

Part III:

Description Processing on the XML level

XSLT

Description Transformations

Introduction to XSLT

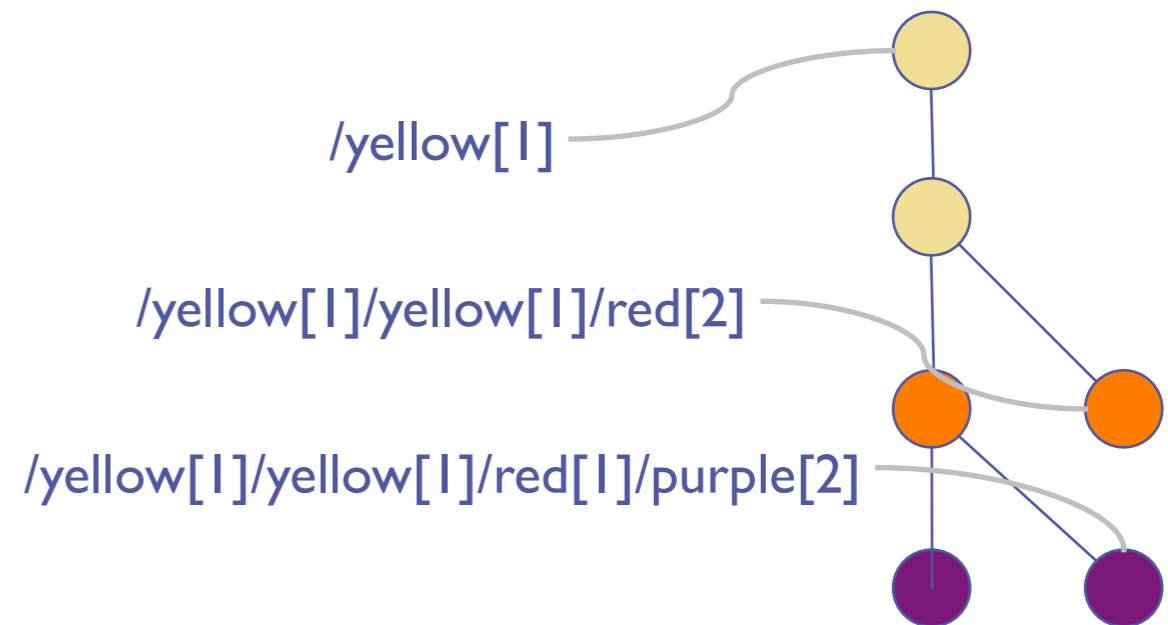
- ◆ Extensible Stylesheet Language:
Transformations
 - ◆ Had its origins in stylesheets, to accompany what was eventually XSL:FO
 - ◆ But fairly early on it was discovered to have a life of its own
- ◆ Builds heavily on the XPath (and XLink) standards

XSLT Basics

- ◆ Syntax is XML
- ◆ Build a series of templates that match elements
- ◆ The template that is applied at any one point is the most specific
- ◆ Essentially
 - ◆ a query match (XPath) followed by
 - ◆ an action/output

XPath Basics

- ◆ XPath walks a tree
- ◆ A path is much like a directory structure
- ◆ Nodes are enumerated amongst their siblings



Simple query: Grep for XML

```
<?xml version='1.0' encoding='iso-8859-1'?>
```

```
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform' xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:xml="http://www.w3.org/XML/1998/namespace" >
```

```
<xsl:output method='text' version='1.0' encoding='iso-8859-1' indent='no'/>
```

```
<xsl:strip-space elements="*" />
```

```
<xsl:template match="//mpeg7:AudioSegment">
```

```
<xsl:text>
```

```
Audio Segment: </xsl:text>
```

```
<xsl:value-of select="mpeg7:TextAnnotation"/><xsl:text>
```

```
Start: </xsl:text>
```

```
<xsl:value-of select="mpeg7:MediaTime/mpeg7:MediaTimePoint"/>
```

```
</xsl:template>
```

```
<xsl:template match="text()"/>
```

```
</xsl:stylesheet>
```

