Einleitung

Absatz über Baumann, Projekt und so.

Dennis Ried

Die Infrastruktur

Setting

Um Kritische Anmerkungen systematisch erfassen zu können, müssen zunächst einige Grundparameter festgelegt werden: Welche Quellen gibt es? Welche Instrumente sind beteiligt? Wird auf Editionen referenziert? Usw.

Für die Erfassung der Kritischen Anmerkungen ist es grundlegend, dass es die zuvor genannten Objekte definiert sind. Die Abhängigkeiten untereinander (bspw. bei den Quellen [Stemma]), ist an dieser Stelle aus technischer Sicht noch nicht relevant; die Abhängigkeiten untereinander haben keinen Einfluss auf das Datenmodell.

Im sogenannten Setting werden die Weichen für die spätere Kodierung gestellt, weshalb dieser Bereich im Folgenden kurz dargestellt werden wird.

- *setting* Basic information, e.g., relates Sources, Instrumentation/Voices, Editions.

Hierbei handelt es sich um einen Wrapper, der sämtliche Defintionen für die vorliegende Datei enthält. Dieses Element dient allein der Definition der Infrastruktur.

Musikalische Struktur

- *mdivs*
- *mdiv*

Das an MEI angelehte Element <mdiv> steht für einen musikalsichen Abschnitt (musical division). Dieses generische Element kann verwendet werden, um Sätze eines Werkes, Lieder oder Stücke einer Sammlung, Nummern einer Oper oder sonstige Strukturen abzubilden. Die Numerierung der <mdiv> Elemente mit dem Atttribut *no* ist obligatorisch und als technische Zählung zu betrachten. Als Werte sind ausschließlich Natürliche Zahlen (inklusive der *0*) erlaubt. Diese Zählung ist für die spätere Zuordnung und Sortierung maßgeblich, weshalb hier bspw. keine römische Zählung erlaubt ist. Zur verdeutlichung worum es sich konkret handelt, kann der Satztitel inkl. einer Zählung hinzugefügt werden. Der Textknoten des Elementes fungiert als Freitext.

Das Element <mdivs> fungiert als Wrapper, der alle <mdiv> Elemente in sich beherbergt.

Instrumentation bzw. Besetzung

- *voices* A Wrapper for voices (voice-elements)
- *voiceGrp*
- *voice*

Die Instrumentation kann sich äußerst komplex gestalten, weshalb es hier nötig sein kann, dass einzelne Stimmen zu Gruppen zusammengefasst werden. Das Element <voices> ist hier wieder ein Wrapper, der die Definition der Instrumentation enthält, egal wie diese organisiert ist.

Das Element <voiceGrp> ist ebenfalls ein Wrapper, der zur Gruppierung verwendet wird. Dies kann auch auf mehreren Ebenen geschehen, wie das folgende Beispiel zeigt.

```
<voiceGrp key="woods">
 <voiceGrp key="flutes">
  <voice key="flute.1">Flöte I</voice>
  <voice key="flute.2">Flöte II</voice>
 </voiceGrp>
</voiceGrp>
```

Der verwendete Wert im Atttribut *key* muss eindeutig sein, da hierüber die Zuordnung der Anmerkungen zu den betroffenen Stimmen hergestellt wird. Es empfielt sich die Werte wie im vorangegangenen Beispiel generisch zu wählen unter Vermeidung von Umlauten und Leerzeichen, da damit technische Probleme wie bspw. eine unpraktische Sortierung in den Fehlermeldungen vermieden wird.

Der Grund, der für eine Gruppierung spricht ist, dass damit später auf eine ganze Stimmgruppe verwiesen werden kann wie z.B. *flutes* oder *woods*, ohne jede Stimme einzeln nennen zu müssen.

Das Datenmodell ist in der Lage eine gewisse Kompexität (siehe voriges Beispiel) abzubilden, die gleichzeitig durch das Schema validiert werden kann.

Assoziierte Werke (related works)

- *relWorks* A Wrapper for sources related to the critical apparatus
- *work* A reference to a work this critical apparatus is made for.

