PAGE XML Update Ideas

Christian Clausner, 11/07/2013

Contents

[Drop Capitals (#945) 2](#_Toc361304966)

[Polygons (#944) ✓ 2](#_Toc361304967)

[Confidence Value (#980, #994) ✓ 3](#_Toc361304968)

[Named Reading Order Groups and Layers (#207, #411) ✓ 3](#_Toc361304969)

[Text Production Type (#899) 3](#_Toc361304970)

[Text Style (#331, #958) 4](#_Toc361304971)

[Language at Text Line and Word Level (#230) 5](#_Toc361304972)

[New Region Types (#239, #240) 5](#_Toc361304973)

[New Sub-Types and Languages (#235, #925, #941) 5](#_Toc361304974)

[Notes/Comments/Custom Types (#956) 6](#_Toc361304975)

[Alternative Images (#452) 6](#_Toc361304976)

[Extended Page Element (#138) 6](#_Toc361304977)

[Baseline for text lines (#957) ✓ 6](#_Toc361304978)

[Plain Text Optional (#276) ✓ 7](#_Toc361304979)

[Named Entities (#992) 7](#_Toc361304980)

[Nested Regions (#600) 7](#_Toc361304981)

[Unicode Element for Non-Text Regions (#394) ✓ 8](#_Toc361304982)

# Drop Capitals (#945)

Problem:

* There is no way to tell if a drop capital represents a whole word or just the first letter(s) of a word (Example for one word: “I”)

Solution:

* New attribute for text regions: “continuedBy” (IDREF)
* If the drop capital is only the beginning of a word, this attribute points to the text region with the rest of the word
* Advantage: This could also be used to link paragraph regions that span two columns
* One step further: We could think about a more sophisticated model for relationships between regions (not just an attribute). That way, we could link all drop capitals to their corresponding paragraph, even if the drop-capital is a whole word (already covered by reading order though).
  + One-to-one or one-to-many relation for:
    - Drop-cap – paragraph () "DropCap"
    - Paragraph – paragraph (across columns) "Continuation" (link or join depending on last word of first paragraph)
    - Caption – image/chart/table "Link"
    - Floating – paragraph
    - Graphic / handwritten annotation - any region
    - Hyphenation (word level)
* New element ‘Relations’ (child of ‘Page’ element)
* New element ‘Relation’ (with attributes ‘source’ (IDREF) and ‘type’ (String) and RegionRefs as child element(s))
  + To cover the problem of whole-word drop caps we would either need two different types of relations or the proposed ‘incomplete’ attribute (see below)
* Add an attribute “fragment” to indicate the region is continued elsewhere (the link would then be made through the reading order and/or new relation type).

**Solution**:

Introduce relation element with two possible types (link, join):

<Relation type="link"><RegionRef regionRef="r200"/><RegionRef regionRef="r201"/></Relation>

Add comment attribute?

Add validator rule to check if drop caps (warning)/captions(info)/? have relation set

Add examples from above to documentation.

# Polygons (#944) ✓

Problem:

* Current way of representing polygons is rather inefficient (one element per point)
* ALTO: Uses x,y x,y x,y

Solution:

* Use a list of points (“Coords” element with attribute “points”)
* We can do some validation using regular expressions (“pattern restrictions”). For example:

([0-9]+ [0-9]+,){2,}([0-9]+ [0-9]+)

Meaning: At least two points ending with a comma and a final point. Each point is x and y with a space as separator. Example: 2 5,8 11,20 32

See: <http://www.w3schools.com/schema/schema_facets.asp>

Final: ([0-9]+,[0-9]+ )+([0-9]+,[0-9]+) (e.g. “1,2 3,4”)

# Confidence Value (#980, #994) ✓

Need:

* Recognition systems often produce confidence values for OCR results. This information is useful in some scenarios and should be preserved.
* ALTO: Has confidence values for primary words and characters, but not for alternative words.

Idea:

* Add (optional) “conf” attribute (float) to TextEquiv. That way we cover all levels of page elements. (Constraint: values 0...1)
* To think about: Should there be confidence values for polygons as well?

