Cues for tools

Thomas Eckart/Twan Goosen

Reminder

- 4 topics:
  - Localised documentation
  - Documentation on compontents and attributes
  - Extended display information
  - Derived values

- Issue: Status: Uncontroversial
  - Documentation has no language property & maxOccurs=1
- Solution:
  - Documentation in new elements instead of attributes (maxOccurs=unbounded, @xml:lang)
  - Transformation to <xs:annotation> in CMDI profiles
- Impact:
  - ComponentRegistry
  - Editors (Arbil, Proforma)

Documentation on compontents and attributes

- Issue: Status: Uncontroversial
  - Right now only documentation for elements
  - Missing for components and attributes
- Proposal:
  - Component specification:

```
<CMD_Component CardinalityMax="unbounded" CardinalityMin="0" name="name">
    <Documentation xml:lang="en">Name of the described document</Documentation>
...
</CMD>
```

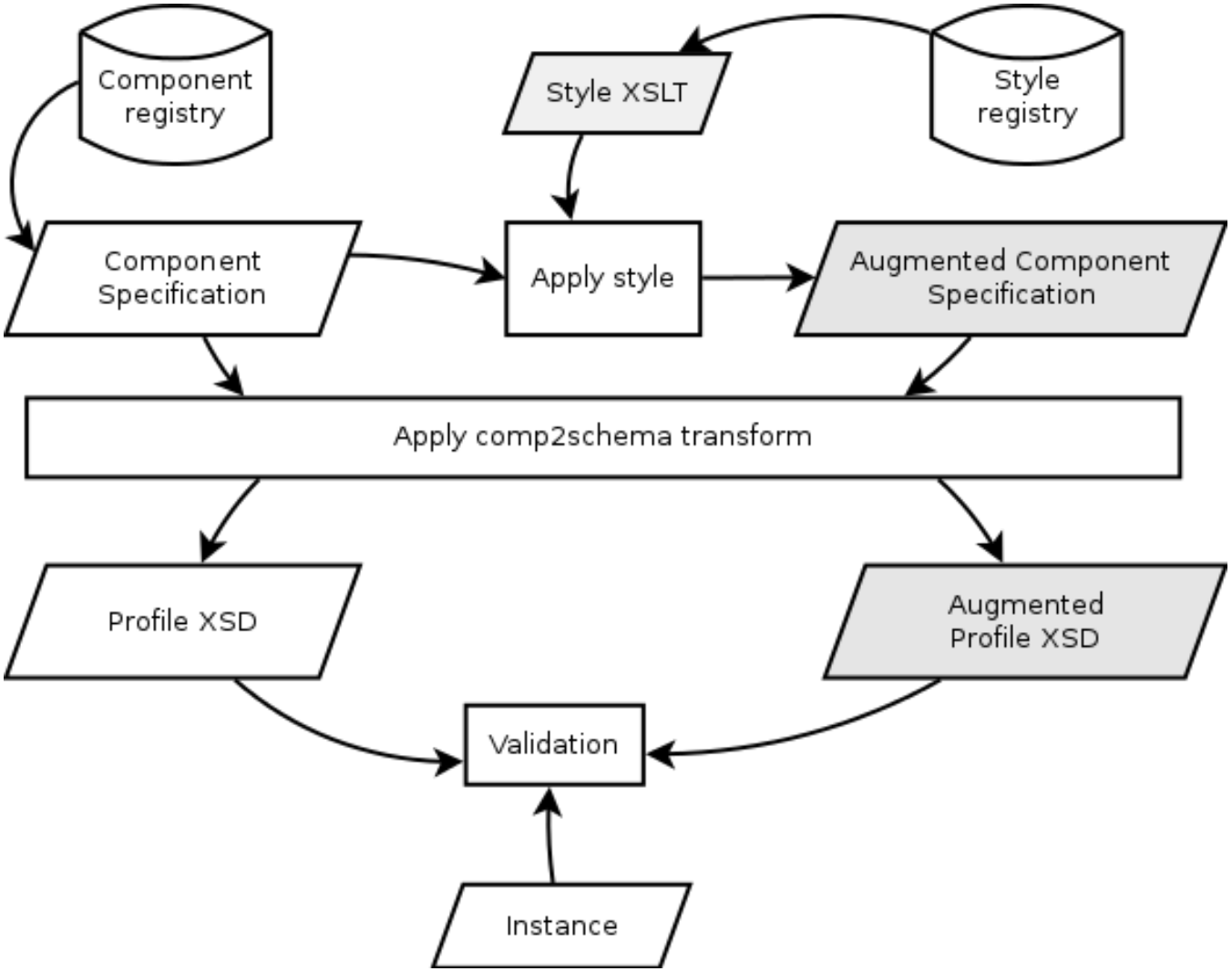
  - XSD:

```
<xs:element name="name" minOccurs="1" maxOccurs="1">
    <xs:annotation>
        <xs:documentation xml:lang="en">Name of the described document</xs:documentation>
    </xs:annotation>
```