An dieser Stelle erfolgt die Zuordnung der kritischen Anmerkungen zu einem Werk. Dies kann mittels ID-Verweis geschehen (*key*) oder über einen expliziten Dateipfad, der im Attribut *target* angegeben wird. Hier ist zu beachten, welchen Zweck die Zuordnung verfolgen soll bzw. in welchen Kontext die Kritischen Anmekrungen eingebunden werden.

Sollen die erfassten Anmerkungen über die crApp-Applikation visualisiert werden, so ist es es ausreichend, wenn die Referenzierung über *key* erfolgt. Vorausgesetzt natürlich, dass sich die Referenzdatensätze in der selben Datenbank befinden in der die Applikation installiert ist.

Im Falle, dass der Pfad zum Referenzdatensatz an einen anderen Ort zeigen

soll, kann der entsprechende Pfad als Wert von *target* eingegeben werden. Dies bietet sich beispielsweise dann an, wenn lokal Transformationen durchgeführt werden, bei denen eine Ansprache über eine ID zu viel Rechenzeit erfordern würden.

Assoziierte Quellen (related sources)

- *relSources* A Wrapper for sources related to the critical apparatus
- *source*

Die Referenzierung von Referenzdatensätzen zu Quellen erfolgt wie unter Assoziierte Werke beschrieben.

Im Vergleich zu <work> weist das Element <source> zwei Besonderheiten auf: Zum einen wird ein *siglum* definiert, welches nötig ist, um betroffene Quellen in den Einzelanmerkungen zu referenzieren. Weiter kann das Attribut *sortNo* dafür verwendet werden, um eine Sortierreihenfolge für die Siglen festzulegen.

Assoziierte Editionen (related editions)

- *relEditions* A Wrapper for editions related to the critical apparatus
- *edition*

Diese Elemente verhalten sich analog zu den Quellen.

Klassifikationen

- *classifications* A wrapper to organize classifications.
- *classGrp* A Wrapper for grouping classes.
- *class* An Element that defines a class or category to be used for classifying.

Klassifikationen sind das A und O bei der Erfassung strukturierter Daten. Diese sind unabdingbar, wenn Anmerkungen bestimmten Gruppen zugeordnet oder im Nachhinnein Gruppiert oder ausgewertet werden sollen.

```
<classifications>
 <classGrp label="sourceType">
  <class key="ms">Handschrift</class>
  <class key="pr">Druck</class>
 </classGrp>
</classifications>
```

Im angeführten Beispiel wird eine Klassifikationsgruppe gezeigt, die Arten von Quellen definiert; hier: Handschriften (*ms*=manuscripts) und Drucke (*pr*=prints).

Es können beliebig viele <classGrp>s definiert werden, die wiederum beliebig viele <class> Elemente enthalten.

Die Kritische Anmerkung

- *remarks* A wrapper for critical <remark> elements.

- *remark* A critical remark.

Der Wrapper <remarks> beinhaltet die Kritischen Anmerkungen. Das Element <remark> stellt die Einzelanmerkung dar.

```
<remark>
 <class>ms</class>
 <mdiv>1</mdiv>
 <occurances>
  <occurance>
   <position/>
<!--                <position measure="1" count="1"/>-->
<!--<range type="start" measure="1" count="1"/>
                 <range type="stop" measure="2" count="3"/>-->
  </occurance>
 </occurances>
 <voices>
  <voice>flute.1</voice>
  <voice>oboe.1</voice>
 </voices>
 <annots>
  <annot>Dies ist eine Anmekrung von großer Wichtigkeit.</annot>
 </annots>
 <sources>
  <source>SV</source>
 </sources>
 <editions>
  <edition>BauA</edition>
 </editions>
</remark>
```

- *class* An Element that defines a class or category to be used for classifying.

Das Element <class> dient der Klassifizierung. Die Werte, die als Text eingegeben werden können, sind in <setting> in der Klassifizierung definiert. Wird ein Wert eingegeben, der zuvor nicht definiert wurde wirft die Schematron-Validierung einen Fehler aus.

