Users of online digital editions have a need for annotation support to contribute explanatory material that complements what is already available in the edition itself, for purposes of private study or for publication in conjunction with a scholarly article (Boot 2009; Robinson 2004; Siemens et al. 2012). One challenge for browser-based annotation support is anchoring the annotation to a specific location in the digital edition, as the browser typically only has an HTML representation of the edition that describes the page layout.

Recently we proposed an ontology-based approach to describe digital editions and their components to a browser-based annotation tool so that it understands and can reason about what it is annotating (Boot et al. 2017; Boot and Koolen 2018). While the ontology was formulated in order to support principled and robust targeting of annotations, it should have a much wider applicability to ensure interoperability between the edition and other information items.

In our approach, the edition server will describe the components of the edition to annotation tools (as well as to other clients). This description will be in terms of an ontology for the editorial domain for which we have proposed a draft. The question that we discuss in this paper is: on the basis of which information is the edition application going to provide this information? How does it know to which classes certain text fragments belong, which URIs to assign and how to name the property that connects, say, a chapter and its paragraphs? Assuming a Text Encoding Initiative (TEI) context, where the user interface of the edition is generated from (a) TEI source(s), the natural answer to these questions would be to store the relevant information in, or at least with, these TEI sources. This implies the need for TEI files to refer to the annotation ontology: the fragments of the edited text have to be assigned URIs, they have to be assigned a class, and their mutual relationships have to be defined in terms of the properties described in the ontology. In other words, we need to overlay the graph model describing the edition and its content on the (hierarchical) TEI XML.

The aim of this paper is not to give a final answer to the question of how and where to store the source information for the linked data necessary to facilitate interoperable annotation on a digital edition; rather we discuss a number of options, with their advantages and disadvantages.

THE ANNOTATION ONTOLOGY

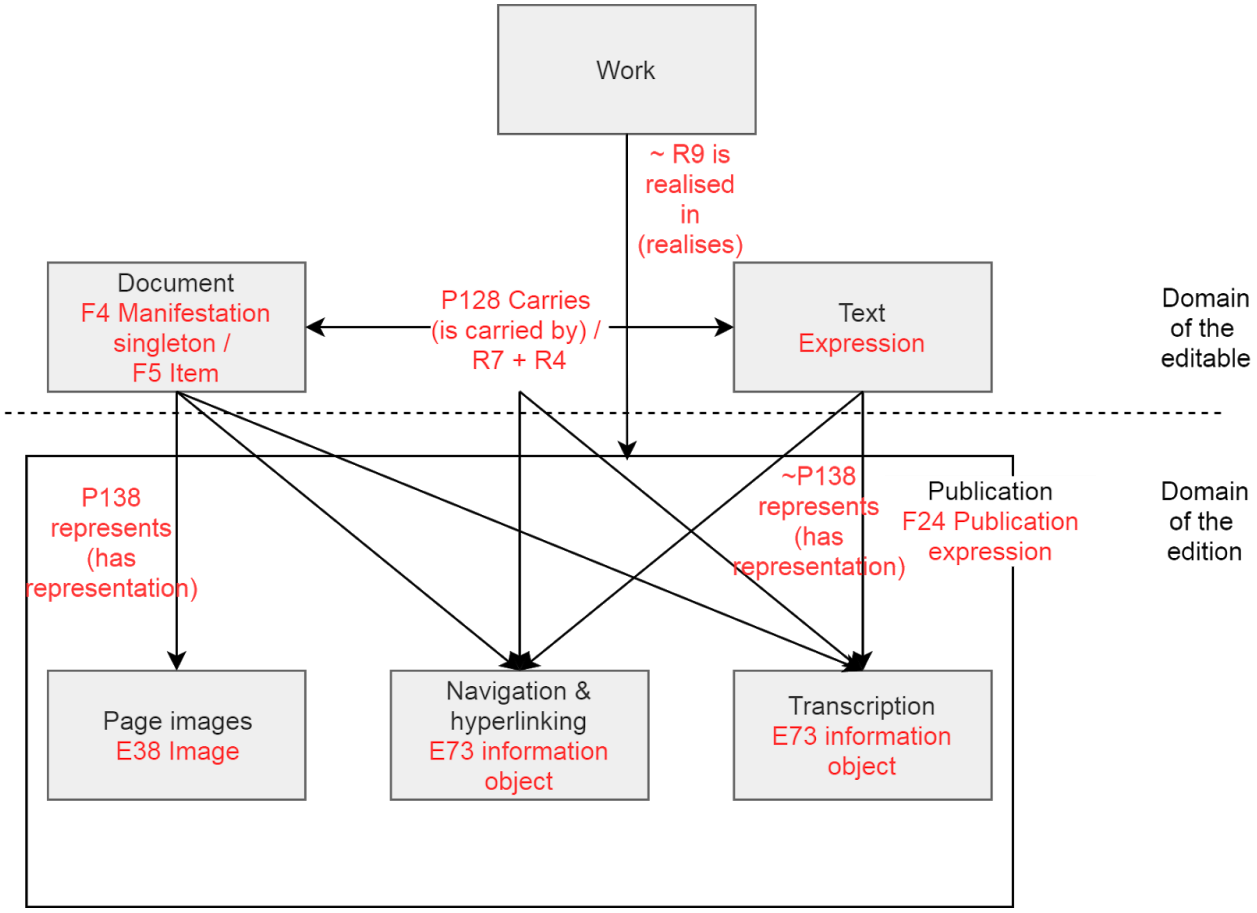
The Annotation Ontology is based on concepts from the $FRBR_{oo}$ ontology (IFLA 2015), which combines concepts from FRBR[1] and CIDOC/CRM[2] to describe a.o. manuscripts and their editions (LeBoeuf 2012). With the ontology, it is possible to describe the abstract work and its multiple representations as an RDF graph.

For a fuller discussion of the ontology, we refer to (Boot and Koolen 2018). The ontology (See figures 1 and 2) contains two important dimensions. First, a distinction between two domains, the editable and the edition domains, and second, a distinction between works and documents.

