CODA final report

September 29, 2012
Hennie Brugman

Table of Contents

1 Executive summary

The CATCHPlus Open Document Annotation project (CODA) was executed as one of the OAC phase II experiments with funding of the Andrew W. Mellon Foundation. The project team consisted of researchers and software developers at the Meertens Institute and at the Computing and Cognition department of the University of Groningen. The team had help and advice from Henny van Schie at the Dutch National Archive, the provider of the collections used for CODA.

The project builds on and adds to the results of the CATCHPlus project. CATCHPlus is a valorization project that is associated with the Dutch CATCH research programme. CATCH includes a number of application driven research projects at large Dutch cultural heritage institutions. CATCHPlus builds tools and services on basis of research prototypes and demonstrators from CATCH. In this way it contributes to the digital cultural heritage infrastructure in the Netherlands and Europe.

CODA had two main use cases: 1. Convert existing line-by-line transcriptions of scanned documents from the index books of the Queen's Cabinet to OAC and use these OAC annotations as input for a Named Entity Recognition service. Output of this NER service is also compliant with the OAC model. 2. Automatically detect bounding boxes surrounding written lines on scans from the Sailing Letters collection and represent these bounding boxes as Open Annotations. The bounding boxes can be used in subsequent manual transcription tasks.

Our experiment's use cases raised many non-trivial challenges concerning representation of annotations of scanned documents using OAC, which we managed to tackle. Other interesting issues arose from publication and search scenarios. We refer to chapter 7 for a summary of the most important conclusions with regard to the applicability of OAC for our use cases.

The CODA project successfully delivered the following products:

- Mappings and conversion software for 'monk', the proprietary format used for line and word transcriptions of the Queen's Cabinet collection and for FoLiA, a linguistic annotation format produced by the Named Entity Recognition software used.
- An OAC compliant Annotation server (OAS) that supports upload of annotations and efficient and scalable text based search for annotations on basis of all OAC classes and properties, Dublin Core properties and all other text based properties. OAS instances can exchange annotation sets using its built-in OAI-PMH data provider and harvester. It provides a (linked data) publication platform for Open Annotations and contains a SPARQL endpoint.
- Named Entity Recognition web service for Dutch texts
- Line strip cut out service for image scans
- A view and search web application that contains synchronized viewers on named entities, transcription text and image scans.

- A website: http://coda-project.org that will be maintained, also after the project's end.
- Several reports about OAC modeling issues
- Two publications at international conferences/workshops

The CODA project was executed in time and within budget. Finalization of software, making it available as online web applications and services and publication of sources on GitHub will go on for a few weeks after finishing CODA.

Some of the software products of CODA may be of general use for the Open Annotation community: the OAS annotation server was built by an expert company using industry quality well tested components, and with substantial CATCHPlus budget. The line strip cut out service may be very interesting for the Open Annotation users that work with scanned scholarly documents.

The Meertens Institute intends to go on using Open Annotation and software components developed in CATCHPlus and CODA in future projects and hopes to be involved in Open Annotation activities after CODA ends.

2 Overview

This report is a combination of final report and the reports that CODA is committed to deliver. We start with a description of the main use cases that CODA is covering: add a layer of named entity annotations to the transcription text for scanned manuscripts, extract line strips and their boundaries from image scans and show layered annotations in a synchronized viewer web applications. We list challenges that these use cases provide for the Open Annotation model. Chapter 4 presents the representation and samples of the annotation model that we selected to cover our use cases. Chapter 5 plus the Appendices discuss the problems, issues and modeling choices concerning the OA(C) model that we had to make in detail. Chapter 6 reports about the actual execution of the CODA project and the status of its deliverables. Finally, in chapter 7, we summarize general considerations and recommendations about Open Annotation and its model.

3 Use case context and description

The next figure 1 show the overall architecture of CODA and CATCHPlus software components. Components with check marks are developed in the context of CODA. Included in the diagram is the sequence of operations implementing our first use case.

3.1 Entity detection on manually created manuscript transcription texts

The steps in this use case are:

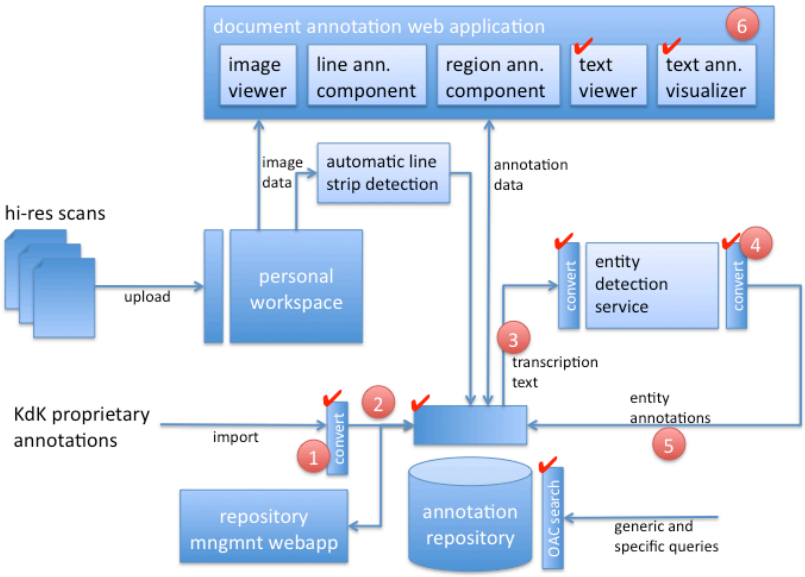

Figure 1 CODA architecture

1. *Conversion* of existing manual transcriptions for scanned pages from the Queen's Cabinet index books. Both scans and manual annotations are made available by the Dutch National Archive. The conversion is from a proprietary format used by the Artificial Intelligence group at the University of Groningen ('Monk') to OAC phase II beta format. For a discussion of this conversion, see *Appendix E, issue 1*.
2. Uploading these annotations to the CATCHPlus *Open Annotation Service* (OAS). This uploading is done using the SRU/Update protocol. *Appendix D* discusses the OAS and its design considerations.
3. Retrieval of transcription annotations from OAS using SRU/CQL and/or SPARQL. The transcription text is sent to a Named Entity Recognition service.
4. The *NER service* is a REST style web service implemented on basis of existing tools from the ILK Research Group at the University of Tilburg. 'Frog' is a morpho-syntactic analyzer and dependency parser that also generates named entity annotations. Frog is combined with a CODA OAC converter and wrapped with a web service wrapper (CLAM). Frog exports linguistic annotations complying with the FoLiA model and format. The converter maps FoLiA tokens and entity annotations to OAC phase II

format. For a discussion of the FoLiA – OAC conversion, see *Appendix E, issue 2*.

5. The resulting annotations are uploaded to and published via the OAS annotation repository.
6. Combined image, transcription and linguistic annotations for a scanned handwritten document are retrieved from OAS and displayed in *synchronized viewers*: entities can be highlighted in a document text viewer and their corresponding regions are shown in the image viewer (see section 3.3).

This use case imposes a number of non-trivial requirements on the application of the Open Annotation model:

- Layered annotations: the outcome of one annotation process (manual image transcription) is the input of another annotation process (automatic linguistic annotation of text).
- Intermediate storage between these processes is needed.
- Alternative text segmentations: both Bodies and Targets refer to different, sometimes overlapping, segments of textual resources.
- Annotation of segments of inline body text.
- Retrieval of text annotations on basis of aggregated texts from multiple targets ("the Hague", a named entity composed from text from two consecutive line transcriptions)
- Representation of sequence information
- For the transcription of scanned manuscripts there is a preference to use the SharedCanvas model.

These issues are discussed in more depth later in this report.

3.2 Automatic line strip detection as transcription aid

The original Monk software starts with preprocessing images and automatically detecting line strips: vertical coordinates indicating the top and bottom of lines. These line strips can be overlapping, for example because of line curvature or overlapping loops in the handwriting.
Lines on scanned images are best automatically detected if they are straight lines going exactly from left to right on scanned pages.
The Sailing Letters collection has many examples of pages were lines are at best straight with respect to rectangular bounding boxes that have different orientations. (see figure 2).

The objective for this use case is to help human annotators by providing them with automatically detected bounding boxes for written lines and/or cutout line strip images that can be transcribed one by one. This makes manual drawing of boxes obsolete and stimulates more fine-grained spatial alignment of transcription texts. This spatial alignment helps scholarly analysis and is also useful input for the Monk handwriting search software.