- *mdiv*

Über <mdiv> erfolgt eine Zuordnung zu dem musikalischen Abschnitt, in dem die Anmerkung vorkommt. Für den eher seltenen Fall, dass eine Anmerkung sich auf mehrere Sätze bezieht (bspw. bei einer Parallelstelle, die in einem anderen Satz in Erscheinung tritt), können mehrere <mdiv> Elemente verwendet werden. In diesem Falle wird von der Schematron-Validierung eine Information angezeigt, dass dieses Verfahren möglich jedoch eher unüblich ist. Ein Validierungsfehler wird dadurch nicht verursacht. Im Gegensatz dazu kann auf <mdiv> verzichtet werden, wenn es sich um eine Generalanmerkung handelt, die für das gesamte Werk oder eine einzelne Quelle gilt.

- *occurances*
- *occurance*

- *position*

Das Element <occurances> ist ein Wrapper und verzeichnet diejenigen Stellen, an denen besagte Anmerkung von Relevanz ist. Da es durchaus sein kann, dass eine Anmerkung des öfteren von Relevanz ist (bspw. in den Takten 2,4,5 und 8 enden die Bögen bereits eine Achtel früher), ist es auch möglich das Element <occurance> mehrfach zu verwenden. Ziel dieser Kodierung ist es, dass ein und diesselbe Anmerkung, die für mehrere Stellen gilt nicht mehrfach geschrieben wird, sondern die einzelnen Stellen dem Befund zugeordnet werden.

Die Stelle kann auf zwei verschiedene Arten identidiziert werden: als Stelle (<position>) oder als Bereich. Im ersten Fall handelt es sich um ein Vorkommen dessen Position sich durch eine Taktzahl und ggf. durch eine Zählzeit angeben lässt. Eine Schema-basierte Validierung von Taktzahlen und Zählzeiten ist derzeit nicht verfügbar, da eine Erfassung der Taktstruktur sowie des möglicherweise wechselnden Metrums zur Zeit nicht vorgesehen ist. Es ist allerdings denkbar, dass diese Struktur später über die referenzierten Quellen eingebunden werden könnte.

```
<occurance>
 <position count="3" measure="2"/>
</occurance>
```

Betrifft die Anmerkung nicht eine einzelne Position, sondern einen Bereich (bspw. Takt 1 Zählzeit 1 bis Takt 3 Zählzeit 4 ohne Dynamikangaben) kann dieser ebenfalls über das Element <position> unter Verwendung des Attributes *type* (Werte *start*/*stop*) erfasst werden.

```
<occurance>
 <position count="1" measure="1"
  type="start"/>
 <position count="4" measure="3" type="stop"/>
</occurance>
```

Specifications of crApp

Elements

<annot>

<annot>		
Module	remark	
Attributes	resp	An attribute that gives a key of a person that is responsible for the content. **Status** Optional **Datatype**text
Contained by	remark: annots	

May contain Character data only

Content model

```
<content>
 <textNode/>
</content>
```

Schema Declaration

```
element annot { attribute resp { text }
?, text }
```

<annots>

<annots> A Wrapper for annotations (annot-elements)

Module remark

Contained by remark: remark

May contain remark: annot

Content model

```
<content>
 <elementRef key="annot"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

```
element annots { annot+ }
```

<apparatus>

<apparatus> The root element for the critical apparatus.

Module setting

Attributes att.id (*@xml:id*)

Contained by —

May contain remark: remarks
setting: setting

Content model

```
<content>
 <elementRef key="setting" maxOccur
s="1"
  minOccurs="1"/>
 <elementRef key="remarks"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

```
element apparatus { att.id.attributes, s
etting, remarks+ }
```

<class>

<class> An Element that defines a class or category to be used for classifying.

Module	setting
Attributes	att.key (*@key*)
Contained by	remark: remark setting: classGrp
May contain	Character data only
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" val ue="class"/> <sch:param name="attrStr" value="ke y"/> </sch:pattern>
Content model	<content> <textNode/> </content>
Schema Declaration	element class { att.key.attributes, text }

<classGrp>

<classGrp> A Wrapper for grouping classes.

Module	setting
Attributes	att.label (*@label*)
Contained by	setting: classifications
May contain	setting: class
Content model	<content> <elementRef key="class" maxOccurs="unbounded" minOccurs ="1"/> </content>
Schema Declaration	element classGrp { att.label.attributes, class+ }

<classifications>

<classifications> A wrapper to organize classifications.