Extended display information

- Idea: Status: Open questions
  - Add visual hints to components
    - CSS-like functionality
    - Information about Grouping/Merging of components
    - Replacement for DisplayPriority
    - *Importance* of information for editor (mandatory, recommended etc.)
  - Proposal:
    - Separate namespace(e.g. http://www.clarin.eu/cmdi/cues/display/1.0) with attributes for display cues
    - Component specification allows all attributes on components, elements, attributes
    - Additional XSLT stylesheets create on-the-fly extended component specification (optional attribute for schema URL at ComponentRegistry)

Extended display information



- Needs some *style registry* (near or in the ComponentRegistry) + some style editor (could be created at a later stage)
- Impact:
  - ComponentRegistry
  - Editors (Arbil, ProForma)
  - Viewer (VLO)

Extended display information

- General component schema:
  - Extend attribute groups with:

    <xs:anyAttribute namespace= "http://www.clarin.eu/cmdi/cues/display/1.0" processContents="strict"/>
- Component model:
  - <CMD_Element name="Name" />

    →

    <CMD_Element name="Name" display:backgroundcolor="#f00"/>
  - <CMD_Component name="Actor" >

    →

    <CMD_Component name="Actor" display:displayvaluefield="Name,Role">
- Profile schema:

```
<xs:element name="Actor" minOccurs="1" maxOccurs="1" display:displayvaluefield="Name,Role">
 <xs:complexType>
    <xs:sequence>
       <xs:element name="Name"
          display:backgroundcolor="#f00"
          minOccurs="1" maxOccurs="1">
```

Extended display information

- Pro:
  - Only (optional) extension
  - Supports multiple styles
  - Generic styles could be helpful (VLO)
- Contra:
  - XSLT allows arbitrary transformations (not only adding display cues)
  - Cues may not always be applicable (some tolerance in tools required)
- Different name space for different kind of cues (editor, viewer) ?

- Idea: Status: Open questions
  - Some metadata can be generated/derived/inserted automatically
    - AgeOfPerson = floor(DateOfDeath-DateOfBirth)
    - NameOfLanguage = LanguageName(LanguageCode)
    - $CreationDate
  - Allow derivation of values
  - 2 proposals for CMDI 1.2

- Proposal 1:
  - Extend general component schema with optional attributes *AutoValueProcedure* and *AutoValueParameters*
  - AutoValueProcedure contains URI for a function that is defined externally (functionality, allowed arguments)
  - AutoValueParameters contains arguments as XPath references (if any)
  - No impact on instances

- Proposal 1: Implementation
  - Extend general component schema with two attributes:

```
<xs:attributeGroup name="clarin_element_attributes">
    ...
    <xs:attribute name="AutoValueProcedure" type="xs:anyURI">
        <xs:annotation>
            <xs:documentation>The URI of a procedure that is used to derive
            the content of this element based on external
            information.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="AutoValueParameters" type="xs:string">
```

  - Component model:

```
<CMD_Element name="Duration"
AutoValueProcedure="http://www.clarin.eu/cmdi/autovalue/procedure/datediff
" AutoValueParameter="../StartRangeDate ../EndRangeDate"/>
```

- Impact:
  - ComponentRegistry
  - Editors
- Pro:
  - By using references functionality can be easily extended
  - Procedures can be implemented as services (hides complexity and helps editor tools)
- Con:
  - Needs additional *function registry* (allowed operations & constants + allowed arguments)
  - Referenced nodes may not exist: optional elements & various contexts (some tolerance in editors required)

- Proposal 2:
  - Extend component specification with optional attribute *AutoValue*
  - AutoValue contains function to generate content based on referenced nodes (XPath expressions)
  - Based on fixed set of functions and constants
    - Numeric functions
    - Simple string operations
  - No impact on instances

- Proposal 2: Implementation
  - Extend general component schema by two attributes:

```
<xs:attributeGroup name="clarin_element_attributes">
    …
    <xs:attribute name="AutoValue" type="xs:string">
        <xs:annotation>
            <xs:documentation>A function that is used to derive the
            content of this element based on external
            information</xs:documentation>
        </xs:annotation>
    </xs:attribute>
```

  - Component model:

```
<CMD_Element name="AgeOfPerson"
AutoValue="$CurrentDate-date({../BirthDate})"/>
```

Insertion of derived values

- Impact:
  - ComponentRegistry
  - Editors
- Pro:
  - Simpler solution that proposal 1
  - No new external function registry needed
- Con:
  - May lack expressiveness (missing functions)
  - New functions result in changes in editors
  - Referenced nodes may not exist: optional elements & various contexts (some tolerance in editors required)

- Other questions:
  - Chaining?
  - Probably extra namespace?
  - Syntax?

  - *Additional use cases?*