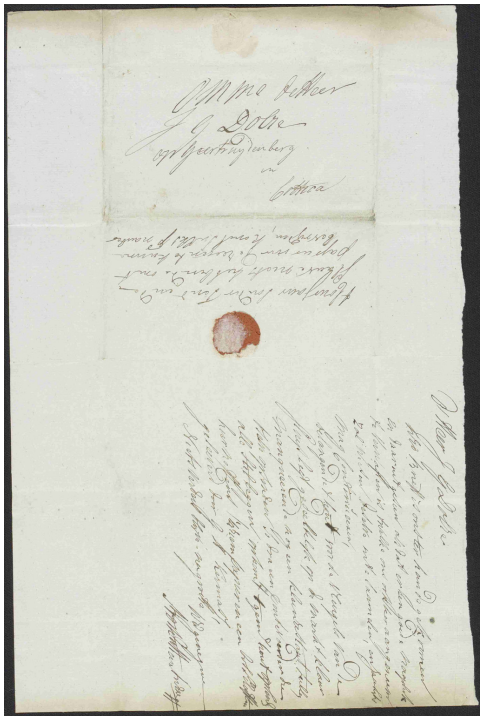

Figure 2 Sailing Letter containing several text blocks with different orientations

The workflow is as follows:

1. The user uploads and inspects a scanned image in a special web application, rotates it to the correct angles and draws bounding boxes around regions that have more or less regular lines. These boxes have position, heights and widths and an orientation angle on the page.
2. The image (URL) and the bounding box information are sent to a line detection web service that is a wrapper around existing Monk image processing software.
3. Cut out line strip images plus the coordinates of their bounding boxes are returned and stored in OAS using an Open Annotation compliant representation.
4. These images and line strip annotations can be retrieved at transcription time as starting point for creation of manual transcriptions.

This use case introduces a number of challenges for the Open Annotation model and for the SharedCanvas[1] model (Sanderson, 2011) that is an application and extension of Open Annotation:

- Constraints/Selectors are relatively complex: image regions within other image regions, sometimes with a relative rotation, non-rectangular regions.
- At several stages coordinate transformations are necessary. There are alternative (SVG) representations to encode these transformations.
- Transcription is done in two stages: therefore we need a representation for the intermediate, 'empty' annotation.

[1] http://shared-canvas.org

- We end up with a multiple images: the original scan and the cut out blocks and line strips.

3.3 Show layered annotations in a synchronized viewer

Use case 3.1 describes the creation of a complex graph of connected Annotations. When choosing an Open Annotation representation for such a graph we need requirements on basis of which we can make model decisions. Our requirements are derived from the information necessary and the queries that are needed to implement the following search and viewing environment:

1. A user searches for a specific type of named entity (persons, locations, …) or a specific named entity (Albert Einstein, The Hague, ..) in an annotation repository.
2. The system shows a list of document names that contain instances of this named entity.
3. The user selects one of these documents.
4. The system loads all annotations for this document from the repository: annotations that connect image scans to regions of a Canvas, annotations that connect transcription text to regions of that Canvas (lines or words), annotations that connect named entities to text segments of the transcription texts.
5. The system displays image, text and entities. Selecting an entity will highlight the corresponding text segments in the text view and the corresponding regions in the image view.

4 Annotation modeling choices

Implementing the use case from 3.1 requires a careful choice for the specific OAC representation that is used. Figure 3 illustrates the outcome of our modeling efforts.

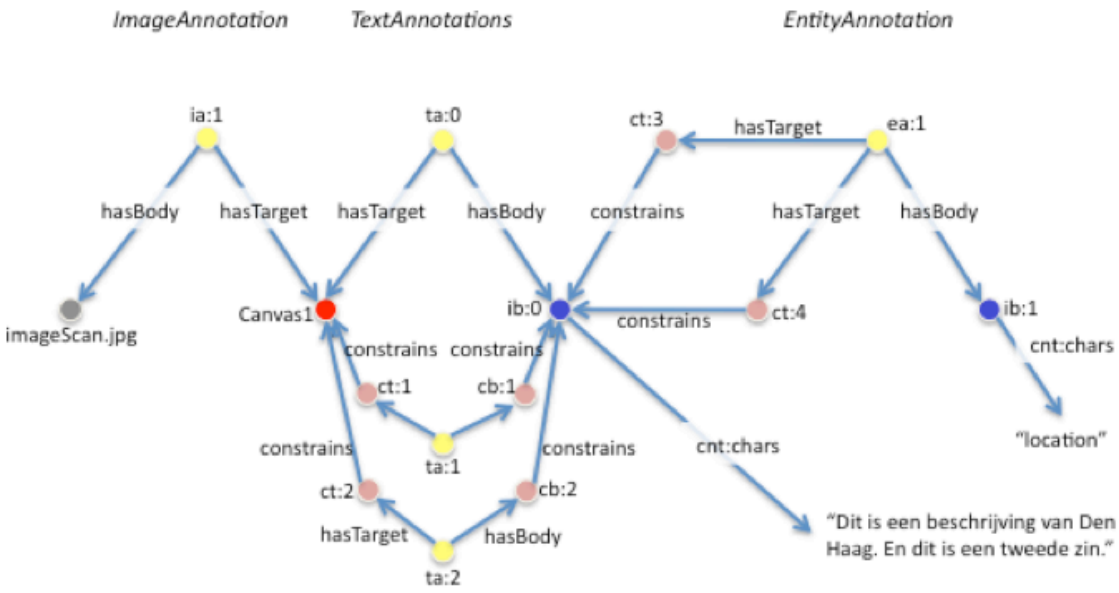

Figure 3 OAC representation of CODA use case 3.1

Canvas1 is the 'base' target for all annotations involved. Page scans or derived image representations for line strips are associated using ImageAnnotations (like *ia:1*).
Transcription texts are typically aligned with 'word zones' and/or 'line strips'. These are rectangular regions in the scanned image corresponding to a word or line in the scan. Note that these lines do not coincide with sentences in the transcription text. Since the Named Entity Recognizer uses complete sentences as its starting point we took the following approach:

- We aggregate all transcription text for a manuscript page to one text body (*ib:0*) that we associate with Canvas1 with a TextAnnotation (*ta:0*).
- Line or word zone annotations are associated with the corresponding rectangles on the Canvas with separate TextAnnotations (*ta:1*, *ta:2*).
- These TextAnnotations are associated with the proper text segment of the full text transcription in *ib:0* via ConstrainedBodies. We need a special type of Constraint for this.

The NER then uses the complete transcription text of the manuscript page, splits it into paragraphs, sentences and words/tokens using its built-in tokenizer, and adds linguistic analysis, including named entity annotations that point to one or more tokens. Tokens are represented by oac:ConstrainedTargets. EntityAnnotations associate an entity class (person, organization, location, miscellaneous) with one or more of these ConstrainedTargets.

A schematic RDF representation:

```
<ia:1>  a       oac:Annotation ;
        a       catchplus:ImageAnnotation ;
  oac:hasBody <imageScan1> ;
  oac:hasTarget <Canvas1> ;
  dcterms:creator "Hennie Brugman" .

<ta:0>  a       oac:Annotation ;
        a       catchplus:TextAnnotation ;
  oac:hasBody <ib:0> ;
  oac:hasTarget <Canvas1> ;
  dcterms:creator "Hennie Brugman" .

<ib:0>  a       oac:Body ;
        a       cnt:ContentAsText ;
  cnt:chars "Dit is een beschrijving van Den Haag. Dit is een tweede zin." ;

<ta:1>  a       oac:Annotation ;
        a       catchplus:TextAnnotation ;
  oac:hasBody <cb:1> ;
  oac:hasTarget <ct:1> .

<cb:1>  a       oac:ConstrainedBody ;
  oac:constrains <ib:0> ;
  oac:constrainedBy <c:1> .

<ct:1>  a       oac:ConstrainedTarget ;
  oac:constrains <Canvas1> ;
  oac:constrainedBy <c:2> .

<c:1>   a       oac:Constraint ;
        a       catchplus:InlineTextConstraint ;
        a       cnt:ContentAsText ;
  cnt:chars "<textsegment offset="0" range="30">" .

<c:2>   a       oac:Constraint ;
        a       catchplus:SvgConstraint ;
        a       cnt:ContentAsText ;
  dc:format "image/svg+xml" ;
  cnt:chars> "<rect x="20" y="20" width="400" height="20">" .

<ea:1>  a       oac:Annotation ;
        a       catchplus:EntityAnnotation ;
  oac:hasBody <ib:1> ;
  oac:hasTarget <ct:3> ;
  oac:hasTarget <ct:4> ;
  catchplus:chars "Den Haag" .

<ib:1>  a       oac:Body ;
        a       cnt:ContentAsText ;
  cnt:chars "location" .

<ct:3>  a       oac:ConstrainedTarget ;
  oac:constrains <ib:0> ;
  oac:constrainedBy <c:5> .

<ct:4>  a       oac:ConstrainedTarget ;
  oac:constrains <ib:0> ;
  oac:constrainedBy <c:6> .

<c:5>   a       oac:Constraint ;
        a       catchplus:InlineTextConstraint ;
        a       cnt:ContentAsText ;
  cnt:chars "<textsegment offset="27" range="3">" .

<c:6>   a       oac:Constraint ;
        a       catchplus:InlineTextConstraint ;
        a       cnt:ContentAsText ;
  cnt:chars "<textsegment offset="31" range="4">" .
```

At the beginning of CODA we have been discussing the extremes of use case 3.1 in the context of SharedCanvas with Robert Sanderson, Benjamin Albritton and

Herbert van de Sompel. Modeling solutions resulting from this discussion can be found in (Sanderson, 2011b).