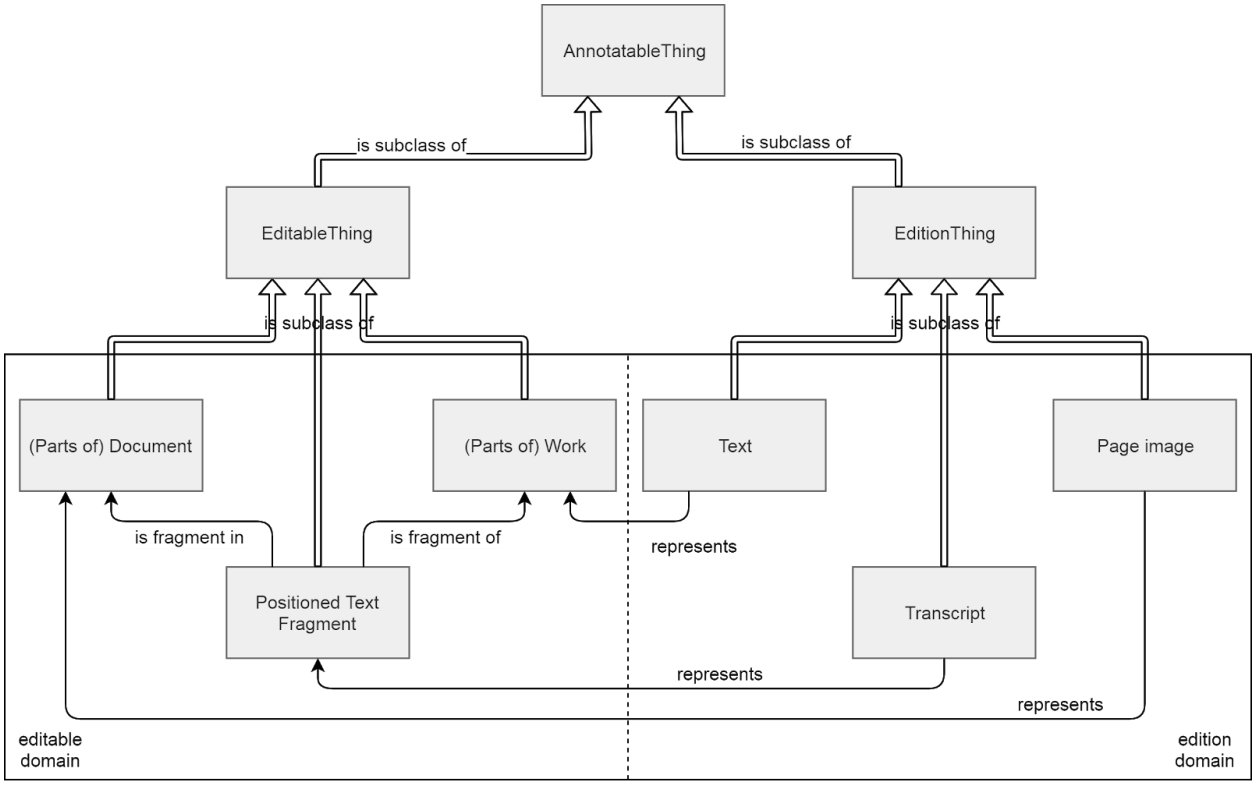
First, the editable domain has concepts for objects that are edited, such as documents and works, while the edition domain has concepts for the outcome of the editing process, such as digital texts and images. The two domains are connected to each other. Within each domain, the classes are subclasses of a main class, of respectively editable things (EditableThing, left side of Figure 2) and edition things, (EditionThing, right side of Figure 2). As they are all potential targets for annotation, they are all subsumed under a main class of annotatable things (AnnotatableThing).

Second, the Work is an abstract intellectual and creative entity instantiated in one or more Documents, or text bearers. Both Works and Documents can have parts. Following Robinson (2017), we also describe what we call Positioned Text Fragments (PTF): intersections of the Work and Document hierarchy, such as the part of a poem that appears on a certain manuscript page.

[1] Functional Requirements for Bibliographic Records
[2] The Conceptual Reference Model of the International Committee of Documentation (CIDOC)



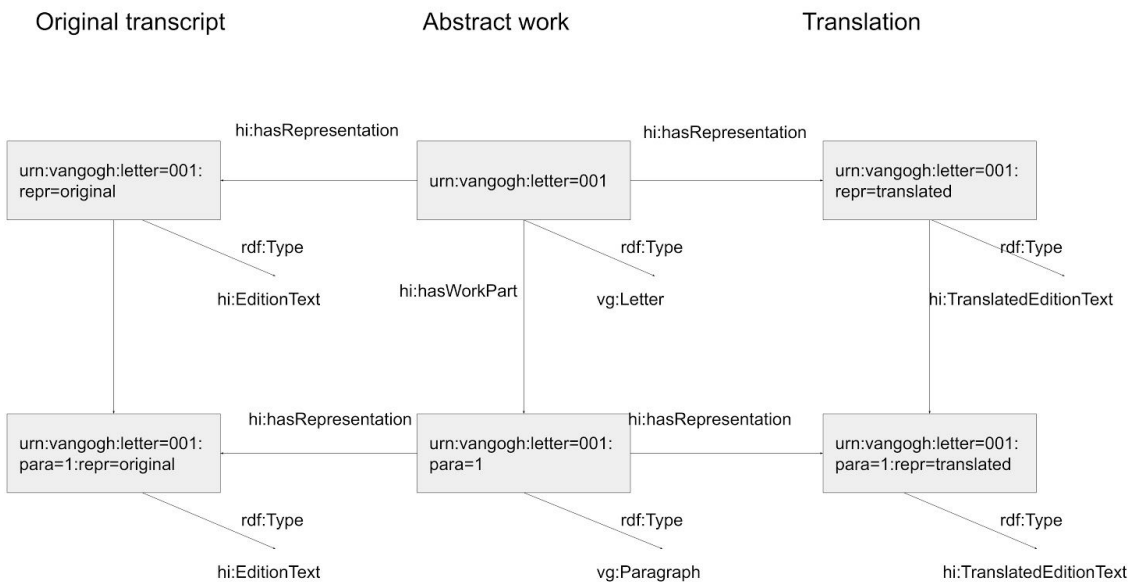
[Figure 1. The Annotation Ontology extends the $FRBR_{oo}$ and CIDOC/CRM ontologies. The concepts in black are concepts in our annotation ontology. The concepts in red are their (rough) equivalents in $FRBR_{oo}$ and CIDOC/CRM.]