Simple query: Grep for XML

```
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 .\Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="AudioType">
      <Audio>
        <MediaTime>
          <MediaTimePoint>T00:00:00</MediaTimePoint>
          <MediaDuration>PT1M30S</MediaDuration>
        </MediaTime>
        <TemporalDecomposition gap="false" overlap="false">
          <AudioSegment>
            <TextAnnotation>
              <FreeTextAnnotation> Chase </FreeTextAnnotation>
            </TextAnnotation>
            <MediaTime>
              <MediaTimePoint>T00:00:00</MediaTimePoint>
              <MediaDuration>PT0M15S</MediaDuration>
            </MediaTime>
          </AudioSegment>
          <AudioSegment>
            <TextAnnotation>
              <FreeTextAnnotation> Capture </FreeTextAnnotation>
            </TextAnnotation>
            <MediaTime>
              <MediaTimePoint>T00:00:15</MediaTimePoint>
              <MediaDuration>PT1M15S</MediaDuration>
            </MediaTime>
          </AudioSegment>
        </TemporalDecomposition>
      </Audio>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

Result

Audio Segment: Chase
Start: T00:00:00
Audio Segment: Capture
Start: T00:00:15

Parsing/transforming Schema for visualisation

- ◆ First transform for getting down to the hierarchy to a simple XML tree

- ◆ Look for inheritance keywords

- ◆ Each unique element triggers search for children inheriting from that element

```
<xsl:template  
  match="//complexType[@name  
    and not(*/*/@base or */*/*/@base)]  
  | //simpleType[@name  
    and not(*/*/@base or */*/@base)  
    or */*/*/@base]">
```

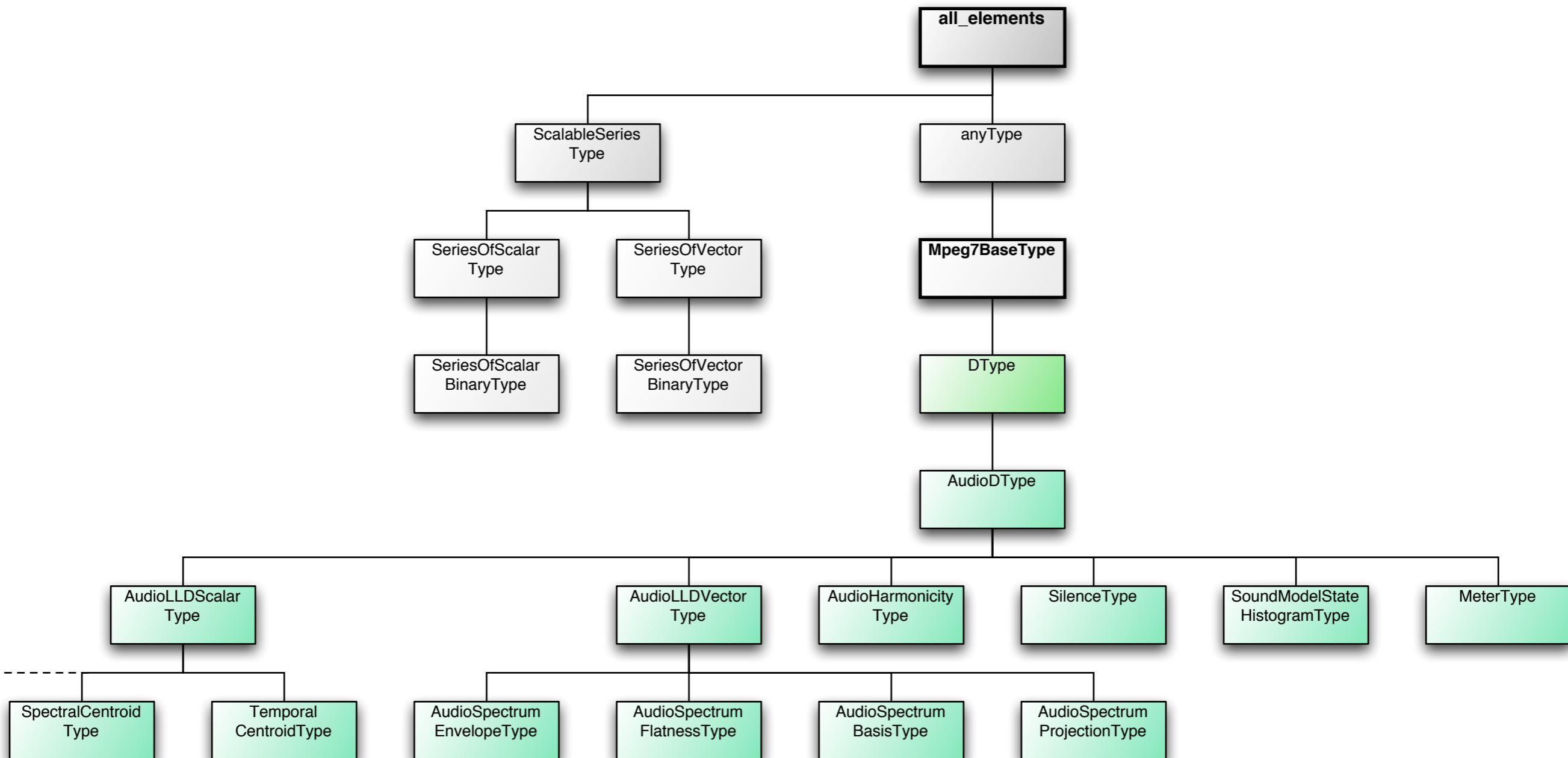
```
...  
</xsl:template>
```

Viewing the result

- ◆ Transform tree again
- ◆ result
- ◆ transform generic tree again into a suitable import format for visualisation