For use case 3.2 (the line strip cutout service) we designed the following RDF representation for the annotations that are returned by the service. This time we used definitions from the W3C Open Annotation draft specification because this service might be useful to others using Open Annotations:

```
<textBlock>    a oa:Annotation ;
               a catchplus:LineStripRegion ;
       oa:hasTarget <textRegion>

<textRegion>   a oa:SpecificResource ;
       oa:hasSelector <svg1> ;
       oa:hasSource <canvas> .

<svg1>  a oax:SvgSelector ;
       a cnt:ContentAsText ;
       cnt:chars "<rect x="200" y="300" width="100" height="100"
                           transform="rotate(-45,0,0)"/>" .

<lineStripAnno1>       a catchplus:LineStrip ;
                       a oa:Annotation;
       oa:hasTarget <relativeLineBox> ;
       oa:hasTarget <absoluteLineBox> ;
       oa:hasTarget <polygonLineBox> .

<relativeLineBox>      a       oa:SpecificResource ;
       oa:hasSelector <svg2> ;
       oa:hasSource <textRegion> .

<svg2>  a       oax:SvgSelector ;
       a       cnt:ContentAsText ;
       cnt:chars "<rect x="0" y="0" width="100" height="50"/>" .

<absoluteLineBox>      a       oa:SpecificResource ;
       oa:hasSelector <svg3> ;
       oa:hasSource <canvas> .

<svg3>  a       oax:SvgSelector ;
       a       cnt:ContentAsText ;
       cnt:chars "<rect x="200" y="300" width="100" height="50"
                           transform="rotate(-45,0,0)/>" .

<polygonLineBox>       a       oa:SpecificResource ;
       oa:hasSelector <svg4> ;
       oa:hasSource <canvas> .

<svg4>  a       oax:SvgSelector ;
       a       cnt:ContentAsText ;
       cnt:chars       "<polygon points="200,300 270,230 305,265 235,335"/>" .
```

For the line strip service several images are relevant: the full source scan, cut out image blocks that contain more or less homogeneous handwritten lines and that are send off to the line detection image processing software and the cut out line strip images for each line. The connections with these images are represented with a number of Open Annotations:

```
<lineStripImageAnn>    a oa:Annotation ;
       oa:hasTarget <relativeLineBox> ;
       oa:hasBody <lineStripImageURI> .

<lineStripImageURI> rdf:type dcmitype:Image .

<fullImageAnn> a oa:Annotation ;
       oa:hasBody <fullImageURI> ;
       oa:hasTarget <canvas> .

<fullImageURI> rdf:type dcmitype:Image .
```

As illustration we added three alternative representations for the SpecificResource (or oac:ConstrainedTarget) that represents a line strip box: as a rectangle relative to the cut out text block that it is part of, as a rotated rectangle relative to the overall Canvas and as a polygon relative to the Canvas. Each of these representations is viable, which one is best depends on the consumer.

In our the Open Annotation representation it is not explicitly clear how to interpret coordinates and rotations. The model does not define which coordinate systems to use. We used the following heuristic for annotation of image resources:

"The coordinate axes that are used in the Selector of a SpecificResource are determined by the origin and rotation of the Source of this SpecificResource"

This holds for all three of our alternative representations.

In our practical implementation of the CODA line strip service we chose one alternative: line boxes relative to the larger text boxes that are selected from the image.

A final remark: in the (Sanderson, 2011b) paper it was suggested to represent rotated areas in an image with a 'readingAngle' property on SharedCanvas Zones. For our case we did not use Zones, but SpecificResources, and used these SpecificResources (text block regions) as Sources for another layer of SpecificResources (line strip regions). 'Rotation' is now defined as part of our SvgSelectors.

5 Technical lessons learned

Technical lessons concerning the application of Open Annotations for the CODA use cases are presented and discussed in a number of reports that are attached to this document in appendices. Most of these reports were already planned at the start of CODA and are project deliverables. On request a report about the application of SharedCanvas to our use cases is added to the appendices.

6 Project execution report

This chapter contains a report about actual execution of the project, deliverables, related activities and obstacles met. It also guides the reader in finding the actual deliverables.

6.1 Task overview

Design and implementation of Open Annotation Server
Description: in frequent 'sprint' meetings with developers of the implementing company (Seecr) many aspects of publishing, serving and querying Open Annotations were discussed and subsequently implemented.

Status: A full implementation of OAS for the core OAC specification will be finished before the end of September 2012 (on CATCHPlus budget). CATCHPlus requires production quality software that can be used in daily practice. Therefore we found it necessary to attempt to do an upgrade of OAS to the most recent W3C Open Annotation core specification (also on CATCHPlus budget). This will be done in the first half of October 2012.

When: December 2011 – October 2012

Who:

- Requirements and design – Hennie Brugman (Meertens Institute) and Erik Groeneveld and Johan Jonker (Seecr).
- Implementation – Johan Jonker, Eric Groeneveld

SDH discussions concerning SharedCanvas
Description: We explored applicability of the SharedCanvas model for the most extreme use cases of CODA, added a few extensions to SC as a result, jointly wrote a paper about it and presented that at the Supporting Digital Humanities 2011 conference in Copenhagen.

Status: paper was accepted and presented, it proved the applicability of SC for CODA cases.

When: October – November 2011

Who: Robert Sanderson, Hennie Brugman, Benjamin Albritton, Herbert van de Sompel

Testing OAS with SharedCanvas data
Description: Existing SC data from Robert Sanderson was used to test upload and query functionality of the Open Annotation Server.

Status: after minor adaptations to OAS the SharedCanvas data uploaded to OAS without errors and was sufficiently searchable.

When: March 2012

Who: Hennie Brugman

Study FoLiA format
Description: the FoLiA linguistic annotation format and the tool that creates it (frog) were discussed with their developer (Maarten van Gompel, Nijmegen University). A new version of frog that creates named entities was made available.

Status: 'frog' with entity recognition functionality is now made available from the website at ILK at Tilburg University (http://ilk.uvt.nl/frog/)

When: March 2012

Who: Hennie Brugman

Manual creation of sample data (canvas and named entities)
Description: We selected the an OAC representation for our use case 3.1 (add a layer with named entity annotations on top of transcriptions of scanned handwritten images). Test data in RDF/XML was manually created for the example shown in figure 3. This data was uploaded to OAS and queried to determine whether the chosen OAC representation was adequate for the CODA use cases.

Status: sample RDF is available. The representation is adequate for our use cases and was used as the target format for our annotation conversion software.

When: April 2012

Who: Hennie Brugman

Linguistic Annotation discussion and modeling of Named Entities
Description: for CODA the only elements of FoLiA that are relevant are words/tokens and entity annotations. However, FoLiA is an example of multi-layer linguistic annotation. Several other annotation models and formats for multi-layer annotations already exist. It is interesting to investigate in how far these models can be represented using Open Annotation, or in combination with Open Annotation. This topic was discussed in personal meetings and over email for some time with other investigators involved in linguistic annotation.

Status: The discussion stopped without joint conclusions. As an experiment a mapping from the POWLA model (Chiarcos, 2012) to OAC was made together with the designer of POWLA (Christian Chiarcos). It seemed very well possible to do such a mapping without the need for adaptations to the OAC model.