[Figure 2. The Annotation Ontology of the editable and edition domains.]

To illustrate how this ontology can be used to describe digital editions, we use the digital edition of the correspondence of Vincent van Gogh (Jansen et al. 2009). Van Gogh's correspondence is a good example of the type of material that would benefit from interoperable annotation. Not only does the modern edition, as we will see, contain multiple representations of the same letter; the letters are also present in other forms or translations, on a number of other platforms (Douma ny; Koninklijke Bibliotheek 2016; Van Gogh 2019). In the modern edition, the abstract letter (middle column in Figure 3) is an instance of the class vg:Letter, which is a subclass of the $FRBR_{oo}$ class F1 Work.[3] It is a complex work consisting of multiple paragraphs, with each individual paragraph being an instance of Paragraph which is a subclass of (Part of) Work. The abstract letter and its paragraphs are related to two representations, namely a Dutch text (left column in Figure 3) and an English translation (right column in Figure 3). Each instance has a unique identifier that can be the target of an annotation. To make annotations interoperable, the identifiers should be generated according to a set of guidelines, so that different online editions, whether based on the same TEI files or not, use the same identifiers. In the next section we discuss how such identifiers could be generated from the TEI files.

[3] For the Van Gogh edition we use a specialized version of the ontology; classes from the specialized ontology are prefixed by 'vg'; for the general ontology we use the 'hi' prefix.



[Figure 3. An example of the Annotation Ontology describing a letter of Vincent van Gogh as abstract work in the editable domain (middle) and its representations in the edition domain in the form of a reading text in Dutch (left) and a translation in English (right).]

CONNECTING TEI AND THE ANNOTATION ONTOLOGY

In this section we will discuss a number of approaches to storing the RDF required for attaching the edition to the annotation ontology. For each approach, we look at how that approach attaches the URIs to the relevant objects and how it stores the triples defining the relations between the URIs. We discuss two cases: first, a case where we distinguish between a work and its (multiple) representations in the edition, and second, the familiar issue of overlapping page and textual hierarchies. For both cases, we use as an example a letter by Vincent van Gogh. For all of the approaches XML and XSLT samples are available in GitHub (Koolen et al. 2019a). We should be aware that these are relatively complex cases. There may be textual traditions where all annotation is, say, by line number in a single manuscript. In such cases, the required number of triples to describe the relevant context might be less daunting than in the cases we are about to discuss.

CRITERIA

The solution that we seek should satisfy a number of criteria: (i) it should be minimally obtrusive, (ii) maximally explicit, (iii) minimally redundant and (iv) maximally generic and flexible. Preferably, (v) it should work without extending the TEI.

Ad (i): XML is sometimes seen as verbose and distracting (Bambaci, Boschetti & Del Gratta 2018). It is important that human readability and manageability of the XML is not impaired by computer code that is meaningless for the textual researcher.
Ad (ii): As much as possible, all information necessary for creating the edition should be contained in the XML file. If it is not included in the file, the file should unambiguously indicate how additional information is to be obtained.

This is important for preservation purposes: an archived TEI file should contain all of the information required to create an edition.
Ad (iii): Information should not be unnecessarily duplicated
Ad (iv): The solutions we propose should not constrain editors to use TEI in a way that would otherwise be unsuitable for their projects. In fact, none of the approaches that we test create serious constraints.
Ad (v): In the ideal case, our solution works in TEI as-is. But we should not be afraid of extending it where necessary.

ONE WORK, MULTIPLE REPRESENTATIONS

The situation that we discuss here is the one depicted in Figure 3: one work, a letter, for which two texts are available, a version in the original language and a translated version. We assume a situation where the annotatable units are the letter/text as a whole, and its individual paragraphs. The required RDF identifies the work, texts and paragraphs, creates hasRepresentation properties between work and texts (paragraphs) and describes the hierarchies between the works, texts and their paragraphs.
Apart from practical and convenience issues (readability), the essential question here is: how to make it possible for a single XML hierarchy to be associated with two conceptual hierarchies: the hierarchy of the work and the hierarchy of the work's representation.

GRDDL, private URI schemes and dedicated URI attribute

The GRDDL approach is based on the fact that for all triples, the necessary information is already contained in the XML file. Rather than including redundant information in the XML, this approach takes advantage of the GRDDL standard to associate an XML file with a program that derives an RDF representation of its contents (Conolly 2007). For defining the URIs, we create a dedicated attribute (here called ontRef). To keep the URIs short, we use the TEI facility of a private URI scheme: a prefix that is associated with a prefixDef in the TEI header which defines a pattern replacement to create a URI out of a brief pattern (TEI Consortium 2019, section 16.2.3). To associate two URIs (for work and text) with a single XML element, we create two prefix definitions for the same prefix.[4]

In practice, this approach might look like this:

Association between the TEI file and a GRDDL script:

```
<TEI xmlns:vg="http://www.vangoghletters.org/ns/"
    xmlns="http://www.tei-c.org/ns/1.0"
    xmlns:grddl='http://www.w3.org/2003/g/data-view#'
    grddl:transformation="grddl.xsl">
```

prefixDefs at the letter (div) level, defined in the TEI header:

```
<prefixDef ident="letter" matchPattern="([a-z]+),([0-9]+)"
replacementPattern="urn:vangogh:letter=$2">
    <p>Associates letter div with uri of letter as work</p>
</prefixDef>
<prefixDef ident="letter" matchPattern="([a-z]+),([0-9]+)"
replacementPattern="urn:vangogh:letter=$2:repr=$1">
    <p>Associates letter div with uri of this edition of the letter</p>
</prefixDef>
```

ontRef attribute on divs holding original and translated text:

```
<div type="original" ontRef="letter:original,1">
<div type="translation" ontRef="letter:translated,1">
```

[4] An option discussed in the same section of the *TEI P5 Guidelines*.

