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
MPEG96/M1055
Tampere, Finland

Title: **Adding Parameter Types to MSDL-S**
Source: **Alexandros Eleftheriadis and Yihan Fang (Columbia University)**
Status: **Proposal**
Group: **MPEG-4 Integration/MSDL**
Date: **June 30, 1996**

1. Introduction

We propose the addition of *parameter types* in MSDL-S, to rectify problems of object data member references that are frequently encountered when describing complex bitstreams using MSDL-S. These references are associated with context (flags, conditions, etc.) that is needed in order to make parsing decisions within objects, and require that information from other objects is available. In keeping with the declarative nature of MSDL-S, such referenced objects are not instantiated at the time the syntax is declared, and hence it is possible to create inconsistent code (referencing an object that has not been instantiated yet). Furthermore, current practice makes it difficult to utilize multiple instances of similar objects when both have to be used in context-sensitive parsing of multiple instances of another object. The concept of parameter types provides a mechanism with which all these issues are resolved without altering the current design of MSDL-S. Very simply, parameter types can be thought of as class parameters (similar to the arguments or parameters of a function) that have to be specified when an object is instantiated. An equivalent mechanism using C++ as an example would be the definition of a class constructor which has a mandatory (and fixed) set of arguments.

2. Parameter Types

A parameter type defines a class with parameter(s). This is to address cases where the data structure of the class is dependent on variables of one or more other objects. The concept of parameter types is consistent with the key design principle in MSDL-S, i.e., a declarative approach that allows conditional parsing in specifying syntax data structures and avoids the use of function calls. In fact, parameter types are a very intuitive facility and—after they are defined—can be seen to be widely needed in representing coded audio-visual bitstreams in MSDL-S.

The key issue which parameter types solve is access to information from other objects. Since MSDL-S pertains to a class, rather the specific instance of a given object, it is not appropriate to use references to other *instantiated* objects. This violates the separation between run-time instances and compile-time declarations, and also prohibits instantiation of objects with differing contexts. Using parameter types, objects that are needed as context in the declaration of parsable components of a class are declared as "parameters" of the class. Instances of such a class can only be created with these parameters specified.

More specifically, the template for class declarations (Rule 3 of Part 5 of the MSDL Working Draft [1]) is modified as follows:

```
[modifiers] class class_name [(parameter list)] [is parent_class] [:
        [aligned] bit(length) [id_name]=objet_id|id_range]{
    [element; ...]          // zero or more elements
};
```

A class that uses parameter types is dependent on the objects (could be class objects or simple variables) in its parameter list. When instantiating such a class into an object, the parameters in the parameter list have to be instantiated objects of their corresponding classes or types. For example,

```
class A {
      // class body
      ...
};
      class B {
      int i;
      A a;
      C(a, i) foo;
};
class C(A a, int i){     // class C uses parameter types
      ...
      if (i) {
      ...
      }
};
```

A parameter type cannot be instantiated without first instantiating the required dependent instances. It will be instantiated automatically when a "compositing" class (a class that contains an instance of this class) is instantiated.

We should note that parameter types can be thought of as equivalent C++ constructors with mandatory parameters. In the case of MSDL-S, where constructors are outside its scope, parameter types are a natural way to convey identical information.

3. Example: Video VM

As an example of the use of parameter types from the Video VM, the `VOP` class has to use parameter types since the content of the VOP layer in the bitstream depends on the value of `more_than_one_VOP`, which in turn is defined in the class `Session`. The code would be as follows.

```
class Session:
        const bit(32) session_start_code=0x000001B0 {
      uint(1) more_than_one_VOP;
      ...
      while (![session_end_code])
      VOP vop(more_than_one_VOP);
      const bit(32) session_end_code=0x000001B1;
};
```

```
class VOP(int more_than_one_VOP):
      const bit(27) VOP_start_code=0b0000.0000.0000.0000.0000.0001.111{
      ...

      if (VOP_visibility && more_than_one_VOP) {
            uint(5) VOP_composition_order;
            ...
      }
...
};
```

Draft versions of the bitstream syntax representations of the MPEG-2 Video IS and the MPEG-4 Video VM Version 2.2 using parameter types are available in the following URLs:

MPEG-2 Video IS: `http://www.itnm.columbia.edu/~fang/MSDL/MPEG2.html`
MPEG-4 Video VM: `http://www.itnm.columbia.edu/~fang/MSDL/VideoVM.html`

References

[1] ISO/IEC SC29/WG11 N1246, MSDL Specification, Working Draft Version 1.1, Florence, Italy, March 1996.