When: May 2012

Who: Hennie Brugman, Karin Verspoor, Kevin Livingston, Christian Chiarcos

Specify mapping of Monk and FoLiA to OAC

Description: we identified all elements of Monk and FoLiA that we needed to generate the OAC representation mentioned above. In the case of Monk we also did the reverse: for all information containing in Monk we identified an OAC representation.

When: April 2012

Who: Hennie Brugman

Implementation of converters
Description: A Monk-to-OAC converter and a FoLiA-to-OAC converter were programmed (using Java, the Sesame RDF library and Xpath). In case of Monk a full conversion is made, maintaining all information in Monk. In case of FoLiA only a partial conversion is made, extracting only 'words' and 'entities' from FoLiA.

Status: the source code for the converters will be made available on GitHub at the end of CODA. The FoLiA-2-OAC converter will also be made available as the core of a Named Entity Recognition web service that takes (Dutch) texts as input and returns named entities represented as Open Annotations. A variation of the Monk-to-OAC converter is part of the CODA line strip detection service.

When: June 2012

Who: Hennie Brugman

Conversion of sample data sets
Description: a demonstration subset of the Queen's Cabinet was obtained via Groningen University, preprocessed, converted and imported into OAS. Subsequently, named entity annotations were generated for this demonstration subset and also uploaded to OAS.

Status: ImageAnnotations, TextAnnotations and EntityAnnotations for the demonstration subset are available from OAS (and can be retrieved as the result of querying, or as the result of an OAI-PMH harvesting operation).

When: June 2012

Who: Hennie Brugman

Synchronized viewer prototype
Description: a proof-of-concept web application was made that uses all three types of Annotations in an integrated, synchronized image-text-entity view and search application.

Status: the viewer will be available on the web by the end of September 2012

When: June 2012 – September 2012

Who: Marc Kemps-Snijders (Meertens Institute)

Publish Sailing Letters data set
Description: the complete currently digitized set of scans for the Sailing Letters collection is made available online.

Status: available online from October 8, 2012 on.

When: September 2012

Who: Rob Zeeman (Meertens Institute)

Build line strip detection service
Description: a line strip cut out service is wrapped inside an interactive web application with OA compliant output.

Status: will made available online by Target Holding from October 2012 on

When: August - September 2012

Who: Lambert Schomaker (Groningen University), Rolf Fokkens and Minne Oostra (Target Holding, Groningen), Hennie Brugman

Finalizing software for OAC
Description: debugging, final testing, packaging software and making it available online.

Status: still going on at the moment of writing this report.

When: September – October 2012

Who: Hennie Brugman, Marc Kemps-Snijders, NN (Meertens programmer)

Upgrade software components to OA W3C spec
Description: Software components that potentially will be used by others will be upgraded from OAC to the most recent W3C OA specification to make them compliant with the emerging Open Annotation 'standard'.

Status: to be done (on CATCHPlus budget)

When: September – October 2012

Who: Hennie Brugman (Meertens), Erik Groeneveld and Johan Jonkers (Seecr)

6.2 CODA products

The next sections describe the CODA deliverables. Development on these deliverables will go on for a few weeks after finishing this report. Complete and up to date information will be made available on the CODA website as products become available: www.coda-project.org .

6.2.1 Deliverable: mappings and conversion tools

"Mapping and conversion of existing and automatically generated annotations to OAC compliant format (Queen's Cabinet annotations, line strip rectangles, internal format of the entity detection service, format of the annotation tool)"

Status: We mapped two annotation formats to OAC, Monk format and FoLiA. For both we built converters. The Monk-to-OAC converter is used to convert existing annotation for the Queen's Cabinet collection and, in an adapted and OA compliant version, for the line strip cut out service. The FoLiA converter is used to convert output of the entity detection service. Our annotation-viewing tool takes OAC annotations as input, no further conversion is required.

How to try it yourself: source code for both converters will be available on GitHub. Executable instances of the software will be available as part of the line strip cut out service and the Named Entity Recognition service that will be available online by the end of the project.

6.2.2 Deliverable: OAC compliant annotation server

"REST API with OAC compliant resource representations for the CATCHPlus annotation repository and service. The API supports storage and retrieval of annotations. "

Status: this service is implemented, tested and implied for CODA use cases. The supported version of the Open Annotation specification at the time of writing this report is OAC phase II beta. Because we feel that this annotation service potentially is useful for a much wider community, and because we intend to apply the server in future projects we reallocated some CATCHPlus money to upgrade the server to the current W3C Open Annotation core specification.

How to try it yourself: The server is based on existing, industry quality software components that are open source (Meresco[2]). The server's own sources will be made available on GitHub by the end of the project. We will also publish at least one 'one-click-install' versions to allow people to run instances of the server themselves. Finally, we will run an own instance of the server containing annotation samples from the CODA project at the Meertens Institute. Further details will be put on the CODA website.

6.2.3 Deliverable: OAC search interface

"OAC search interface (RESTful): different levels for generic and specific search for document annotations. The corresponding search user interface will be build in the context of the CATCHPlus project".

Status: the search interface is part of the Open Annotation Server software. Search can be performed at the level of Open Annotations (using SRU/CQL as query language) or at the RDF level (using SPARQL). Proprietary extensions of Open Annotations are efficiently searchable as long as they are properties with textual values. The OAS also contains a simple GUI that supports SRU/CQL and SPARQL queries.

[2] http://meresco.org

How to try it yourself: see “OAC compliant annotation server”

6.2.4 Deliverable: Reports

“Reports about:

Addressing segments of body text for further annotation
Composition of annotation Targets and Bodies
Layered annotations
Generic and specialized query interface specification (RESTful)
Application of the OAC model across CATCHPlus cases
Evaluation of SharedCanvas in the context of CODA cases”

Status: these reports are written and added to this project report in appendices.

6.2.5 Deliverable: Component for annotated text

“A component for annotated text”.

Status: a proof-of-concept interactive web application is developed that provides an integrated and synchronized view on image data, transcription text and connected named entities.

How to try it yourself: we will publish a link to this web application on the CODA website before the end of the project.

6.2.6 Line strip cutout service

Many people and projects transcribe digitized handwritten documents. It is very useful to align transcription texts with regions of the scans as much as possible. However, this spatial alignment takes a lot of time.
Monk annotations are generated starting with automatically generated cut out line strips. The CATCHPlus project provided annotation components that support efficient further alignment to word level with a few simple clicks. Transcription efficiency is very high this way, as is transcription quality.
Because automatic line strip creation can be very useful to others as well, and because we wanted to improve the line strip detection process, we developed a line strip cut out service that exports its results as a set of line strip images plus Open Annotations.
The service supports the following workflow: the user loads an image in an interactive web application. The user can define blocks of text that surround written lines, these blocks can have different rotations on the page. For each block the pixel information is sent to the actual line detector that detects and cuts out lines relative to the block. The results are combined and represented as Open Annotations, and made available for download.

Status: the core service and a RESTful API wrapper are built. Line cutting turns out to be possible with sufficient quality (of course this depends on the quality of the uploaded scan). The web application is planned to be finished before the end of the project.

How to try it out yourself: the service will be online available as part of the Monk software at Groningen University. Sources of the web application will be available from GitHub. Links to service and sources will be provided on the CODA website.

6.2.7 Website

The website www.coda-project.org was created (on basis of Wordpress) to provide information about the project and to make results available. The Meertens Institute will host this website for as long as it is relevant for the Open Annotation community. The contents will be kept up to date with information about ongoing development and potential new Meertens activities concerning Open Annotation.

6.2.8 Data sets and annotation samples

Several data sets that are relevant for CODA are made available online or will be by the end of the project:

Sailing letters scans: all currently available scans of the Sailing Letters collection will be made available online by the Meertens Institute at an official launch event on October 8, 2012.
Sailing letters annotations: the current collection is fully transcribed on a page-by-page basis. These page transcriptions may be aligned at line or even word level at a later stage using software developed in CODA. These (proprietary) annotations are not yet available online, but might be in the future.

To demonstrate CODA results a set of 5 scans from the Queen's Cabinet collection is available online, together with their line strip and word zone annotations as they are converted to OA(C). Also, the connected entity annotations are made available. All annotations for this demonstration collection are available from the OAS annotation server. More samples may be added in the future.

Links to all online datasets and annotations are provided on the CODA website.

6.2.9 Source code

All project source code will be published on github by the end of the period of performance. Names and URLs of the relevant Github repositories will be published on the CODA website, once they are available.

