2 Representing Object Structures i1ruct;yL

3.2 Complex Attributes

An attribute in an object can be a container for a collection of attributes under a common heading. In that case, the attribute is said to be *complex* because its value is not a simple scalar. This is roughly equivalent to a \struct" in the language C.

In UML, a complex type is defined as a separate class. The following is an example of a class, `MyTest`, having three attributes, `attributeA`, `attributeB` and `attributeC`. The first two attributes have simple types, and the third attribute's type is a newly-defined class, `AttrCType`, itself containing two more attributes.

In a programming language, given an object `obj` of class `MyTest`, the attributes might be accessed as

```
obj.attributeA
obj.attributeB
obj.attributeC.anotherAttribute
obj.attributeC.yetAnotherAttribute
```

3.3 Links

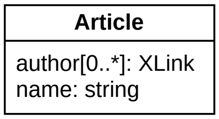
```
<xs:complexType name="XLink">
```

3.5 Lists

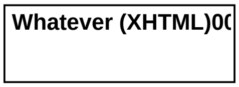
An attribute can be a list of simple types, or a list of complex types, or a list of link or inclusion types. All items in the list must have the same type. In some programming languages such as Java or C, a list might be represented as a vector or array.

In the diagrammatic and textual forms of the current notation, lists are expressed using a style loosely based on C and Java-style array notation, with a multiplicity specifier enclosed in square brackets. The multiplicity specifier consists of numerals or the asterisk character, optionally separated by commas or `..' (the last to indicate a range). Asterisk means \zero or more". For example, \somevar[10]: integer" means that somevar

1	exactly one
0,1	zero or one
0..4	between zero and four
3,7	either three or seven
0..*	zero or more
*	zero or more
1..*	one or more


Article
author[0..*]: XLink
name: string

```
<xs:complexType name="Article">
```



```
<xs:complexType name="Whatever" content="textOnly">
  <xs:attribute name="bigDeal" type="xs:string"/>
  <xs:attribute name="type" use="fixed" value="XHTML"/>
</xs:complexType>
```

UML Form *XML Schema Form*

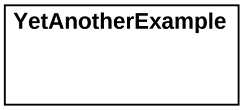
This allows the following kind of XML data object:

```
<Whatever bigDeal="This is an attribute">
  This has a value, but no subelements.
</Whatever>
```

example of an XML Schema constraining the values of an attribute to a limited set of three specific strings for the class `AnExample` defined above:

```
<xs:complexType name="AnExample">
  <xs:attribute name="attrA" type="xs:integer"/>
  <xs:attribute name="attrB">
    <xs:simpleType base="xs:string">
      <xs:enumeration value="val1"/>
      <xs:enumeration value="val2"/>
      <xs:enumeration value="val3"/>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

Optional attribute. Sometimes an attribute in a class should be considered optional. For these situations, the attribute should be given the XML Schema property `minOccurs="0"`. (The default value of `minOccurs` is 1.)



UML Form

```
<xs:complexType name="YetAnotherExample">
  <xs:attribute name="intValue"  type="xs:integer" minOccurs="0"/>
  <xs:attribute name="dateValue" type="xs:date"/>
</xs:complexType>
```

XML Schema Form

;inimum and maximum range values on numeric attributes . The upper and lower value boundaries for an attribute having a numeric value can be defined using the XML Schema `minExclusive`, `minInclusive`, `maxInclusive`, and `maxExclusive` properties. For instance:

The first approach is to define, for each relevant attribute, another attribute whose name has the suffix _units

Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.

St. Laurent, S. (2000). *XML Elements of Style*. New York: McGraw-Hill.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema Part 1: Structures (W3C Working Draft 7 April 2000). Available via the World Wide Web at `http://www.w3.org/TR/xmlschema-1/`.