```
<?xml version="1.0" encoding="UTF-8"?>
<all_elements>
  <Mpeg7BaseType abstract="true" inheritance="restriction" type="complexType">
    <DType abstract="true" inheritance="extension" type="complexType/complexType">
      <AudioDType abstract="true" inheritance="extension" type="complexType/complexType">
        <AudioLLDScalarType abstract="true" inheritance="extension" type="complexType/complexType">
          <AudioWaveformType abstract="" inheritance="extension" type="complexType/complexType">
            <AudioPowerType abstract="" inheritance="extension" type="complexType/complexType">
              <AudioSpectrumCentroidType abstract="" inheritance="extension" type="complexType/complexType">
                <AudioSpectrumSpreadType abstract="" inheritance="extension" type="complexType/complexType">
                  <AudioFundamentalFrequencyType abstract="" inheritance="extension" type="complexType/complexType">
                    <LogAttackTimeType abstract="" inheritance="extension" type="complexType/complexType">
                      <HarmonicSpectralCentroidType abstract="" inheritance="extension" type="complexType/complexType">
                        <HarmonicSpectralDeviationType abstract="" inheritance="extension" type="complexType/complexType">
                          <HarmonicSpectralSpreadType abstract="" inheritance="extension" type="complexType/complexType">
                            <HarmonicSpectralVariationType abstract="" inheritance="extension" type="complexType/complexType">
                              <SpectralCentroidType abstract="" inheritance="extension" type="complexType/complexType">
                                <TemporalCentroidType abstract="" inheritance="extension" type="complexType/complexType">
                                  </AudioLLDScalarType>
                                <AudioLLDVectorType abstract="true" inheritance="extension" type="complexType/complexType">
                                  <AudioSpectrumEnvelopeType abstract="" inheritance="extension" type="complexType/complexType">
                                    <AudioSpectrumFlatnessType abstract="" inheritance="extension" type="complexType/complexType">
                                      <AudioSpectrumBasisType abstract="" inheritance="extension" type="complexType/complexType">
                                        <AudioSpectrumProjectionType abstract="" inheritance="extension" type="complexType/complexType">
                                          </AudioLLDVectorType>