Module	setting
Contained by	setting: setting
May contain	setting: classGrp
Content model	<content>

```
 <elementRef key="classGrp"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

```
element classifications { classGrp+ }
```

<edition>

<edition>

Module	setting
Attributes	att.key (*@key*) att.numbering (*@no*, *@sortNo*) att.siglum (*@siglum*) att.target (*@target*)
Contained by	remark: editions setting: relEditions
May contain	Character data only
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" value="edition"/> <sch:param name="attrStr" value="siglum"/> </sch:pattern>

Content model

```
<content>
 <textNode/>
</content>
```

Schema Declaration

```
element edition
{
   att.key.attributes,
   att.numbering.attributes,
   att.siglum.attributes,
   att.target.attributes,
   text
}
```

<editions>

<editions> A Wrapper for editions (edition-elements)

Module	remark
Contained by	remark: remark
May contain	setting: edition

Content model

```
<content>
 <elementRef key="edition"
  maxOccurs="unbounded" minOccurs
="1"/>
```

</content>

Schema Declaration

element editions { edition+ }

<editor>

<editor> An element that defines an editor responsible for the content.

Module setting
Attributes att.key (*@key*)
Contained by setting: editors
May contain Character data only
Content model

<content>
 <textNode/>
</content>

Schema Declaration

element editor { att.key.attributes, text }

<editors>

<editors> A wrapper for <editor> elements.

Module setting
Contained by setting: setting
May contain setting: editor
Content model

<content>
 <elementRef key="editor"
  maxOccurs="unbounded" minOccurs="1"/>
</content>

Schema Declaration

element editors { editor+ }

<layer>

<layer>
Module setting
Contained by setting: layers
May contain Character data only
Schematron <sch:pattern is-a="checkValues">
<sch:param name="elementName" value="layer"/>
<sch:param name="attrStr" value="key"/> </sch:pattern>
Content model

<content>
 <textNode/>
</content>

Schema Declaration

element layer { text }

<layers>

<layers> A Wrapper for layers (layer-elements)

Module	setting
Contained by	—
May contain	setting: layer
Content model	<content> <elementRef key="layer" maxOccurs="unbounded" minOccurs="1"/> </content>
Schema Declaration	element layers { layer+ }

<mdiv>

<mdiv>	
Module	setting
Attributes	att.numbering (*@no*, *@sortNo*)
Contained by	remark: remark setting: mdivs
May contain	Character data only
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" value="mdiv"/> <sch:param name="attrStr" value="no"/> </sch:pattern>
Schematron	<sch:pattern> <sch:rule context="crapp:remark"> <sch:assert role="info" test="count(crapp:mdiv) = 1">Please note there is more then one occurance of mdiv.</sch:assert> </sch:rule> </sch:pattern>
Content model	<content> <textNode/> </content>
Schema Declaration	

element mdiv { att.numbering.attributes, text }

<mdivs>

<mdivs>

Module setting

Contained by setting: setting

May contain setting: mdiv

Content model

```
<content>
 <elementRef key="mdiv"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

element mdivs { mdiv+ }

<occurance>

<occurance>

Module remark

Contained by remark: occurances

May contain remark: position

Content model

```
<content>
 <alternate maxOccurs="1" minOccurs
="1">
  <elementRef key="position" maxOcc
urs="2"
   minOccurs="1"/>
 </alternate>
</content>
```

Schema Declaration

element occurance { position, position ? }

<occurances>

<occurances>

Module remark

Contained by remark: remark

May contain remark: occurance

Content model

```
<content>
 <elementRef key="occurance"
  maxOccurs="unbounded" minOccurs
```

="1"/>
</content>

Schema Declaration

element occurances { occurance+ }

<position>

<position>

Module remark

Attributes att.timing (*@measure*, *@count*)

type **Status** Optional

Legal values are:

start (starting point) The location where the range of the occurance starts.

stop (stopping point) The location where the range of the occurance stops.

