INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG2001/M7819
July 2001, Pattaya

Source: **AIST and University of Brescia**

Title: **Simple Tools for Abstraction and Instantiation**

Author: **Kôiti Hasida and Riccardo Leonardi**

Status: **for discussion**

1 Introduction

A simplified version of tools for abstraction and instantiation [1,2,3] are proposed here. The tools are merely two new attributes applicable to any DS instances.

The merit of abstraction and instantiation is the economy of description and clarification of structural commonality. Abstractions allow you to avoid repeating similar descriptions. You write one template to address the shared structure of descriptions and reuse it when you want similar descriptions. Suppose you want to describe many occurrences of a common pattern of events and states of affairs, such as a type of configuration of soccer players in the field. You will describe the positions of eleven or twenty two people to address this configuration. Once you have described this common pattern, you can capture each occurrence of the pattern by just substituting the eleven or twenty-two people to with particular players, without repeating the descriptions of their positions, among others.

Although the AbstractionLevel DataType in MPEG-7 FDIS is apparently claimed to address abstraction and instantiation, its usage and semantics have not been clarified enough to actually employ it. What follows should provide a far simpler and practically usable set of tools for abstraction and instantiation.

2 Specification

The `copy` and `substitute` attributes are applicable to all the DS instances by amending the definition of DSType as follows.

```
<!-- ##################################################  -->
<!-- Definition of the DSType                            -->
<!-- ##################################################  -->

<complexType name="DSType" abstract="true">
  <complexContent>
```

```
    <extension base="mpeg7:Mpeg7BaseType">
      <sequence>
        <element name="Header" xsi:type="“:HeaderType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
      <attribute name="id" type="ID" use="optional"/>
      <attribute name="copy" type="anyURI" use="optional"/>
      <attribute name="substitute" type="anyURI" use="optional"/>
      <attributeGroup ref="mpeg7:timePropertyGrp"/>
      <attributeGroup ref="mpeg7:mediaTimePropertyGrp"/>
    </extension>
  </complexContent>
</complexType>
```

Semantics of these additional attributes is:

Name	*Definition*
`copy`	Points the description (XML element) to copy. It is not content (data) but description that is copied. The description in the pointed element except for ID attributes is copied while substituting the parts as specified by the `substitute` attributes if any.
`substitute`	Points the element to substitute with the current element while copying the element pointed by the `copy` attribute of the parent element.

3 Application

Abstraction is a very common tool employed in programming languages as reentrant codes such as loops, procedures, functions, macros, and so forth. Computer programming is practically impossible without them. Abstraction generally abound also in traditional documentation, perhaps in technical documentation in particular. Documents introduce new terms and refer to them afterwards to both simplify the description and highlight common structures.

Multimedia documents can equally enjoy such functionalities as well. For instance, the example descriptions attached to MPEG-7 FDIS include a number of descriptions of the following same form, as children of `<Collection>` and `<ContentCollection>` elements.

```
<Content xsi:type=”ImageType”>
    <Image>
        <MediaLocator xsi:type=”ImageLocatorType”>
            <MediaUri>soccer1.jpg</MediaUri>
        </MediaLocator>
    </Image>
</Content>
```

We can both simplify the whole description and highlight this repetition by replacing each but one of these descriptions with:

```
<Content copy=”#soccer1”/>
```

if there is one description as follows.

```
<Content xsi:type=”ImageType” id=”soccer1”>
    <Image>
        <MediaLocator xsi:type=”ImageLocatorType”>
            <MediaUri>soccer1.jpg</MediaUri>
        </MediaLocator>
    </Image>
</Content>
```

(This case may be more adequately dealt with by employing another new attribute `equal` instead of `copy`.)

Below is an example of linguistic data containing an ellipsis of ‘wants to date with’ in the second conjunct (`copy=”#TM”` may be omitted here).

```
<Sentence synthesis=”coordination”>
  <Phrase id=”TM”>
    <Phrase id=”TOM”>Tom</Phrase>
    wants
    <Phrase>to
      <Phrase>date
        <Phrase>with
          <Phrase id=”MARY”>Mary</Phrase>
        </Phrase>
    </Phrase>
  </Phrase>
  and
  <Phrase copy=”#TM”>
    <Phrase substitute=”#TOM”>Bill</Phrase>,
    <Phrase substitute=”#MARY”>Sue</Phrase>
  </Phrase>.
</Sentence>
```

This ellipsis is explicitly interpreted by copying `love’ from the first conjunct to the second, which is described by the copy and substitute attributes. This description does not mean that the original linguistic data is ‘Tom wants to date with Mary and Bill wants to date with Sue’ because what is copied is not the original content but the description.

For another instance, consider the following semantic description, which may be interpreted that somebody (`kisser`) kisses Mary.

```
<Semantics id=”kissing”>
    <SemanticBase xsi:type=”ObjectType” id=”kisser”/>
    <SemanticBase xsi:type=”EventType”>
        <Relation type=”:r:agent” target=”#kisser”/>
        <Relation type=”:r:object” target=”#Mary”/>
    </SemanticBase>
</Semantics>
```

This may be reused as a template for describing other semantic content, as follows, which means that somebody (probably other than the above `kisser`) kisses Mary as well.

```
<Semantics copy=”#kissing”>
  <SemanticBase subsitute=”#kisser”>
    <MediaLocator …>…</MediaLocator>
  </SemanticBase>
</Semantics>
```

Another effective application of abstraction and instantiation is to summary. Given a summary, you can make other similar summaries by shrinking or extending parts of it.

Reference

[1] Kôiti Hasida and Riccardo Leonardi. *Abstraction and Instantiation*. ISO/MPEG M7048, MPEG Singapore Meeting, March 2001.

[2] Kôiti Hasida and Riccardo Leonardi. *Workplan of CE on Abstraction and Instantiation*. ISO/MPEG W3970, MPEG Singapore Meeting, March 2001.

[3] Kôiti Hasida and Riccardo Leonardi. *Report of CE on Abstraction and Instantiation.* ISO/MPEG M7295, MPEG Sydney Meeting, July 2001.