Even more:

* We can think about allowing multiple TextEquiv elements per text object (to allow for alternatives)

# Named Reading Order Groups and Layers (#207, #411) ✓

Need:

* Giving groups and layers can be useful (e.g. foreground and background layer)

Solution:

* Straightforward: Add (optional) attribute “name” or **“caption”** to OrderedGroup, UnorderedGroup and Layer

# Text Production Type (#899)

Need:

* Mainly to distinguish between handwritten and printed

**Solution**:

* Add (optional) “**production**” (productionType, productionMethod) attribute to TextRegion, TextLine, Word and Glyph
* Documentation/GUI: Use inheritance like in CSS (text line inherits production type from parent region)
* Proposal for types: **printed, typewritten, handwritten-cursive, handwritten-printscript, medieval-manuscript**, (carved), (engraved)

Explain in documentation - see block letters (not necessarily capital letters)

# Text Style (#331, #958)

Need:

* Information about text style

Idea:

* Add Style element
* Add “style” attribute (that points to a style element) to all text elements
* Styles: bold, italic, underlined, strikethrough, subscript, superscript, letter-spaced, small-caps
* Typeface (font family): blackletter, monospace, roman, ...
  + Stefan: E.g. Latin typeface, gothic typeface, Kurrent, Süterlin, Carolingian Minuskel, or consider a **custom string** for more flexibility
* We could add more things like: underline style, ...
* Problem: Style can only be applied to objects, not text. That means to say a word is BOLD the word has to exist as a word object. But that is OK, we don’t want to create a new mark-up language.
* New element ‘Styles’ (**ALTO** has this as well) (maybe as child of PcGts and not of Page)
* ALTO has ‘TextStyle’ and ‘ParagraphStyle’
  + TextStyle:
    - Id
    - FontFamily (“name of font”)
    - FontType (serif or sans serif)
    - FontWidth (fixed or proportional), FontSize
    - FontColor
    - FontStyle (list attribute with bold, italics, subscript, superscript, smallcaps, underline)
  + ParagraphStyle:
    - Id
    - Align (left, right, center or justify)
    - Left, Right (indent of the paragraph in relation to the column)
    - LineSpace (Line spacing between two lines of the paragraph. Measurement calculated from baseline to baseline.)
    - FirstLine (Indent of the first line of the paragraph if this is different from the other lines. A negative value indicates an indent to the left, a positive value indicates an indent to the right.)
  + Every layout element has an ‘StyleRefs’ attribute (of type IDREFS)

Note: Some text region attributes might become obsolete!

**Solution**:

Every text object has a style element (same level as TextEquiv)

Use inheritance like in CSS

* follow ALTO as far as possible:
  + TextStyle:
    - FontFamily (“name of font”) - custom string due to a lack of standardisation (can be used e.g. also to distinguish between black letter and Antigua fonts)
    - **Serif** (serif or sans serif)
    - Monospace (true false)
    - FontSize (move to style)
    - Text Colour (move to style)
    - Background colour (move to style)
    - Kerning (move to style)
    - Reverse video (move to style)
    - FontStyle (list attribute with bold, italic, underlined, subscript, superscript, strikethrough, small-caps, letter-spaced)
  + TextRegion style attributes:
    - Align (left, right, centre or justify)
    - LineSpace (Line spacing between two lines of the paragraph. Measurement calculated from baseline to baseline.) - keep Leading (do not move to style)
    - FirstLine (Indent of the first line of the paragraph if this is different from the other lines. A negative value indicates an indent to the left, a positive value indicates an indent to the right.) - use Indented as we had it before

# Language at Text Line and Word Level (#230)

Need:

* Mostly for inline citations using a different language than the main text body

Solution:

* Add “language” attribute to TextLine and Word (LanguageSimpleType)
* Inherit language from parent object if not specified (like CSS)
  + Use primaryLanguage

# New Region Types (#239, #240)

Need:

* Chemical formula (proposed for ALTO)
* Musical notation
* Advertisement

Proposals:

* ChemRegion
* MusicRegion
* AdvertRegion (Container for sub-regions)

# New Sub-Types and Languages (#235, #925, #941)

* Text: endnote, other
* Graphic:
  + barcode (maybe also QR-code, though this is a special type of 2D-barcode)
  + decoration
  + (frame) – If we allow nesting of region we could get rid of FrameRegion altogether and use graphic region **of type frame instead**.
* Languages: from latest ISO 639.x specification (e.g. Slovenian) ✓
  + Now 187 languages (made sure it is a superset of the old languages)

# Notes/Comments/Custom Types (#956)

Need:

* In some cases it would be nice to have *custom* value for “type” and similar attributes.