Contained by remark: occurance

May contain Empty element

Schematron

```
<sch:pattern>
<sch:rule context="crapp:position">
<sch:report test="if(count(parent::nod
e()/crapp:position) = 1)
then(self::node()
[@type])else(false())">If there is only
one position @type is not
allowed.</sch:report>
<sch:assert test="if(count(parent::nod
e()/crapp:position) = 2)
then(parent::node()/crapp:position[1]/
@type='start')else(true())">The value
of @type in the first position must be
'start'</sch:assert>
<sch:assert test="if(count(parent::nod
e()/crapp:position) = 2)
then(parent::node()/crapp:position[2]/
@type='stop')else(true())">The value
of @type in the second position must
be 'stop'</sch:assert> </sch:rule>
</sch:pattern>
```

<relEditions>

<relEditions> A Wrapper for editions related to the critical apparatus

Module setting

Contained by setting: setting

May contain setting: edition

Content model

```
<content>
 <elementRef key="edition"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

```
element relEditions { edition+ }
```

<relSources>

<relSources> A Wrapper for sources related to the critical apparatus

Module setting

Contained by setting: setting

May contain setting: source

Content model

```
<content>
 <elementRef key="source"
  maxOccurs="unbounded" minOccurs
="1"/>
```

</content>

Schema Declaration

element relSources { source+ }

<relWorks>

<relWorks> A Wrapper for sources related to the critical apparatus

Module setting

Contained by setting: setting

May contain setting: work

Content model

```
<content>
 <elementRef key="work"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

element relWorks { work+ }

<remark>

<remark> A critical remark.

Module remark

Attributes att.id (*@xml:id*)

type **Status** Optional
Legal values are:
editorial Intervention (ed.)
reading Reading
annotation Annotation (ed.)

Contained by remark: remarks

May contain remark: annots editions occurances sources
setting: class mdiv voices

Example

```
<remark>
 <class>dynamic</class>
 <mdiv>1</mdiv>
 <occurances>
```

```
  <occurance>
   <position/>
   <position count="1" measure="1"
    type="start"/>
   <position count="3" measure="2"
    type="stop"/>
  </occurance>
 </occurances>
 <voices>
  <voice>flute.1</voice>
  <voice>oboe.1</voice>
 </voices>
 <annots>
  <annot>No dynamics.</annot>
 </annots>
 <sources>
  <source>SV</source>
 </sources>
 <editions>
  <edition>BauA</edition>
 </editions>
</remark>
```

Content model

```
<content>
 <elementRef key="class"
  maxOccurs="unbounded" minOccurs
="0"/>
 <elementRef key="mdiv"
  maxOccurs="unbounded" minOccurs
="0"/>
 <elementRef key="occurances" maxO
ccurs="1"
  minOccurs="0"/>
 <elementRef key="voices" maxOccurs
="1"/>
 <elementRef key="annots" minOccurs
="1"/>
 <elementRef key="sources" maxOccu
rs="1"/>
 <elementRef key="editions" maxOccu
rs="1"/>
</content>
```

Schema Declaration

```
element remark
{
   att.id.attributes,
   attribute type { "editorial" | "reading
```

```
" | "annotation" }?,
  class*,
  mdiv*,
  occurances?,
  voices,
  annots,
  sources,
  editions
}
```

<remarks>

<remarks> A wrapper for critical <remark> elements.

Module remark

Contained by setting: apparatus

May contain remark: remark

Content model

```
<content>
 <elementRef key="remark"
  maxOccurs="unbounded" minOccurs
="1"/>
</content>
```

Schema Declaration

```
element remarks { remark+ }
```

<setting>

<setting> Basic information, e.g., relates Sources, Instrumentation/Voices, Editions.

Module setting

Contained by setting: apparatus

May contain setting: classifications editors mdivs relEditions relSources relWorks voices

Content model

```
<content>
 <elementRef key="editors"/>
 <elementRef key="mdivs"/>
 <elementRef key="voices"/>
 <elementRef key="relWorks"/>
 <elementRef key="relSources"/>
 <elementRef key="relEditions"/>
 <elementRef key="classifications"/>
</content>
```

Schema Declaration

```
element setting
{
  editors,
```

```
  mdivs,
  voices,
  relWorks,
  relSources,
  relEditions,
  classifications
}
```

<source>

<source>

Module	setting
Attributes	att.key (*@key*) att.numbering (*@no*, *@sortNo*) att.siglum (*@siglum*) att.target (*@target*)
Contained by	remark: sources setting: relSources
May contain	Character data only
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" value="source"/> <sch:param name="attrStr" value="siglum"/> </sch:pattern>
Content model	<content> <textNode/> </content>
Schema Declaration	element source { att.key.attributes, att.numbering.attributes, att.siglum.attributes, att.target.attributes, text }