6.3 Publications and presentations

Sanderson, R. Brugman, H. Albritton, B. Van de Sompel, H. (2011.) *"Evaluating the SharedCanvas Manuscript Data Model in CATCHPlus"*, Supporting Digital Humanities 2011, Copenhagen, Denmark, November 2011. arXiv:1110.3687.

Brugman, H. (2012) *"A Publication Platform for Open Annotations",* ISA-7 Workshop on Interoperable Semantic Annotation at LREC 2012, Istanbul, Turkey, May 2012.

Brugman, H. (July 26-27, 2012). Presentation at the OAC Phase II review meeting, Chicago.

6.4 Obstacles

We did not experience unforeseen technical problems with software development or 'show stopping' problems with applying the OAC annotation model. We found some limitations of the model, but were able to overcome or by-pass those. These issues are discussed in detail elsewhere in this report. Communication with our contacts at University of Illinois at Urbana-Champaign and other members of the Open Annotation community was frequent and effective.

The delayed start of the OAC phase II experiment projects caused some hindrance. The project execution period now lined up with a particularly busy period for the technical development department of the Meertens Institute. It was therefore difficult to organize the required programming capacity. Also, alignment of CODA activities at the Meertens Institute and Groningen University was a bit more difficult.

Finally, the merger of the OAC phase II beta model with Annotation Ontology caused some troubles. Although this could in principle be ignored for the CODA project itself, it had consequences for the usability of the software we developed, especially for the CATCHPlus Open Annotation Service. Luckily, we found some room in the CATCHPlus budget to invest in a version upgrade of OAS.

7 Generalizable Results and Conclusions

This chapter summarizes the main general conclusions and recommendations. Almost all of them are discussed at length elsewhere in this document and its appendices.

- For transcription of documents (scans, audio/video recordings) it is a good strategy to aggregate text segments to a full document text and align this text with (spatial or temporal) segments of the document using Annotations with ConstrainedBodies. In this way sequence information is maintained and independent alternative text segmentations are supported.
- We recommend to add an InlineTextConstraint or InlineTextOffsetSelector to the OA extensions document. This Constraint/Selector should make it possible to select segments of text in inline text Bodies or Targets.
- Although for our use cases we were able to find a workaround, it would be useful to have a general recommendation about how to deal with Sequences of Annotations. For example, the approach taken by the SharedCanvas model could be generalized.
- In several cases we used segmentations of segmentations of Resources. In these cases we recommend
    - to identify the segments (ConstrainedTargets, SpecificResources) with resolvable http URIs
    - to use coordinate systems that are defined by the Source of the SpecificResource in question (for example, define line strip segments using coordinates relative to the text block that they are part of. This block can be shifted or rotated with respect to the document Canvas)
- Conversion of proprietary annotation formats to Open Annotation can have many benefits
    - Easier publication of annotations on the web
    - Often information that is implicitly available is made explicit
    - Open Annotations link related resources together. This may lead to better data organization
    - New types of queries are supported
    - Your annotations can be combined with other people's annotations, or even be further annotated by other people
- Most annotations have textual Bodies. There are some issues concerning search on basis of these texts:
    - Bodies can be (parts of) external documents. To be able to search for these annotations the external texts need to be resolved, retrieved and possibly indexed first.
    - Bodies can be segments of larger texts. To be able to search for these annotations the Constraints/Selectors have to be interpreted first, and the relevant text segments have to be retrieved and possibly indexed first.

- We would like to see a recommendation in the spec about how to represent Sets of Annotations.
- Operations on complete Annotation resources require that it is exactly clear what the boundaries of an Annotation are, so that it is clear what triples are affected. We recommend that the spec defines this boundary (compare this to 'Concise Bounded Description' - http://www.w3.org/Submission/CBD/)
- Publication of annotations on a server platform requires that this platform assigns resolve http URIs where they do not exist, and that there exists a resolver that is able to resolve those URIs.
- Open Annotation represented in RDF is a very verbose way to represent annotations. When converting proprietary annotations to OA this often turns compact, human readable data into large, inaccessible files. This may have a serious negative impact on the acceptance of Open Annotation as a standard.
- Good and easy to use software and especially software libraries may be very important to stimulate acceptance of Open Annotation. This software should not only facilitate consumption of Open Annotations, but also their production.

Appendix A: Addressing segments of (body) text

Problem description

Chapter 4 shows our representation of a complex network of Annotations. Some issues arose and had to be dealt with:

1. EntityAnnotations have ConstrainedTargets that refer to text segments that are part of inline text from the (published) Bodies of TextAnnotations. The specification deals with segments of textual resources, not with segments of inline Body text.
2. Annotations with textual bodies or targets typically have to be searchable on basis of this text. However, this is not straightforward when ConstrainedBodies or ConstrainedTargets are used.
    a. ConstrainedTarget: the annotated text segment cannot be derived just by resolving the Target resource URI, it requires resolving the Constraint. Example: find all 'location' type named entities referring to the text "Den Haag".
    b. ConstrainedBody: the body annotation text cannot be derived just by resolving the Body resource URI, also in this case the Constraint has to be resolved as well. Example: find all line transcriptions that contain "Amsterdam".

Discussion

Issue 1: To support annotation of text in Annotation Bodies we introduced a new type of Constraint (InlineTextConstraint) that is closely related to oax:TextOffsetSelector. It also uses 'offset' and 'range' to indicate a segment, but in this case in the cnt:chars value of inline text represented as a ContentAsText resource.

```
<rdf:Description rdf:about="urn:id:cb:1">
    <rdf:type rdf:resource="http://www.openannotation.org/ns/ConstrainedBody"/>
    <oac:constrains rdf:resource="http://oas.dev.seecr.nl:8000/resolve/urn:id:ib:0"/>
    <oac:constrainedBy rdf:resource="urn:id:c:1"/>
</rdf:Description>

<rdf:Description rdf:about="urn:id:c:1">
    <rdf:type rdf:resource="http://www.openannotation.org/ns/Constraint"/>
    <rdf:type rdf:resource="http://catchplus.nl/annotation/InlineTextConstraint"/>
    <rdf:type rdf:resource="http://www.w3.org/2008/content#ContentAsText"/>
    <cnt:chars>"&lt;textsegment offset="0" range="30"/&gt;"</cnt:chars>
    <cnt:characterEncoding>UTF-8</cnt:characterEncoding>
</rdf:Description>
```

Issue 2: In the CODA case queries are handled by the CATCHPlus Open Annotation Service that is based on a combination of an RDF Store and Apache Solr. Textual properties are indexed by the OAS at the time of uploading annotations. Since OAS is intended to support only the OAC core model it does not interpret text constraints. Therefore, we pragmatically chose to duplicate the relevant search text for an annotation to a proprietary property of oac:Annotation. In the case of our example, we added a 'catchplus:chars' property containing the texts of the constituent tokens "Den" and "Haag" (in the correct order).

Appendix B: Composition of annotation Targets and Bodies

Problem description

Our starting point for the use case described in section 3.1 was a set of proprietary transcription annotations for line strips and or word zones. Line strip transcriptions have sequence numbers per page and spatial coordinates for top and bottom. Word zone transcriptions are tied to segments of the scanned images using rectangular boxes relative to the lines they belong to.
Next to this spatial composition there exists a textual composition that is especially relevant for the subsequent Named Entity Recognition process. This process presupposes document texts that consist of a paragraphs that can be broken up in sentences and tokens. However, sentences or named entities can cross line boundaries in a document scan and can therefore not be based on decomposition of text on basis of written lines on a page.

There were a number of issues concerning spatial and textual composition that we had to deal with:

1. There was no document text to start with, we had to construct it from the underlying transcription annotations, the sequence of lines and the order of words within each line.
2. There was no uniform way to represent sequence of the various text chunks.
3. We wanted to avoid too much redundancy of information.
4. One of our requirements is that it should be possible to determine overlap of detected named entities (or sentences) with line and word zone transcription texts to be able to highlight them in the context of the scanned image. This textual overlap is specified in terms of character offsets and ranges, which initially were not available for lines and word zones.
5. Relative coordinates with respect to line strips had to be transformed to absolute coordinates with respect to a Canvas, to comply with the SharedCanvas model.

Discussion

Issue 1: While converting the proprietary line and word zone transcriptions we kept track of ordering. On basis of this ordering we generated additional Open Annotations: one at the 'page' level and others, where missing, at the line level. For these annotations we also constructed a rectangular bounding box on basis of the spatial information of their component annotations. The additional page level annotation provides the full document text for subsequent entity recognition.