A fragment of GRDDL-generated RDF in turtle format:

```
<urn:vangogh:letter=1> rdf:type vg:Letter.
<urn:vangogh:letter=1:repr=original> rdf:type hi:EditionText.
<urn:vangogh:letter=1> hi:hasRepresentation <urn:vangogh:letter=1:repr=original>.
<urn:vangogh:letter=1:para=1> rdf:type vg:ParagraphInLetter.
<urn:vangogh:letter=1:para=1:repr=original> rdf:type hi:EditionText.
<urn:vangogh:letter=1:para=1> hi:hasRepresentation
    <urn:vangogh:letter=1:para=1:repr=original>.
<urn:vangogh:letter=1> hi:hasWorkPart <urn:vangogh:letter=1:para=1>.
<urn:vangogh:letter=1:repr=original> hi:hasTextPart
    <urn:vangogh:letter=1:para=1:repr=original>.
```

At the top of the document, the reference (in the GRDDL namespace) to grddl.xsl defines the stylesheet to execute in order to create an RDF representation of the document. This stylesheet, when processing the XML file, encounters the ontRef attribute on the divs holding the original language-version. It recognizes its value `"letter:original,1"` as a private URI scheme reference with the prefix `"letter"`. It searches for corresponding prefixDef elements and encounters two of these. It executes the corresponding match-and-replace operations and creates these two URIs: `urn:vangogh:letter=1/` and `urn:vangogh:letter=1/repr=original/`. The logic for defining the triples (i.e. to assign the URIs to classes and to describe their properties) is contained in the stylesheet.

Advantages and disadvantages

We discuss advantages and disadvantages in terms of the criteria that we mentioned above.

Minimally obtrusive: Referring to a GRDDL stylesheet does not affect the readability of the XML. Including ontRef attributes for each addressable element to hold an abbreviated pointer does affect readability, even though it could be much worse (if we included full URI's, e.g.)

Maximally explicit: To generate the triples, the GRDDL stylesheet needs to be run. This implies that we need to be able to run an XSLT processor before the full information is available. The chances of being able, without significant effort, to run an XSLT processor in, say, fifty years' time, are slim. From that perspective this is a very weak solution. Indeed, GRDDL itself 'is still a mere theoretical specification' (Grüntgens & Scharde 2016). A positive aspect is that usage of the dedicated ontRef attribute to refer to the URI is more explicit than using a general-purpose attribute such as corresp. A less than desirable aspect, however, is that the software has no way of knowing which of the prefixDefs produces the work URI and which the text URI.

Minimally redundant: The triples are only generated as needed and therefore raise no redundancy concerns. The abbreviated pointers also avoid redundancy. However, the prefix definitions are included in each of the XML files, and for the van Gogh edition that means more than 900 times. They could be included through XInclude.

No TEI extension: The solution requires two new attributes: grddl:transformation and ontRef.

GRDDL, no URI representation in TEI file

This approach, like the preceding one, uses GRDDL to deduce the necessary triples rather than include their representation in the TEI file. It goes one step further as it also leaves the URI definition out of the TEI. Instead, the TEI file contains xml:id attributes, and the GRDDL stylesheet generates triples that reference the xml:id attributes, as follows:

Div elements with xml:id attributes:

```
<div type="original" xml:id="letter.original.1">
<div type="translation" xml:id="letter.translated.1">
```

Generated triples included now (beyond the ones from the previous approach):

```
<urn:vangogh:letter=1> hi:refersTo
    file:/.../let001.xml#letter.original.1.
<urn:vangogh:letter=1:repr=original> hi:refersTo
    file:/.../let001.xml#letter.original.1.
```

In this case, the stylesheet handles the generation of the URIs. The XML no longer needs to contain an ontRef attribute and prefixDefs to define the logic of the URI generation. The XML must provide xml:id attributes for the elements that are to be attached to a URI.

Advantages and disadvantages

Minimally obtrusive: Referring to a GRDDL stylesheet does not affect the readability of the XML. The relevant elements need to carry xml:id attributes, but this is good practice anyway.

Maximally explicit: This approach is even less explicit than the previous one, as we have now removed the URI definition (through the prefixDefs) from the XML into the GRDDL stylesheet.

Minimally redundant: An improvement with respect to the previous approach, as the prefixDefs no longer need to be defined in each TEI file.

