ComplexType

Complex type maps to UML Class with the stereotype XSDcomplexType.

- abstract - to UML Class abstract value(true | false).
- annotation - to UML Class documentation.
- attribute – to inner UML Class Attribute or UML Association End.
- attributeGroup – to UML AssociationEnd or UML Attribute with type XSDattributeGroup.
- name – to UML Class name.

This class also can have the stereotypes XSDsimpleContent, XSDcomplexContent, XSDall, XSDchoice, and XSDsequence.

No stereotype – the same as “XSDsequence”.

A generalization between a complex type and another type has the stereotype XSDrestriction or XSDextension. We assume the stereotype XSDextension if the generalizations do not have stereotypes.

Some complex mapping:

- complexType with simpleContent – to UML Class. This class must be derived from another class and must have stereotype XSDsimpleContent.
- complexType with complexContent – to UML Class. This class must be derived from another class and must have stereotype XSDcomplexContent.

complexType with group, all, choice or sequence – to UML class with appropriate stereotype

```
<complexType

        abstract = boolean : false

        block = (#all | List of (extension | restriction))

        final = (#all | List of (extension | restriction))
        id = ID

        mixed = boolean : false

        name = NCName

        {any attributes with non-schema namespace…}>

        Content: (annotation?,(simpleContent | complexContent | ((group | al
((attribute | attributeGroup)*, anyAttribute?))))

</complexType>
```

When you choose the <simpleContent> alternative, the following elements are relevant, and the remaining property mappings are as below. Note that you must either choose <restriction> or <extension> as the content of <simpleContent>.

<restriction> chosen as the content of <simpleContent>

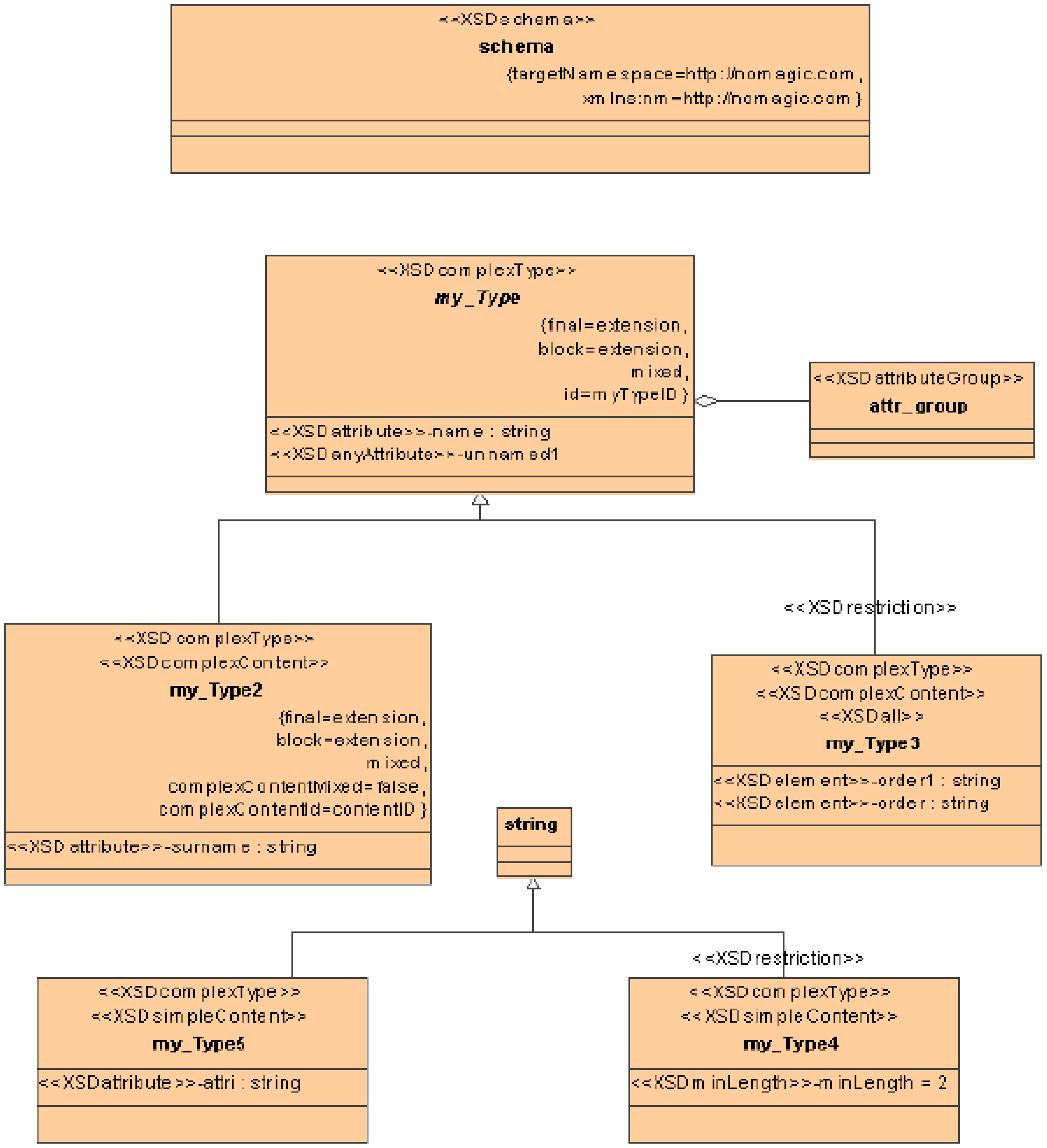
```
<simpleContent

        id = ID

        {any attributes with non-schema namespace…}> Content: (annotation?, (restriction | extension))
        </simpleContent>

<restriction

        base = OName

        id = ID

        {any attributes with non-schema namespace…}>

        Content: (annotation?, (simpleType?, (minExclusive | minInclusive  | maxExclusive |

maxlnclusive | totalDigits | fractionDigits | length | minLength | maxLength

whitespace | pattern) *)?, ((attribute | attributeGroup)*, anyAttribute?))

</restriction>
<extension

        base = OName

        id = ID

        {any attributes with non-schema namespace…}>

        Content: (annotation?, ((attribute | attributeGroup)*, anyAttribute?))

</extension>

<attributeGroup

        id = ID

        ref = OName

        {any attributes with non-schema namespace…}>

        Content: (annotation?)

</attributeGroup>

<anyAttribute
```