<sources>

<sources> A Wrapper for sources (source-elements)

Module	remark
Contained by	remark: remark
May contain	setting: source
Content model	<content> <elementRef key="source" maxOccurs="unbounded" minOccurs="1"/>

</content>

Schema Declaration

element sources { source+ }

<voice>

<voice>

Module	setting
Attributes	att.key (*@key*)
Contained by	setting: voiceGrp voices
May contain	Character data only
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" val ue="voice"/> <sch:param name="attrStr" value="ke y"/> </sch:pattern>

Content model

<content>
 <textNode/>
</content>

Schema Declaration

element voice { att.key.attributes, text }

<voiceGrp>

<voiceGrp>

Module	setting
Attributes	att.key (*@key*)
Contained by	setting: voiceGrp voices
May contain	setting: voice voiceGrp character data
Schematron	<sch:pattern is-a="checkValues"> <sch:param name="elementName" value="voiceGrp"/> <sch:param name="attrStr" value="ke y"/> </sch:pattern>

Content model

<content>
 <alternate maxOccurs="1" minOccurs ="1">
  <sequence maxOccurs="unbounded"
   minOccurs="1">
   <elementRef key="voiceGrp"
    maxOccurs="unbounded" minOccur s="0"/>
   <elementRef key="voice"

```
    maxOccurs="unbounded" minOccur
s="0"/>
  </sequence>
  <textNode/>
 </alternate>
</content>
```

Schema Declaration

```
element voiceGrp { att.key.attributes, (
( voiceGrp*, voice* )+ | text ) }
```

<voices>

<voices> A Wrapper for voices (voice-elements)

Module setting

Contained by remark: remark
setting: setting

May contain setting: voice voiceGrp

Content model

```
<content>
 <sequence maxOccurs="unbounded"
  minOccurs="1">
  <elementRef key="voiceGrp"
   maxOccurs="unbounded" minOccurs
="0"/>
  <elementRef key="voice"
   maxOccurs="unbounded" minOccurs
="0"/>
 </sequence>
</content>
```

Schema Declaration

```
element voices { ( voiceGrp*, voice* )+
}
```

<work>

<work> A reference to a work this critical apparatus is made for.

Module setting

Attributes att.target (*@target*) att.key (*@key*)

Contained by setting: relWorks

May contain Character data only

Content model

```
<content>
 <textNode/>
</content>
```

Schema Declaration

```
element work { att.target.attributes, at
```

t.key.attributes, text }

Attribute classes

att.id

att.id A class for id like attributes.

Module	derived-module-crApp	
Members	*apparatus remark*	
Attributes	xml:id	ID of the file. Must be unique in the context of this framework. **Status** Optional **Datatype** ID

att.key

att.key A class for key like attributes.

Module	derived-module-crApp	
Members	*class edition editor source voice voiceGrp work*	
Attributes	key	An attribute that gives a Key as a pointer to a resource. **Status** Optional **Datatype** text

att.label

att.label A class for labelling attributes.

Module	derived-module-crApp	
Members	*classGrp*	
Attributes	label	An attribute for labelling an element in any way. **Status** Optional **Datatype** text

att.numbering

att.numbering A class for attributes that are used for numbering or counting.

Module	derived-module-crApp	
Members	*edition mdiv source*	
Attributes	no	An attribute for a numbering. The value must be an integer.

	sortNo	An attribute for ordering purpose. The value must be a positive integer. **Status** Optional **Datatype** integer

att.siglum

att.siglum An attribute class for classifying using an abbreviation (e.g., a siglum).

Module	derived-module-crApp	
Members	*edition source*	
Attributes	siglum	An attribute containing a text value to classify a source using an abbreviation (siglum). **Status** Optional **Datatype** text

att.target

att.target A class for attributes to defining targets.

Module	derived-module-crApp	
Members	*edition source work*	
Attributes	target	An attribute for defining an explicit target to an other destination. The value must be an URI. **Status** Optional **Datatype** URI

att.timing

att.timing An attribute class for timing concerns.