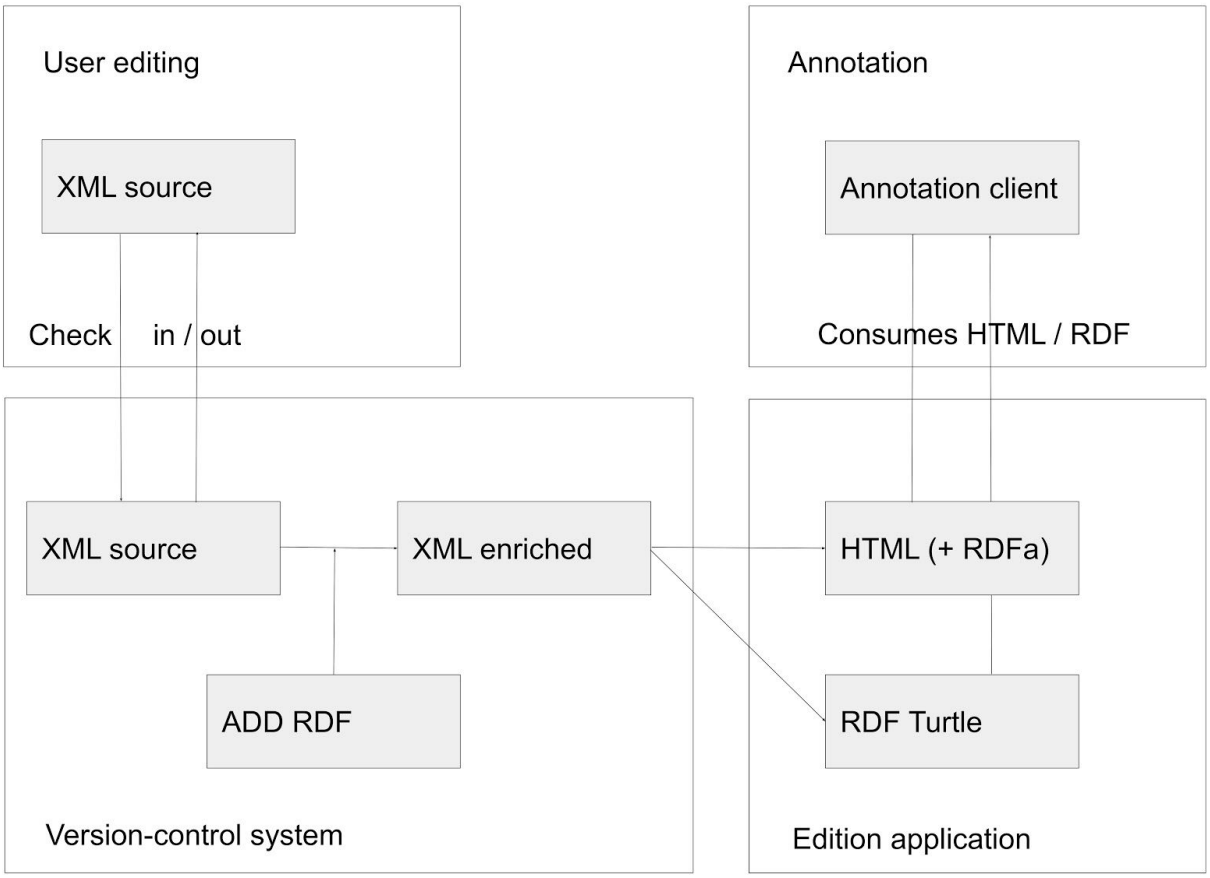
No TEI extension: The approach requires one new attribute, grddl:transformation.

RDF in xenodata, URIs generated from xml:ids

This approach does not use GRDDL. The information necessary to deduce triples and URI has to be included in the TEI/XML file. Here we store that information in the xenodata element in the TEI header, introduced expressly to contain non-TEI (but still XML) metadata. The triples will therefore be included as RDF/XML. In this approach, we store the generated URI's on the relevant elements in a newly defined ontRef attribute.

In any realistic scenario, this still implies the need for a stylesheet or other program to generate the triples. Manually typing that RDF would be prohibitively time-consuming and error-prone. The difference with the GRDDL approach is twofold: (1) the output here is an enriched XML file, rather than (in the GRDDL case) an external RDF file; and (2) here the encoder is responsible for execution of the stylesheet, before the TEI is published, while in the GRDDL case the stylesheet is (theoretically) executed by the consumer of the XML file, after publication. This also implies that in this solution, for the benefit of the annotation tool (and possibly other RDF processing tools), the edition server will lift the triples out of the TEI environment into a format that the annotation tool will be able to understand.

We envisage a situation where an editor works on an XML file without the RDF information. Upon ingestion in a version control system, the system executes a post-editing program that creates a file enriched with RDF. The editor will never need to see the enriched file but keeps working on the original file. The situation looks like this (Figure 4):



[Figure 4. Overview of the editing and annotation workflow.]

A fragment of the generated xenodata element might look like this:

```
<tei:xenodata xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:hi="http://boot.huygens.knaw.nl/vgdemo1/editionannotationontology.ttl#">
    <rdf:RDF>
        <rdf:Description about="urn:vangogh:letter=1" rdf:type="&vg;Letter"/>
        <rdf:Description about="urn:vangogh:letter=1:repr=original"
            rdf:type="&hi;EditionText"/>
        <rdf:Description about="urn:vangogh:letter=1"
            hi:hasRepresentation="urn:vangogh:letter=1:repr=original"/>
        … …
    </rdf:RDF>
</tei:xenodata>
```

As usual in XML/RDF, here we use XML entities (`&hi;`, `&vg;`) as abbreviations, because namespaces don't work in xml attributes. The entities are defined at the top of the document:

```
<!DOCTYPE any  [
  <!ENTITY hi 'http://...../.../editionannotationontology.ttl#'>
  <!ENTITY vg 'http://...../.../vangoghannotationontology.ttl#'>
]>
```

The generated URIs for the letter's components are stored in their ontRef attribute (two URIs! One for the work, one for the edited text):

```
<div type="original" xml:id="letter.original.1"
    ontRef="urn:vangogh:letter=1 urn:vangogh:letter=1:repr=original">
    <ab xml:id="para.original.1.1"
        ontRef="urn:vangogh:letter=1:para=1
                urn:vangogh:letter=1:para=1:repr=original">
       [content of para]
    </ab>
```

Advantages and disadvantages

Minimally obtrusive: If the RDF enrichment can indeed be handled as a post-editing process, where there is no need for the editor to work with the enriched file, the XML remains maximally readable for the editor.

Maximally explicit: All RDF is contained in the enriched XML. In that sense, the approach is fully explicit. It might be considered a disadvantage that the RDF is not easily accessible to general-purpose RDF tools; however, the conversion to, say, a turtle file, is trivial.

Minimally redundant: The choice for a post-editing approach, where the original XML and the enriched XML will be maintained separately for an indefinite period of time, entails some redundancy. However, as the enriched file is automatically created upon check-in in the version control system, this needn't present any problem.

No TEI extension: The approach requires adding an ontRef attribute to the TEI schema.

Triple representation on relation element, URIs generated from xml:ids

Here we follow the choice made by the SAWS project (Jordanous, Lawrence, Hedges and Tupman 2012). In this scenario the relation element, originally meant to describe relations between persons, is used to carry any sort of triple. The @active attribute points to the subject of the triple, the @ref contains the property and the @passive attribute the object. The SAWS project used the @active and @passive attributes to refer to the xml:id attributes of the corresponding xml elements. In our case, this will not work, because we need to refer to the xml elements in their double capacity of work and text. We need to express, for instance, that the text is a representation of the work. On the relation element we therefore use URIs rather than just pointers to xml:ids. As we assume the post-editing scenario outlined above, using full URIs presents no legibility problems.