Issue 2: This page level annotation also uniformly stores the order of line and word zone texts in its Body: all line and word zone texts are represented as segments within this page level annotation text using character offsets.

Issue 3: We chose not to duplicate texts at page, line and word levels but instead use ConstrainedBodies constraining the full text at page level. (However, to support searchability we later had to add redundant text for text indexing – see section 6.1, issue 2).

Issue 4: The page level text is segmented in lines and words by means of character offsets with respect to this full text. The same page level text can be alternatively segmented by our Named Entity Recognition service. The first step of this NER service is decomposing the text into sentences and words (using sentence splitting and tokenizing) within these sentences. This decomposition is represented using character offsets with respect to the full text as well. Named Entities have one or more tokens as their Target. Overlap of Named Entities with line strips or word zones is now a matter of comparison of character offsets.

Issue 5: Coordinate transformations were a simple matter of combining available spatial information about lines and word zones.

Appendix C: Layered annotations

Problem description

Named Entity Recognition is a good example of a common practice in the domain of linguistic annotation: use 'annotation pipelines' to stack 'layers' of annotations on top of each other. Starting point is usually a text document (written, or the transcription of a recorded speech act). Several annotation modules then add analysis in the form of annotations, sometimes attaching the annotations to the text, sometimes to annotations from previous annotation steps.
For the use case from section 3.1 this is also the case. The starting point is the full transcription text for a scanned document. Our 'annotation pipeline' (called 'frog'[3]) consecutively does sentence splitting, tokenization, morpho-syntactic and dependency analysis and named entity recognition, adding the results each time as additional annotations.
There are cases where multiple annotations are associated with a previous annotation, in a specific *order* (order of tokens in case of NE recognition, order of morphemes in case of morphological decomposition). For visualization and query purposes it is important that this sequence information is maintained when converting to Open Annotations.

Summarizing, we had to deal with the following issues:

1. Representation of sets of Annotations, corresponding to documents, or to annotation layers within a document. It is very useful to be able to select sets of Annotations for retrieval, updating, deleting or to restrict queries to.
2. Sequences of annotations. Our use cases require that sequence information is maintained after conversion to Open Annotations.
3. Using sub classes of oac:Annotation to differentiate between types of Annotations.

Discussion

Issue 1: Strictly spoken, support for Annotation sets is a requirement for annotation servers or annotation exchange protocols, not for the annotation model itself. Furthermore, the OAC model implicitly supports definition of Annotation sets by means of a combination of constraints on property values and hasBody/hasTarget relations. Example: the set of line and word transcription annotations for a specific document can be retrieved by a Sparql query like:

```
PREFIX oac: <http://www.openannotation.org/ns/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?anno WHERE
{
      ?anno rdf:type <http://www.catchplus.nl/annotation/TextAnnotation> .
      ?anno oac:hasTarget <http://www.catchplus.nl/annotation/NL_HaNa_H2_7823_0057>
}
```

[3] http://ilk.uvt.nl/frog/

Nevertheless, an explicit provision or best practice recommendation for sets as part of the OA specification would be very practical and provide a standardized way to define sets (similar to named graphs in RDF).
Our, pragmatic, solution was to support sets at our annotation server, where they coincide with OAI sets (being harvestable and deletable units).

Issue 2: In the CODA case we need the order of word and line annotations to construct and display a full document text (with different text segmentations) and to properly represent the text of multiword named entities ("The Hague" is unequal to "Hague The"). In more elaborate use cases it may also be desirable to be able to search for sequences (a "location" immediately followed by a "person").

For the conversion of Monk line strip and word zone annotations we used available ordering information (line numbers in combination with horizontal positions within lines) to construct an additional annotation containing the full-page text and attached the OAC equivalents of line and word zone annotations to segments of this full-page text. In this way, the full text provides the ordering information. This is sufficient for visualization purposes. However, to be able to search for specific lines or word zones we had duplicate text (see Appendix A, issue 2). If sequence information would have been represented explicitly in OA we could have constructed the full-page text at display time.

For conversion of the linguistic annotations from the output of the 'frog' linguistic analyzer we used a comparable approach. The output XML format (called FoLiA[4]) contains hierarchical sequence numbers for paragraphs, sentences and tokens. Relevant to maintain in the case of Named Entity Recognition is the order of tokens in multi-token named entities (represented by us as multiple Targets). Also in this case we additionally represented the sequences by adding redundant text, this time in a properly associated with our EntityAnnotations.

Although we managed to work around the absence of a representation for annotation sequence in OAC in general it would be useful if there existed at least a best practice recommendation. Possibly such a recommendation can be based on the solution taken for SharedCanvas: a combination of rdf:Lists with ore:Aggregations.

Issue 3: In line with the SharedCanvas model we used sub classes of oac:Annotation to differentiate between Body data types. The main reason was that we want to be able to retrieve those annotations by type: "return all EntityAnnotations for document A". From the W3C Open Annotation spec and later discussions it became clear that there are alternative ways to use sub classes of oa:Annotation, e.g. to express motivations. For future work, we now prefer to represent Body data types using an additional rdf:type property on the Body: `<bodyURI> rdf:type dcmitype:Image .`

[4] http://ilk.uvt.nl/folia/

Appendix D: Open Annotation Server and annotation queries

Problem description

The workflows in our use cases require storage and retrieval of annotations at several stages. Also, we need a publication and exploitation platform for resulting annotation sets.

In the CATCH program work was done on a generic RDF based annotation repository and a model was developed (Brugman et al, 2008) for that. The main aim was to provide a shared publication and exchange platform for heterogeneous annotations of resources and resource fragments from cultural heritage collections of participating institutions. In the CATCHPlus project a robust version of this Annotation Repository is one of the deliverables.

Given the similar requirements and approach, the large community behind Open Annotation and the quality of the OAC model at the start of CODA it was decided to completely base the CATCHPlus Annotation Repository on the Open Annotation model. Parallel to CODA, CATCHPlus developed its Open Annotation Server (OAS). Development work was done by a small and highly specialized company, Seecr[5].

Objective was to create an annotation server that fully supports the OAC core model. Supported functionality includes upload/ingest of batches of Open Annotations (embedded in RDF files), efficient queries on Dublin Core properties and all of the OAC core classes and properties, exchange of sets of annotations using the OAI-PMH protocol, a SPARQL endpoint, web publication of annotations according to linked data principles, including a built-in resolver. Annotations are served complying to the OAC model and have all associated information inlined. The OAS has a simple interactive Dashboard for management of users, annotation sets and OAI-PMH harvesting jobs.

The OAS is implemented on basis of an industry quality software component suite (Meresco) that provides functionality for SRU/Update, SRU/CQL, OAI-PMH data providing and harvesting and efficient indexing modules for efficient search. The internals of OAS include an RDF Store (OWLIM) and Apache SOLR for fast and scalable text search.

OAS is easily installable and configurable. Sources are available on Github (see the CODA website for details). For a more detailed discussion of the Open Annotation Service see (Brugman, 2012).

During OAS development, the CODA use cases were the most challenging ones used to test OAS. Many interesting issues concerning publication, sharing and retrieving Open Annotations showed up and were discussed:

[5] http://seecr.nl

1. For ingesting, updating, deleting annotations and for returning results decisions have to be made about the 'boundaries' of an Annotation in an RDF graph.
2. What are the 'fields' that have to be indexed for efficient querying?
3. How to store and search SharedCanvas data in OAS (that only supports 'core' OAC).
4. Publication requires resolvable http URIs for resources that often do not have those when uploaded. Which resources require resolvable http URIs? How do all sorts of identifiers of incoming annotations and their components have to be processed?
5. Searching is usually done on basis of text. In many cases this text is not part of the uploaded annotation (external bodies, external targets, foaf profiles).
6. What queries have to be supported to implement our use cases?
7. There is a need for 'annotation sets'.
8. Search on ad hoc collections of annotations ("harvest and search"). How to compose those ad hoc collections?
9. How to deal with subclasses and rdf type inferencing?

Discussion

Issue 1: (annotation boundaries) Although annotations in OAS are represented as RDF, operations are defined on Annotations, not on triples. For the usual (C)RUD operations (create, read, update, delete) this means that all triples that 'belong to' an annotation are returned by, substituted in or removed from the OAS. Therefore, we had to make clear decisions about where the boundaries of an Annotation are in its enclosing RDF container. We had to take into account that Bodies or Targets can be shared by multiple Annotations, and that Annotations or Annotation Body text can be the Target of other Annotations.
We used the following strategy for ingesting Annotations: starting at each Annotation resource in an RDF file, every resource that is connected to it by an OAC property belongs to the Annotation, except when this resource is itself an Annotation. All triples connected to these resources via OAC, DC or any other, properties, are also considered part of the Annotation and stored in OAS. All other triples are ignored.
When serving Annotations, for example as results of a query, we use a similar strategy to collect all of the Annotation's triples and return them in a uniform way. So, all information contained within the Annotation boundaries is inlined in the result.
When deleting or replacing Annotations we take into account that there can be more than one Annotation for a given Body or Target.