Solution:

* Add “*other”* to all relevant attribute constraints (e.g. TextRegion type, ...)
* Add new optional attribute “**custom**” or “customType”, ”extendedType”, “typeInfo”, “typeDetails” (to be more specific?) of type String
* Recommendation: Add “**comments**” (or “notes”, “info”) attribute to various (all?) elements

Documentation - make it clear that this is intended as a mechanism for making the format more flexible for individual projects and that this may heavily depend on project(dataset)-specific conventions.

Custom added to: Page, all regions, TextLine, Word, Glyph, Relation

Comments added to: All regions, TextLine, Word, Glyph, Relation

# Alternative Images (#452)

Need:

* Save filenames of all available images (B/W, grey, colour, ...)

Solution:

* New elements:
  + - AlternativeImage (0-\*) (attribute filename)
* ~~Should we maybe get rid of the old “imageFilename” attribute and handle all images in the same structure (should not be called AlternativeImages in that case). Then we need a mechanism to distinguish between the ‘master’ image and alternatives~~

# Extended Page Element (#138)

New attributes for “Page” element:

* “**type**” – front-cover, back-cover, title, table-of-contents, index, blank, content, other
* ~~“matter” (or structureSection?) – front-matter, body-matter, back-matter - leave out for now~~
* ~~ALTO has generic ‘PageClass’ attribute (“Any user-defined class like title page.”)~~

# Baseline for text lines (#957) ✓

Need:

* Relevant mostly for handwritten text

Solution:

* Poly-line for TextLine element
  + “Baseline” element with “points” attribute (same format as cords)

# Plain Text Optional (#276) ✓

Stefan: Make PlainText optional but keep Unicode mandatory as this is the preferred element to store typed text (TextEquivType).

# Named Entities (#992)

Need:

* Tagging named entities (such as persons, locations, organisations)

Idea:

* Named entities make most sense on word level
* An entity can consist of multiple words!
* Have a separate structure
  + NamedEntities
    - NamedEntityGroup (“type” attribute with *location, person, organisation, other*, ...)
      * NamedEntity
        + WordRef (“wordRef” attribute of type IDREF, maybe also with “sortIndex”) (possibly multiple)

Comment: Isn’t this going too far? Maybe save this for the next schema iteration?

# Nested Regions (#600)

Need:

* Often there is embedded text in graphics and other regions.

Idea:

* Allow nesting for more region types (at the moment only frame region)
  + Typical parent regions: Graphic, image, line drawing, chart, table, maths, frame
  + Typical child regions: text, graphic, image, line drawing, separator, ... (basically all types)
* Maybe we should add a RegionContainer type to simplify the schema (the Page element and all allowed parent elements would inherit from this new type to be able to add child regions)
* We really need to think about how much freedom we want to allow. Maybe more restrictive is better for this iteration (small steps). Also this is more in the tradition of PAGE. Do we for instance want to embed text in tables (as cells) or should this be left to a more sophisticated external format. ...
* Do we want to allow recursive nesting with no limit in depth? Or should we (for now) restrict to one level (parent – child, no children’s children)
* What about Noise regions? Do they need to be nested or is noise rather in a different layer?
* Maybe we should collect examples and see what is most common...

Proposal for allowed nesting:

* Text
  + (Graphic), (Maths), (Chem) ?
* Image / Line Drawing
  + Text, (Graphic)
* Graphic / Frame / Table / Advert
  + Everything except Noise
* Separator / Noise / Unknown
  + –
* Chart / Maths / Chem
  + Text?

**Solution:**

Allow nesting of everything within everything

Explain in docu and implement validator rules (for specific use scenarios)

Explain in change log how to convert frame regions to graphic/frame

# Unicode Element for Non-Text Regions (#394) ✓

Redundant when implementing region nesting (see above).