Some of the generated relation elements will look like this:

```
<tei:listRelation xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <tei:relation active="urn:vangogh:letter=1" ref="rdf:type"
        passive="http://.../vangoghannotationontology.ttl#vg:Letter"/>
    <tei:relation active="urn:vangogh:letter=1:repr=original" ref="rdf:type"
        passive="http://.../editionannotationontology.ttl#hi:EditionText"/>
    <tei:relation active="urn:vangogh:letter=1"
        ref="http://.../editionannotationontology.ttl#hi:hasRepresentation"
        passive="urn:vangogh:letter=1:repr=original"/>
    ...
</tei:listRelation>
```

As will be clear, this is essentially the same solution as the xenodata approach, just expressed in a different syntax. The advantages and disadvantages are similar.

Triples as RDFa, URIs generated from xml:ids

RDFa was designed with the purpose of embedding RDF into hierarchical languages such as HTML and XML (W3C 2015). The idea is that the information that is already contained in the document can be labeled with semantic information using a number of extra attributes, the most important of which are about, resource, typeof and property. An RDFa processor can then extract RDF from the document by combining the structure and content of the document with the RDFa labels. The approach was successfully used in (Tittel, Bermúdez-Sabel & Chiarcos 2018). RDFa is mostly used for expressing hierarchical structures, say an agenda with agenda items. A very common structure is that a URI is assigned to a high-level element (say a div holding the agenda) using the about attribute; underlying elements (say agenda items in p elements) are assigned a URI through a resource attribute; and the relation between the top and the lower level element is expressed using a property attribute. For each element, the typeof attribute defines the class that the attribute belongs to.

A single hierarchy, for example the relation between the work and its paragraphs, could be easily expressed using RDFa:

```
<div type="original" xml:id="letter.original.1"
    about="urn:vangogh:letter=1" typeof="vg:Letter">
    <ab xml:id="para.original.1.1"
        about="urn:vangogh:letter=1:para=1" typeof="vg:ParagraphInLetter"
        property="hi:hasWorkPart">
        <!-- content of para -->
    </ab>
    <!-- rest of paragraphs -->
</div>
```

The hierarchy of the XML translates naturally into the triple describing the relation between work and paragraph using the property hi:hasWorkPart.

However, in the case of our double hierarchy this no longer works. We have either to duplicate the entire hierarchy elsewhere in the XML document or to create the triples representing the hierarchy manually by adding empty seg elements.

A fragment of the RDFa-enhanced XML could look like this:

```
<div type="original" xml:id="letter.original.1" about="urn:vangogh:letter=1"
    typeof="vg:Letter">
    <tei:seg resource="urn:vangogh:letter=1:para=1" property="hi:hasWorkPart"/>
    <!-- similar segs for other work paras>
    <tei:div resource="urn:vangogh:letter=1:repr=original"
        property="hi:hasRepresentation" typeof="hi:EditionText">
        <tei:seg resource="urn:vangogh:letter=1:para=1:repr=original"
            property="hi:hasTextPart"/>
        <!-- similar segs for other text paras>
        <ab xml:id="para.original.1.1" about="urn:vangogh:letter=1:para=1"
            typeof="vg:ParagraphInLetter">
            <tei:seg resource="urn:vangogh:letter=1:para=1:repr=original"
                property="hi:hasRepresentation" typeof="hi:EditionText">
                <!-- content of para -->
            </tei:seg>
        </ab>
        <!-- rest of paragraphs -->
```