                                        <AudioHarmonicityType abstract="" inheritance="extension" type="complexType/complexType">
                                          <SilenceType abstract="" inheritance="extension" type="complexType/complexType">
                                            <SoundModelStateHistogramType abstract="" inheritance="extension" type="complexType/complexType">
                                              <MeterType abstract="" inheritance="extension" type="complexType/complexType">
                                                </AudioDType>
                                              </DType>
                                            </Mpeg7BaseType>
                                          </all_elements>
                                        </all_elements>
                                      </all_elements>
                                    </all_elements>
                                  </all_elements>
                                </all_elements>
                              </all_elements>
                            </all_elements>
                          </all_elements>
                        </all_elements>
                      </all_elements>
                    </all_elements>
                  </all_elements>
                </all_elements>
              </all_elements>
            </all_elements>
          </all_elements>
        </all_elements>
      </all_elements>
    </all_elements>
  </all_elements>
</all_elements>
```

Audio Hierarchy Visualisation



Description transformations are useful...

- ◆ as a utility for converting general ontologies to MPEG-7 Classification Schemes
- ◆ Pruning descriptions
 - ◆ Like making a description less deeply nested
- ◆ Simple queries
- ◆ For transforming the underlying data

Best practice with descriptions

Description Structure • Balance • Symbolic vs Numeric

Many possibilities in Description Structure

- ◆ Heavily partitioned
 - ◆ Hierarchical segments
 - ◆ Potentially along different axes
- ◆ Flat
 - ◆ More commonly seen with Numeric descriptions
 - ◆ Scalable Series

A Recommendation

- ◆ What to do with a combination?
 - ◆ Remember that Scalable Series (any descriptor) can be attached at any level
 - ◆ not just leaf, so one can attach to a segment that makes for a "sensible" array
 - ◆ Can use natural boundaries or "clocked" boundaries

Keep it straightforward for others' use

- ◆ You never know what sort of parsers or assumptions go on at the other end
- ◆ Simpler to parse, simpler to process
- ◆ As with HTML, you never know when you'll need to hand-edit things