When the <complexContent> alternative is chosen, the following elements are relevant (as are the <attributeGroup> and <anyAttribute> elements, not repeated here), and the additional property mappings appear below. Note that you must choose either <restriction> or <extension> as the content of <complexContent>, but their content models differ in this case from the case above when they occur as children of <simpleContent>.

The property mappings below are also used in the case where the third alternative (neither <simpleContent> nor <complexContent>) is chosen. This case is understood as shorthand for complex content restricting the **ur type definition**. The details of the mappings should be modified as necessary.

Sample of either <restriction> or <extension> must be chosen as the content of <complexContent>

```
<complexContent

        id = ID

        mixed = boolean

        {any attributes with non-schema namespace…}>
        Content: (annotation?, (restriction | extension))

</complexContent>

<restriction

        base = OName

        id = ID

        {any attributes with non-schema namespace…}>

        Content: (annotation?, (group | all | choice | sequence)?, ((attribute | attributeGroup)*,
anyAttribute?))

</restriction>

<extension

        base = OName

        id = ID

        {any attributes with non-schema namespace…}>

        Content: (annotation?, ((group | all | choice | sequence)?, ((attribute | attributeGroup)*,
anyAttribute?)))

</extension>
```



Example of complexType UML model.

Example of complexType UML model with associated XML

```
<?xml version='1.0' encoding = 'Cp1252'?›

<xs:schema xmlns:nm = "http://nomagic.com" xmlns:xs = "http://www.w3.org/2001/XMLSchema" targetNamespace = "http
://nomagic.com" >

        <xs:complexType name = "my_Type2" block = "extension" final = "extension" mixed = "true" >

                <xs:annotation >

                        <xs:documentation >my_type2

Documentation</xs:documentation>

                </xs:annotation>

                <xs:complexContent id = "contentID" mixed = "false" >
                        <xs:extension base = "nm:my_Type">

                                <xs:attribute name = "surname" type = "xs:string" />

                        </xs:extension>

                </xs:complexContent>
        </xs:complexType>

        <xs:complexType name = "my_Type3" >

                <xs:complexContent >

                        <xs:restriction base = "nm:my_Type">

                                <xs:all >

                                        <xs:element name = "order" type = "xs:string"/>

                                        <xs:element name = "order1" type = "xs:string"/>

                                </xs:all>
                        </xs:restriction>

                </xs:complexContent>
        </xs:complexType>

        <xs:complexType name = "my_Type4">

                <xs:simpleContent >

                        <xs:restriction base = "xs:string" >

                                <xs:minLength value = "2" />

                        </xs:restriction>

                </xs:simpleContent> </xs:complexType>

        <xs:complexType name = "my_Type5">

                <xs:simpleContent >

                        <xs:extension base = "xs:string">

                                <xs:attribute name = "attri" type = "xs:string"/>

                        </xs:extension>

                </xs:simpleContent>
        </xs:complexType>

        <xs:complexType name = "my_Type" abstract = "true" block = "extension" final = "extension" id =
"myTypeID" mixed = "true" >

                <xs:annotation >

                        <xs:documentation >my_type documentation</xs:documentation>
                </xs:annotation>

                <xs:attribute name = "name" type = "xs:string" />
                <xs:attributeGroup ref = "nm:attr_group" />

                <xs:anyAttribute/>
        </xs:complexType>

        <xs:attributeGroup name = "attr_group" />

</xs:schema>
```