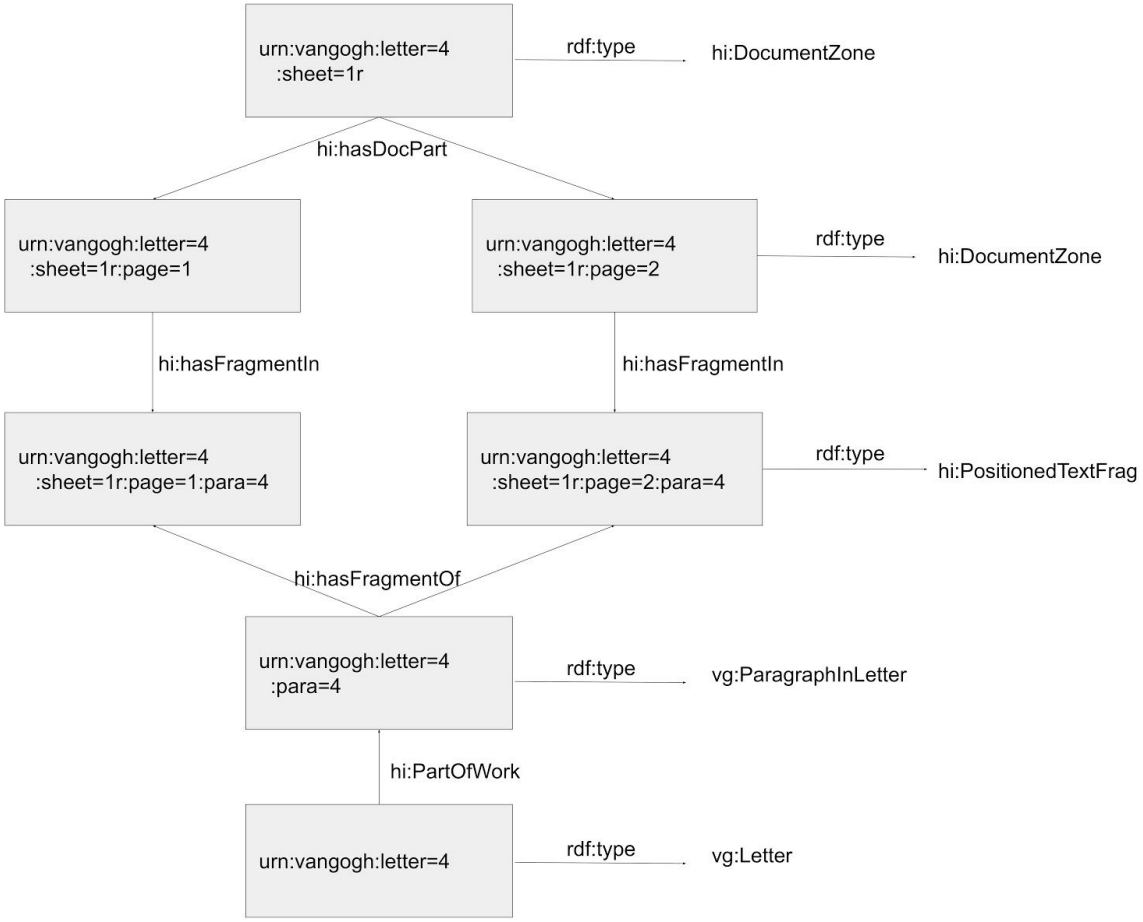
```
    </tei:div>
</div>
```

The outer div defines the letter (work) URI. The first empty seg element inside it relates the letter to its first paragraph. To simplify, we only show the first of these. The inner div defines the text URI. Here too there is an empty seg to relate it to its (first) paragraph. The paragraph (ab element) carries the work paragraph URI. Inside, there is a seg-element that carries the text paragraph URI.

Advantages and disadvantages
Minimally obtrusive: As this is again a post-editing process, the original XML remains readable for the editor. However, the enriched RDFa-XML has a convoluted structure and becomes close to unreadable.
Maximally explicit: As in the xenodata and relation approaches, all RDF is contained in the enriched XML. Here too, it can be trivially extracted for use by general-purpose RDF tools.
Minimally redundant: Manageable redundancy, as above.
No TEI extension: The approach requires adding the RDFa attributes to the TEI schema.

TEXT AND PAGE HIERARCHIES

The second example that we discuss is concerned with the annotation of fragments of text in a double hierarchy: the logical hierarchy of text and text parts and the physical hierarchy of pages and lines. It should be possible to reuse an annotation on a certain manuscript line both in the context of other annotations on that page and in the context of other annotations on the relevant paragraph. Robinson (2017) has devised the DET (Document Entity Text) addressing scheme which allows pointing at intersections of the logical and physical hierarchies, which we have called Positioned Text Fragments (PTFs). A PTF might be a part of a paragraph as it appears on one line in one manuscript, it might also be a part of a work as realized on one manuscript page.
In the example that we discuss here, we look at the intersections of pages and paragraphs. For a case of overlap between those hierarchies, we switch our attention to letter 4. Paragraph 4 is split over two pages. The situation that we want to represent looks like this:



[Figure 5. RDF representation of two positioned text fragments in the text and page hierarchies.]

The main issue in representing this graph in the XML source is that the positioned text fragments as such are not (usually) represented in the XML. They exist only implicitly, as the intersection between the logical hierarchy and the (milestone-based) physical hierarchy.

The XML of paragraph 4 looks like this (simplified):

```
<ab>How sorry I am about Uncle Hein. I sincerely hope he'll get better, but Theo, I
fear he won't. Last summer he was still <pb /> so full of ambition, and had so many
plans and told me that business was going so well. It is indeed sad.
</ab>
```

Representing the paragraph-page PTFs would turn this into something like:

```
<ab>
    <ptf>How sorry I am about Uncle Hein. I sincerely hope he'll get better, but
        Theo, I fear he won't. Last summer he was still
    </ptf>
    <pb/>
    <ptf> so full of ambition, and had so many plans and told me that business was
        going so well. It is indeed sad.
    </ptf>
```

```
</ab>
```

If there were an italicized phrase around the page break, we would have to split the PTFs. Using the n-attribute to connect what should logically be considered as a single PTF, we get:

```
<ab>
    <ptf n="1">How sorry I am about Uncle Hein. I sincerely hope he'll get
        better, but Theo, I fear he won't. </ptf>
    <emph>
        <ptf n="1">Last summer he was still </ptf>
        <pb/>
        <ptf n="2"> so full of ambition</ptf>
    </emph>
    <ptf n="2">, and had so many plans and told me that business was going so well.
         It is indeed sad.</ptf>
</ab>
```

This would be clearly impracticable in a file that should be usable for editors. This seems to imply that we have the choice between a GRDDL-like solution, where a client dynamically generates RDF at runtime, or a post-editing script as discussed in some of the other approaches above.
In fact, however the GRDDL solution will not work. Remember that a GRDDL script generates RDF that other programs can consume, but it leaves the XML unchanged. The generated RDF will use URIs to refer to PTFs, but in the XML there will be nothing that corresponds to these URIs. To find out what an annotation of a PTF refers to, we'd have to inspect the logic of the GRDDL script. This implies that in this case a post-editing script is the only workable answer.

The result of the post-editing script would be an XML file enhanced with triples (e.g. in a xenodata element), with pb elements, divs and paragraphs provided with ontRef attributes, and seg elements with type=ptf to represent the positioned text fragments.

Apart from the work level information about work and paragraphs that we've seen in the previous sections, the xenodata element now would also define the sheet of paper, it would describe the pages on the sheet and the positioned text fragments. It would also describe the relations between these objects:

```
<tei:xenodata>
    <rdf:RDF>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1r"
            rdf:type="http://...../.../editionannotationontology.ttl#DocumentZone"/>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1r"
            hi:hasDocPart="urn:vangogh:letter=4:sheet=1r:page=1"/>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1r:page=1"
            rdf:type="http://...../.../editionannotationontology.ttl#DocumentZone"/>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1r:page=1:para=4"
            rdf:type="http://...../.../editionannotationontology.ttl#PositionedTextFrag"/>
        <rdf:Description about="urn:vangogh:letter=4:para=4"
            hi:hasFragmentOf="urn:vangogh:letter=4:sheet=1r:page=1:para=4"/>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1r:page=1"
            hi:hasFragmentIn="urn:vangogh:letter=4:sheet=1r:page=1:para=4"/>
        <rdf:Description about="urn:vangogh:letter=4:sheet=1v:page=2"
            hi:hasFragmentIn="urn:vangogh:letter=4:sheet=1v:page=2:para=4"/>
        … …
    </rdf:RDF>
</tei:xenodata>
```