Module	derived-module-crApp	
Members	*position*	
Attributes	measure	An attribute for defining a measure as a (part of a) time stamp. The value must be a positive integer. **Status** Optional **Datatype** integer

	count	An attribute for defining a count (within a measure) as detailed part of a time stamp. The value must be a positive integer. **Status** Optional **Datatype** integer

Constraints

Schematron	This constraint checks if the called voices inside a remark are redundant, e.g., if the parent or ancestor group is also called. <sch:rule context="crapp:remark//crapp:voice"> <sch:let name="voiceVal" value="./text()"/> <sch:let name="voiceGrpVals" value="string-join(ancestor::crapp:remark//crapp:voiceGrp/text(),' ')"/> <sch:report role="warning" test=".[ancestor::crapp:apparatus/crapp:setting//crapp:voice[@key = $voiceVal]/ancestor::crapp:voiceGrp[@key][contains($voiceGrpVals, @key)]]"> <sch:value-of select="$voiceVal"/> is part of <sch:value-of select="tokenize($voiceGrpVals, ' ')[last()]"/>. A second call might be redundant.</sch:report> </sch:rule>
Schematron	This constraint checks if the called values are defined somewhere in the <setting> element. <sch:schema queryBinding="xslt2"> <sch:ns prefix="crapp" uri="http://baumann-digital.de/ns/criticalApparatus"/> <xsl:function as="xs:boolean" name="functx:is-value-in-sequence"> <xsl:param as="xs:anyAtomicType?" name="value"/> <xsl:param as="xs:anyAtomicType*" name="seq"/> <xsl:sequence select="$value =

```
$seq"/> </xsl:function>
<xsl:function name="crapp:checkValu
es">
<xsl:param as="node()" name="elem"/
>
<xsl:param as="xs:string" name="att
Name"/>
<xsl:variable name="elemName"
select="local-name($elem)"/>
<xsl:variable name="elemText"
select="$elem/text()"/>
<xsl:variable name="setVals"
select="$elem/ancestor::crapp:apparat
us//crapp:setting//node()[local-
name()=$elemName]"/>  <xsl:value-
of select="functx:is-value-in-
sequence($elemText,
if($setVals/attribute::node()[local-
name()=$attName])then($setVals/
attribute::node()[local-
name()=$attName])
else($setVals/text()))"/>
</xsl:function>
<xsl:function name="crapp:valuesAllo
wed">
<xsl:param as="node()" name="elem"/
>
<xsl:param as="xs:string" name="att
Name"/>
<xsl:variable name="elemName"
select="local-name($elem)"/>
<xsl:variable name="elemText"
select="$elem/text()"/>
<xsl:variable name="setVals"
select="$elem/ancestor::crapp:apparat
us//crapp:setting//node()[local-
name()=$elemName]"/>  <xsl:for-
each select="$setVals">   <xsl:value-
of select="if(attribute::node()[local-
name()=$attName])
then(attribute::node()[local-
name()=$attName]/string())
else(text())"/>  </xsl:for-each>
</xsl:function>
<xsl:function name="crapp:valuesAllo
wedJoined">
<xsl:param as="node()" name="elem"/
>
```

<xsl:param as="xs:string" name="att
Name"/> <xsl:value-of select="string-
join(crapp:valuesAllowed($elem,
$attName), ', ')"/> </xsl:function>
<sch:pattern abstract="true"
id="checkValues">
<sch:rule context="crapp:remark//nod
e()[local-name()='$elementName']">
<sch:assert role="fatal"
test="exists(ancestor::crapp:apparatus
/crapp:setting//node()[local-
name()='$elementName'])">No setting
for <sch:value-of select="local-
name(.)"/> defined. Please define the
setting before use.</sch:assert>
<sch:report role="error"
test="exists(ancestor::crapp:apparatus
/crapp:setting//node()[local-
name()='$elementName']) and
crapp:checkValues(., '$attrStr') =
false()">The value <sch:value-
of select="."/> is not allowed here.
Please use one of these: <sch:value-
of select="crapp:valuesAllowedJoined(.
, '$attrStr')"/>.</sch:report>
</sch:rule> </sch:pattern>
</sch:schema>