- ◆ The art of description building is the art of communication
- ◆ It will eventually come down to industrial conventions

Keep it ideal for your application

- ◆ On the other hand...
- ◆ You can know precisely the parser/processor that you hold
- ◆ No sense in restricting the content structure when there's a limited lifetime outside your processor

In case there's no need for interchange...

- ◆ give a thought as to whether you need to restrict yourself to *MPEG-7* structures at all
- ◆ But you may regret that if ever you do have to export

Balance: Expressivity

- ◆ The structures within *MPEG-7* allow arbitrary precision
- ◆ Highly nested structures
- ◆ Links and descriptors for every eventuality

Balance: Terseness

- ◆ Efficient descriptions that do exactly as prescribed
- ◆ Well-suited to a single purpose with few distractions
- ◆ Fulfils clarity recommendations

Balance in descriptions

- ◆ It again comes down to your description needs
- ◆ Consider partitioning descriptions per use
 - ◆ Main application in one tree
 - ◆ Supplemental information in another

Balancing symbolic with numeric

- ◆ Keep in mind: You get out what you put in with symbolic data
- ◆ Labels, Classification schemes
- ◆ Anything where a search involves an exact match
- ◆ Involves a lot of inference to get anything more than a precise label

Balancing symbolic with numeric (2)

- ◆ You give another user the keys to the kingdom with numeric data
 - ◆ (but they have to build the kingdom, first)
 - ◆ There is a lot more possible in drawing inferences from numeric data than symbolic
 - ◆ But one has to determine what those inferences are
- ◆ Michael's work does a lot to build that bridge

Systems

Dynamic/Progressive Descriptions

Why are systems relevant?

- ◆ It makes descriptions into living documents rather than dead trees
- ◆ Streaming
 - ◆ Progressively building descriptions
- ◆ Real-time
 - ◆ Dynamically revising descriptions

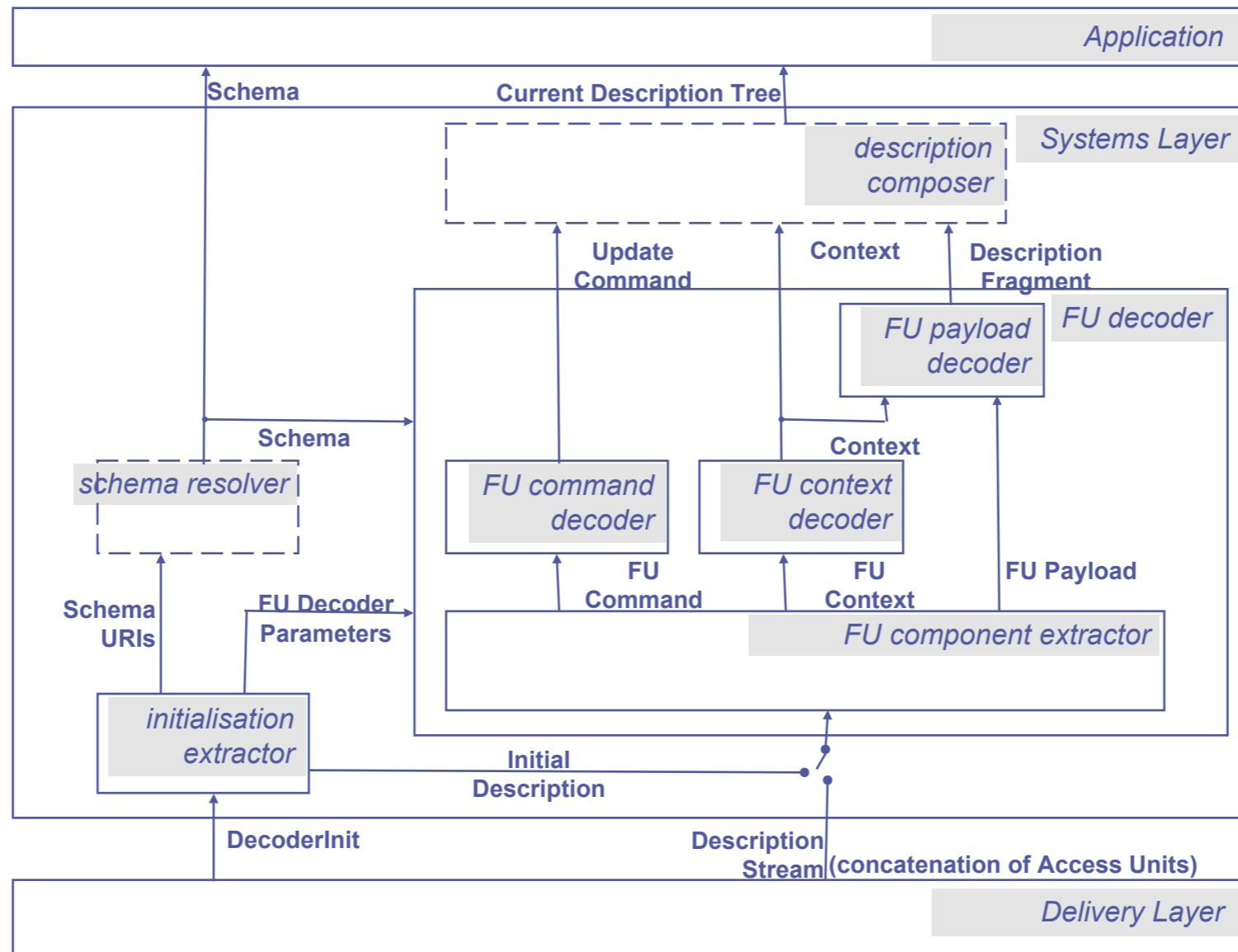
What are the commands?

- ◆ add
 - ◆ + location + payload
- ◆ replace
 - ◆ + location + payload
- ◆ delete
 - ◆ + location
- ◆ reset

Basic Memory model

- ◆ XML Tree
- ◆ Locate places on tree using simplified XPath
 - ◆ (or XPath+Schema knowledge, in binary case)
- ◆ A command is given in the form of
 - ◆ Command @ location with payload

MPEG-7 Systems Architecture



Systems interpreted as XSLT

- ◆ The systems commands can also be reinterpreted as stylesheets
- ◆ two-pass for each update
 - ◆ Generate a stylesheet from the command
 - ◆ Apply the generated stylesheet to the current description tree