The triples would be about the URIs defined in the transcription:

```
<div type="original" xml:id="letter.original.4" ontRef="urn:vangogh:letter=4">
    <pb f="1r" n="1" xml:id="pb-orig-1r-1" facs="#zone-pb-1r-1"
        ontRef="urn:vangogh:letter=4:sheet=1r:page=1"/>
    <lb n="1" xml:id="l-1"/>
    <ab xml:id="para.original.4.1" ontRef="urn:vangogh:letter=4:para=1">
        <tei:seg type="ptf" ontRef="urn:vangogh:letter=4:sheet=1r:page=1:para=1">
            [content]
        </tei:seg>
    </ab>
```

The advantages and disadvantages of this approach are really the same as those of the xenodata approach discussed above. The XML that the editor sees remains fully readable, the enriched XML contains a full RDF representation, the resulting redundancy is manageable and the only TEI extension that is needed is the addition of an ontRef attribute.

CONCLUSION AND DISCUSSION

There is no single best way to store the source for the RDF describing a text structure's embedding in the Linked Data world. We discussed a number of approaches for two different cases, but there are other cases. Combinations of approaches would also be possible. E.g., to work around the double hierarchies in RDFa we could define the work hierarchy under the xenodata element, and use RDFa to define the hierarchies of the representations (original text and translation). However, this would make it harder to lift the RDF triples out of the XML.

From the preceding discussion it appears that the GRDDL approach is weakest in terms of explicitness. The fate of the standard is unclear and the archived TEI files will contain no representation of the triples or even, in our second GRDDL approach, of the URIs. GRDDLs strong point, the minimal redundancy (and therefore, user-friendly XML) can also be obtained if we implement a post-editing script in the version-control system, as used in the other approaches. Of these, the RDFa approach might be suitable for simple cases, where all annotation uses a single hierarchy. In other cases, the generated RDFa becomes unwieldy. The remaining approaches (using either the relation or the xenodata element) are very similar. An advantage of the relation element might be that the triples can be stored near (one of) the elements that they are about, while the xenodata approach collects all triples in the document's header. For our post-editing approach, that consideration is not really relevant. We prefer the xenodata approach because it is closest to the RDF model.

This is not necessarily in general the best way for embedding RDF in TEI XML. One thing to keep in mind is that in our case the RDF can be deduced from the XML structure, based on rules. This situation is fundamentally different from e.g. that of the SAWS project (see above) where the RDF triples represent new knowledge.

Given that we propose to derive the triples upon ingestion in the version control system, our criterion of maximal explicitness would prescribe that the TEI files contain the name of the program that deduces the RDF statements. The version control system should use the programme that the TEI file mentions. As yet we have not formulated a preference for how the program name should be stored in the TEI file. An obvious choice would be to include the information in an XML processing instruction. Another option would be to widen the scope of the existing TEI element appInfo, now used for recording past processing steps, to define also required future processing steps. The latter choice would require a hook in the version control system that knows how to read TEI, and might be less flexible than using a processing instruction.

In this paper we have assumed that all URIs for the objects that we want to refer to are URNs; our objects are abstract information objects and not, e.g., web pages. Our URNs are all persistent, as they should be. More specifically, we have made our URNs transparent: by constructing them as lists of numbered units (e.g. '`letter=4:sheet=1r:page=1:para=1`') software as well as humans can understand them and act upon them. This is not strictly necessary. Projects that require opaque identifiers could choose to assign unique random identifiers to their objects. However, these URIs could no longer be generated on the basis of the XML's hierarchical structure alone and would have to be included in the XML.

OUTLOOK

We have developed a prototype annotation tool that reads the RDFa describing the edition, as displayed in the browser, as well as any linked RDF descriptions of external components and allows users to make various types of annotations (Boot and Koolen in print). The software is open source and consists of a JavaScript client (Koolen et al. 2019a) that edition creators can easily incorporate in their online editions and a server (Koolen et al. 2019b) that can store and retrieve the annotations. Edition creators can decide to run their own annotation servers or configure the client to communicate with other annotation servers. A next step for us will be to test this approach with a number of real editions.

Our approach to annotation creates the possibility for users to choose very specific targets for their annotation. One of the issues that we will have to confront in testing is whether this flexibility might not, paradoxically, decrease the interoperability of user annotations. The flexibility that we offer might be confusing to users and lead them to choose inappropriate annotation targets. It is possible that the distinctions that FRBR makes are too subtle for the average user.[5] We do not expect this in itself to be a problem. After all, we design an annotation tool for scholarly use, and though FRBR terminology might be complex, distinctions such as the one between a work and a copy are the bread and butter of scholarship. Research about the issue is not really available as annotation tools that support the distinctions are scarce. Most research about FRBR usability has been in the context of library catalogs. "FRBR-based displays do make sense to users and better support their tasks" write Zhang & Salaba (2012) and similar findings are reported in (Kim 2015). Hunter and Gerber (2010) present, as far as we know, the only study of FRBR-based annotation. They find that the FRBR ontology is complex to many users indeed. However, the tool they tested only supports annotation at the entry-level, not on text fragments. We really need more experimentation to understand how FRBR-based annotation will work out.

Meanwhile, within the domain of textual editing, our approach may be useful for other forms of interaction with the text besides annotation. The URNs that we use are not coincidentally very similar to the URN's used in the Canonical Text Services protocol (Blackwell and Smith 2014). They could provide the basis for exchange of text fragments between applications and digital text collections. Beyond the domain of textual editing, a similar approach will be useful in fields that also have multiple representations of a single original object, such as in the field of video (think of representations in multiple resolutions, separate audio tracks or textual transcripts, or an entire news program vs. individual items). Even in photography, an annotation can be about the photographed object or about the picture itself. Distinguishing between object and representation and embedding both in their context is a key requisite for interoperable annotation and other applications.

[5] We thank Michele Pasin for this pertinent question.

ACKNOWLEDGEMENTS

This work is partly funded by the KNAW 'Vernieuwingsgelden' and by the NWO project CLARIAH.