Issue 2: (indexes) The most generic and basic way to search in OAS is at the RDF level, using SPARQL. Being an annotation server we added fast and scalable search at the level of OAC Annotations as well. Using SRU/CQL as query language it is possible to search on values of oac:hasBody, oac:hasTarget, rdf:type and a number of Dublin Core properties. Also, it is possible to search on text contained in oac:Bodies or for any text anywhere in an Annotation including in proprietary property values.

In cases where Annotations contain external Bodies, Targets or foaf profiles OAS tries to resolve the URI to these external resources a couple of times. If successful and if the resource is textual or XML this text is inlined and indexed to be able to search on it.
All mentioned fields are made searchable on full text by indexing it (using SOLR).

Issue 3: (SharedCanvas) One test case was importing existing SharedCanvas sample RDF documents (found on http://shared-canvas.org) into OAS. Import turned out to be straightforward. SharedCanvas uses OAC Annotations to lay out image and text on a Canvas. It also introduces a number of classes and properties of its own, e.g. to represent Sequences and Ranges of Canvases, or ordered aggregations of Annotations. The ingest strategy described under issue 1 picks out the Annotations and all associated resources and properties as expected (following 'outgoing' triples), and ignores the additional constructs concerning sequences and aggregations ('incoming' triples). So OA does not offer full support for SharedCanvas, but it supports important parts of it. It is very well possible to use OAS in combination with a Canvas-like approach to express the layout of images and texts using annotations.

Issue 4: (identifier management and publication)
At ingest time or when harvesting data several types of resource URIs can be present in the imported Annotations: anonymous (no identifier), URN or URL. Also, in the OAC model there are several types of resources that have identifiers: Annotation, Body, Target, Constraint, ConstrainedBody and ConstrainedTarget. With exception of ConstrainedBody and Constraint our use cases require that it is possible to refer to instances of all of these resource types, internally (annotations exist in the same instance of the Annotation Server) or externally (over the web). An example is the case of text tokens that are used to attach linguistic annotations to and that we modeled as ConstrainedTargets representing segments of a text. Another example is the Annotation Body that contains the full text of our scanned documents and that we use as the Target for subsequent named entity detection.
Therefore, we implemented identifier-processing algorithms for each of these resource types (see figure 4).

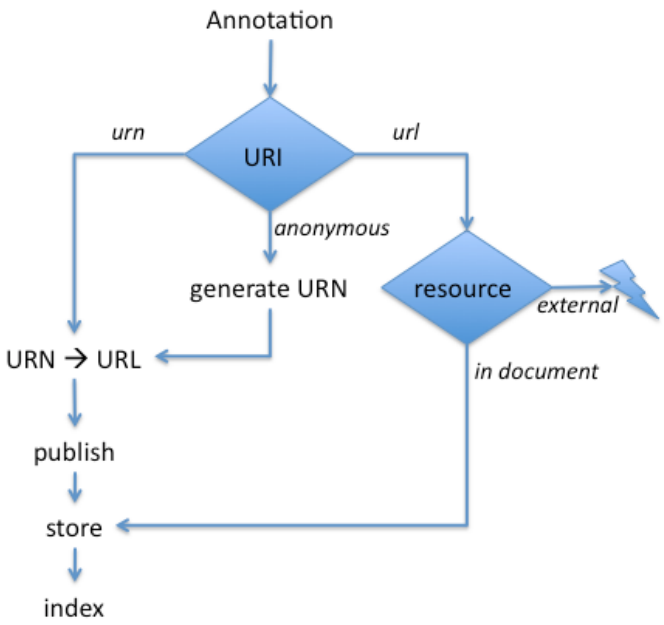

Figure 4 Processing identifiers for oac:Annotation

Figure 4 shows this algorithm for the case of oac:Annotations. Each incoming Annotation ends up with a resolvable http URL. All URLs that are not referring to resources that are internal to the uploaded document are stored and indexed. OAS includes a resolver that resolves http URLs for all resources published by the OAS.
Where present original proprietary identifiers are associated with the published resources by means of an additional dc:identifier property.

Issue 5: (inlining and indexing text) As part of the ingest algorithm the OAS tries to resolve external bodies, external targets and extern foaf profiles three times. If successful and if the resource is a text or XML document the contained text is retrieved and indexed. This makes it possible to search on text values that are not internal to the uploaded Annotations.

Issue 6: (scope of query language) We added full text indexes to the OAS to support a number of queries that are obvious in the Annotation domain. Next to an index for 'hasBody', that allows efficient search for Body URIs we added an index for text inside bodies that enables fast search for Annotations annotated with some text value. Another obvious query is for all Annotations that have a specific resource as Target, directly or via ConstainedTargets ("find all Annotations that have Canvas1 as Target"). The 'hasTarget' index also includes all Annotations that are connected via intermediate ConstrainedTargets to make these queries possible (bypassing oac:constrains triples).

Issue 7: (annotation sets) See Appendix C, issue 1 for a discussion. In OAS annotation sets coincide with OAI-PMH sets. It is possible to restrict SRU/CQL queries to specific Annotation sets ("search only in annotations for a given scanned document, for a given collection, …").

Only the administrator of an OAS instance is capable of creating new Annotation sets. Each set is associated with an API key so that each set can have a different 'owner' with ingest rights. 'Batch delete' of Annotations can only be done on a set-by-set basis by the OAS server instance administrator.

Issue 8: (harvest and search) The CATCHPlus Open Annotation Server is not primarily intended as a one-instance central annotation repository. It is designed to be used at different levels: by individuals, project groups, institutions or inter-institutional collaborations. This requires that OAS is easy to install and to administer, which is the case. Additionally, OAS supports peer-to-peer operation. Different instances of OAS can exchange sets of annotations using the built-in OAI-PMH data provider and harvester. Exchange is on a set-by-set basis.
The idea is that all annotations relevant for a given project or task are harvested to one place, where they are indexed. In this way efficient cross-collection searches are possible (as opposed to distributed queries). This strategy is also used often for linked data applications.

Issue 9: (subclasses and rdf inferencing) For our use cases we decided to differentiate between different types of annotations: ImageAnnotations and TextAnnotation as in SharedCanvas, MonkAnnotations to represent annotations that are converted from Monk line strip and word zone transcriptions, EntityAnnotations to represent named entities and LineStrips for automatically generated bounding boxes around lines of texts in scanned documents. The Open Annotation Server indexes any additional rdf:type property associated with an Annotation and therefore supports search on rdf:type as part of its SRU/CQL interface. It is not required that these additional types are sub types of oac:Annotation, OAS does not have to do any type inferencing. We did not explicitly define our Annotation types as sub types of oac:Annotation, but our naming convention suggests otherwise.
The recent W3C Open Annotation specification introduces recommendations concerning the use of subtypes of oa:Annotation. The recommended use of sub types of oa:Annotation is to represent the reasons why an Annotation was created. Our use of types is mainly to indicate the data type of the Annotation's Body (Text, Image and EntityAnnotations), but also origin (Monk annotation set) and type of Target (LineStrip).
A good alternative for the 'body type' annotation sub types is to represent them as properties or types of the Body. Origin is best represented using the new oa:annotator or oa:generator properties, or with a proprietary property on the Annotation.

Appendix E: Application of the OAC model across CATCHPlus cases

Problem description

There are two cases of existing annotation formats/models that are directly relevant for the CODA use cases: proprietary annotations in so-called 'Monk' format that exist for scanned document images in the Queen's Cabinet collection and linguistic annotations using the FoLiA format. The latter is the proprietary annotation format that is exported by the 'frog' linguistic analysis and named entity recognition software. As part of CODA we defined mappings of these formats to OAC and implemented converters.

1. Monk conversion (in relation to SharedCanvas)
2. FoLiA conversion (and linguistic annotation in general)

Discussion

Issue 1: For the conversion of Monk annotations we chose an OAC representation that is on the one hand compliant with SharedCanvas, and on the other hand supports our requirements of layered annotation, annotation of segments of inline body text and alternative segmentations of document transcription text. For a discussion of the exact representation chosen see Chapter 4. We chose for a complete mapping of Monk to OAC: all information is maintained in the Open Annotation result. Furthermore, information implicit in the Monk format was made explicit in the resulting OAC representation:

- text segment information (offset, range) with respect to the full-page text.
- boundaries of rectangles enclosing all word zones that are on the same line, in case of absence a line strip annotation for this line.
- absolute spatial coordinates of annotation targets with respect to the Canvas
- dimensions of the enclosing Canvas (to be used when Canvas dimensions are not defined otherwise, for example by the dimensions of the corresponding image)
- association with the image scan, and optional cut out text block images or line strip images.

A number of observations about Monk conversion:

- Conversion turned out to be relatively complex. Programming the converter took more time than expected. This was mainly because of the complex OAC representation chosen.
- 20 lines of Monk annotation data resulted in over a thousand RDF triples. Partly this is because we made implicit information explicit and because we added DC and proprietary properties to facilitate human inspection of the RDF result and search. Still, most annotation communities are not RDF-oriented. This 'blowing up' of data may prevent adoption of Open Annotation by these communities.

- The resulting OAC annotations connect related resources. One or more images (full scan and cut-out line strip images), line strip and word zone annotations are now all connected through the same Canvas.
- This, in combination with the implicit information that is made explicit, allows new types of queries compared to the proprietary Monk format.
- Because we store converted Monk annotations in the CATCHPlus Open Annotation server (that only supports core OAC classes and properties) we did not convert to a complete SharedCanvas representation. However, we used Canvas, ImageAnnotation and TextAnnotation classes from the SC model. Sequence information for Annotations is represented by concatenating all annotation texts in order and creating an additional annotation that associates this text with the Canvas.
- Unlike the proprietary Monk format the OAC representation chosen for our conversion allows multiple alternative text segmentations (lines, words, paragraphs, sentences, word tokens).
- To be able to search for words or lines that contain some text we had to duplicate the relevant text segment from the ConstrainedBody to a proprietary property. Otherwise, it would have been necessary that the Open Annotation Server interprets the Constraints of a ConstrainedBody at query time (which implies different strategies for all kinds of Constraints and is outside the scope of the core OAC model).
- We had to introduce a new type of Constraint that we called InlineTextConstraint. This is a variation on oax:TextOffsetSelector where 'offset' and 'range' now refer to text enclosed in a ContentAsText object. Both the OAC annotations for line strips/word zones and EntityAnnotations resulting from the Named Entity Recognition process use InlineTextConstraints.

Issue 2: (FoLiA to OAC conversion)
The CODA Named Entity Recognition Service is based on the 'frog' linguistic analysis tool from ILK at Tilburg University. Frog is a command line based tool that takes plain text as input and returns results in its own FoLiA annotation format. FoLiA is a format to represent linguistic annotations. It is an XML format that partly inlines annotations and is partly standoff. The input text is hierarchically decomposed into paragraphs, sentences and words, each element has a hierarchical identifier that encodes the level and the relative sequence number within that level (e.g. xml:id="docname.p.1.s.4.w.1"). Each word element contains extensive part-of-speech annotation, lemma and a morphological decomposition. This decomposition refers to parts of the word text whereby order of morphemes is represented using relative character offsets.
Additional to this textual decomposition syntactic analysis and dependency structure are encoded in FoLiA using references to word ids. Finally, two alternative sets of named entities are present, also encoded on basis of references to words/tokens, where the order of tokens is relevant. An example:

```
<entity class="per" confidence="0" set="http://ilk.uvt.nl/folia/sets/frog-ner-nl">
     <wref id="untitled.p.1.s.4.w.7" t="L.G.A."/>
     <wref id="untitled.p.1.s.4.w.8" t="Vos"/>
</entity>
```

For conversion of FoLiA there are two alternatives:

1. Full conversion of all linguistic annotations to OAC. During CODA discussions with others in the Open Annotation community took place, comparing alternative approaches and existing standards (like the ISO Linguistic Annotation Format, (Ide, 2007)).
2. Conversion of tokens and entity annotations only. This is actually implemented in conversion software that is part of the CODA NER Service.

Implementation of partial converter
Implementation of this converter was relatively easy and straightforward. The converter is based on a combination of Xpath and an RDF library (Sesame[6]). It takes a FoLiA XML file and a URL to a Body or Target containing a ContentAsText object in OAS as input. Tokens in FoLiA are represented in OAS as ConstrainedTargets with this URL as value for the oac:constrains property. FoLiA lacks character offset information with respect to the source text, so our converter had to derive that on basis of comparison of source text and FoLiA document. Again, our InlineTextConstraint was used to indicate the relevant text segments in the source text.
The Named Entities themselves are encoded as EntityAnnotations that have one or more tokens as Target. The Body of these Annotations contains the entity class (per(son), loc(ation), org(anization), misc(ellaneous)). Finally, a proprietary 'chars' property is added to each named entity to encode the sequence of tokens and to make the entities searchable ("find the hague" versus "find hague the").

[6] http://www.openrdf.org/

Appendix F: Application of SharedCanvas

From the start of CODA it was an objective to check the applicability of SharedCanvas to the CODA use cases. We did this exploration in two ways:

1. We identified extreme use cases by analyzing the CODA use cases, discussed them with Robert Sanderson, Herbert van de Sompel and Benjamin Albritton, wrote a joint paper (Sanderson, 2011b) about our conclusions and presented this at the SDH 2011 conference. We refer to the paper itself for the conclusions. In general, it turned out to be very well possible to address the needs of our use cases with SharedCanvas. Only in a few cases minor improvements/extensions to SC were introduced.
2. We did an actual conversion of existing proprietary annotations for the scanned handwritten documents of the Queen's Cabinet collection, compliant with SharedCanvas principles but restricted by the capabilities of our OAS Open Annotation Server (that by choice only supports core OAC).

Discussion

For our use cases we did not need the full range of SharedCanvas classes and properties to start with. For example Ranges, Lists and Manifests are outside their scope. The classes and properties that we did need (including new classes and properties suggested in the SDH paper) are not part of the core OAC model, and are therefore not explicitly supported by the OAS server. So, our actual experiments with SharedCanvas were not so much about applying its classes and properties, but more about finding alternatives for them that are part of the core OAC model.

- SC uses Zones to provide objects with dimensions that can be associated with a Canvas and can themselves be annotated. For example, in our case we would use Zones to model an image region containing one or more lines of handwritten text. Instead we used ConstrainedTargets/SpecificResources for this, in combination with SvgConstraints/SvgSelectors. Instead of adding a 'readingAngle' property to Zones, we specified the rotation as part of the SVG expression in the Constraint. See Chapter 4 for a discussion of this.
- We represented Sequence information by concatenating annotation text segments and using ConstrainedBodies and ConstrainedTargets that constrain the concatenated text. The sequence information is implicitly stored by the concatenated text. This is very case specific solution and does not make SC Sequences obsolete in any way. On the contrary, there might be several other annotation use cases that require Sequences outside the domain of SharedCanvas.

References

Brugman, H. Malaise, V. & Hollink, L. (2008) A Common Multimedia Annotation Framework for Cross Linking Cultural Heritage Digital Collections, In Procs of 6th International Conference of Language Resources and Evaluation, Marrakech, Morocco

Brugman, H. (2012) A Publication Platform for Open Annotations, ISA-7 Workshop on Interoperable Semantic Annotation at LREC 2012, Istanbul, Turkey, May 2012.

Chiarcos, C. (2012) POWLA: Modeling linguistic corpora in OWL/DL. In: E. Simperl et al. (eds.) Proceedings of the 9th Extended Semantic Web Conference (ESWC 2012). Springer, Heidelberg, Heraklion, Crete, May 2012.

Ide, N., Romary, L. (2007). Towards International Standards for Language Resources. In Dybkjaer, L., Hemsen, H., Minker, W. (Eds.), Evaluation of Text and Speech Systems, Springer, 263-84.

Sanderson, R. et al. (2011). SharedCanvas: A Collaborative Model for Medieval Manuscript Layout Dissemination, In Procs of 11th Joint Conference of Digital Libraries, Ottawa, Canada http://www.arxiv.org/abs/1104.2925

Sanderson, R. Brugman, H. Albritton, B. Van de Sompel, H. (2011.) "Evaluating the SharedCanvas Manuscript Data Model in CATCHPlus", Supporting Digital Humanities 2011, Copenhagen, Denmark, November 2011. arXiv:1110.3687