

1
2
3
4
5
6
7
8

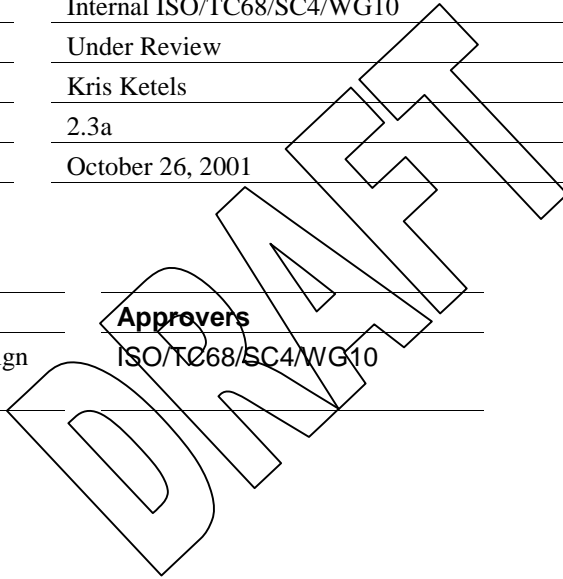
ISO15022 XML design rules

Technical Specification

Confidentiality	Internal ISO/TC68/SC4/WG10
Status	Under Review
Author	Kris Ketels
Document Version	2.3a
Document Issue Date	October 26, 2001

9
10

Reviewers	Approvers
ISO/TC68/SC4/WG10 Design Rules PT	ISO/TC68/SC4/WG10



11 **Table of contents**

12	1	Introduction	3
13	2	Mapping rules from UML to ISO XML	3
14	2.1	General mapping rules	3
15	2.2	ISO XML elements	4
16	2.2.1	ISO XMLTag	4
17	2.2.2	xsi:type	4
18	2.2.3	Representation Class Attribute.....	5
19	2.3	Specific mapping rules	6
20	2.3.1	Data types.....	6
21	2.3.2	Class	24
22	2.3.3	Simple composition.....	25
23	2.3.4	Class attributes	26
24	2.3.5	Composition of vectorial attributes (Collections).....	28
25	2.3.6	Inheritance.....	30
26		Enumerated roles using XOR invariant	32
27		Enumerated attributes using XOR invariant	35
28	2.3.9	Enumerated roles and attributes using <<choice>> stereotype	36
29	3	Schema design rules	40
30	3.1	Common design rules and usage.....	40
31	3.2	Schema Design rules	40
32	3.2.1	XML name clash support within the scope of a message	40
33	3.2.2	XML schema features used in ISO XML	41
34	3.3	Granularity of Schemas	45
35	3.4	Summary of UML invariants related to schema production.....	46
36	4	Character set	47
37		End of document	49
38			

39 1 Introduction

40 XML is a technical standard defined by W3C (the World Wide Web Consortium) and
41 leaves a lot of freedom for the exact way it is used in a particular application. Therefore,
42 merely stating that XML is used is not sufficient, one must also explain HOW it will be
43 used.

44 The use of XML is part of the overall approach for the development. This development
45 focuses on the correct definition of a business standard using modelling techniques. The
46 resulting business standard is captured in UML (Unified Modelling Language¹) and is
47 stored in an electronic repository, the “ISO Repository”. Business messages are defined in
48 UML class diagrams and XML is then used as a physical representation (i.e. the syntax) of
49 the defined business messages. A set of **XML design rules**, called **ISO XML**, define in a
50 very detailed and strict way how this physical XML representation is derived from the
51 business message in the UML class diagram.

52 This document explains these XML design rules.

53 This document does NOT explain how a message should be created in UML. It explains,
54 once a message is created in UML, how it will be mapped into XML.

55

56 2 Mapping rules from UML to ISO XML

57 2.1 General mapping rules

58 Mapping rules from UML to ISO XML are governed by the following design choices:

- 59 • ISO XML representation to be as structured as possible:
 - 60 – Business information is expressed as XML elements/values;
 - 61 – Metadata information is expressed as XML attributes. XML attributes are not to be
62 conveyed ‘on the wire’ in the XML instance, unless required to remove ambiguity.
- 63 • The current work is based on W3C’s Recommendation of May, 2001.
- 64 • The names used in ISO XML are the XML names or, when absent, the UML names.
- 65 • ISO XML elements are derived from the UML representation of a business message.
66 They can only be derived from UML-classes, UML-roles or UML-attributes.
- 67 • Each ISO XML element must be traceable to the corresponding UML model element.

¹ You can find more information about UML on the Object Management Group website at:
<http://www.omg.org/uml>

- 68 • Currently ISO XML only runtime Schemas are generated. Runtime schema's only
69 contains information required to validate XML instances. No documentation nore
70 implementation information (e.g elementID, version, etc.) is mentioned.

71 2.2 ISO XML elements

72 For the ISO XML runtime Schema, any ISO XML element has the following structure:

73 <ISOXMLTag [xsi:type="class_name"] [RepresentationClassAttribute="value"]>

74

75 2.2.1 ISO XMLTag

76 ISO XMLTag is assigned according to following rules:

77 For a ISO XML element derived from a class if that class contains the stereotype
78 <<message>>²:

- 79 ▪ The XML name of the class or by default the name of the class.

80 For a ISO XML element derived from a role:

- 81 ▪ The XML name of the role or by default the name of the role. If no rolename is
82 specified in the UML model, the name (XML name or name by default) of the class
83 which is at the end of the aggregation.

84 For a ISO XML element derived from an attribute:

- 85 ▪ The XML name of the attribute or by default the name of the attribute.

86 2.2.2 xsi:type

87 2.2.2.1 In the schema

88 By using xsi:type in the instance, the schema does not need to define any additional
89 attribute on types. The xsi:type implicitey refers to a type defined in the schema.

90 2.2.2.2 In the corresponding instance

91 In case of polymorphism, the attribute "xsi:type" is required to choose the desired type in
92 the ISO XML instance.

93 *summarizing:*

ISO XML element	Type
-----------------	------

² Classes that don't contain the stereotype <<ISOmessage>> do NOT have a corresponding XML element.

derived from	
Class	Class name
Role	Name of the class at the end of the aggregation
Attribute	<ul style="list-style-type: none">• Name of the class of the attributes' type• attribute type name (for primitive types)

94 Remark: by name, it is meant the XML name or by default the UML name.

95

96 **2.2.3 Representation Class Attribute**

97 When user defined-datatypes are stereotyped by a certain representation classes, an XML
98 attribute might be required to remove ambiguity. See the [chapter on data types](#) for more
99 details.

DRAFT

100

101 **2.3 Specific mapping rules**

102 All model elements, defined accordingly to the methodology, are based on following UML
103 structures. Hence, by defining the conversion rules from those structures into ISO XML we
104 can convert any UML model into its corresponding ISO XML Schema and instance.

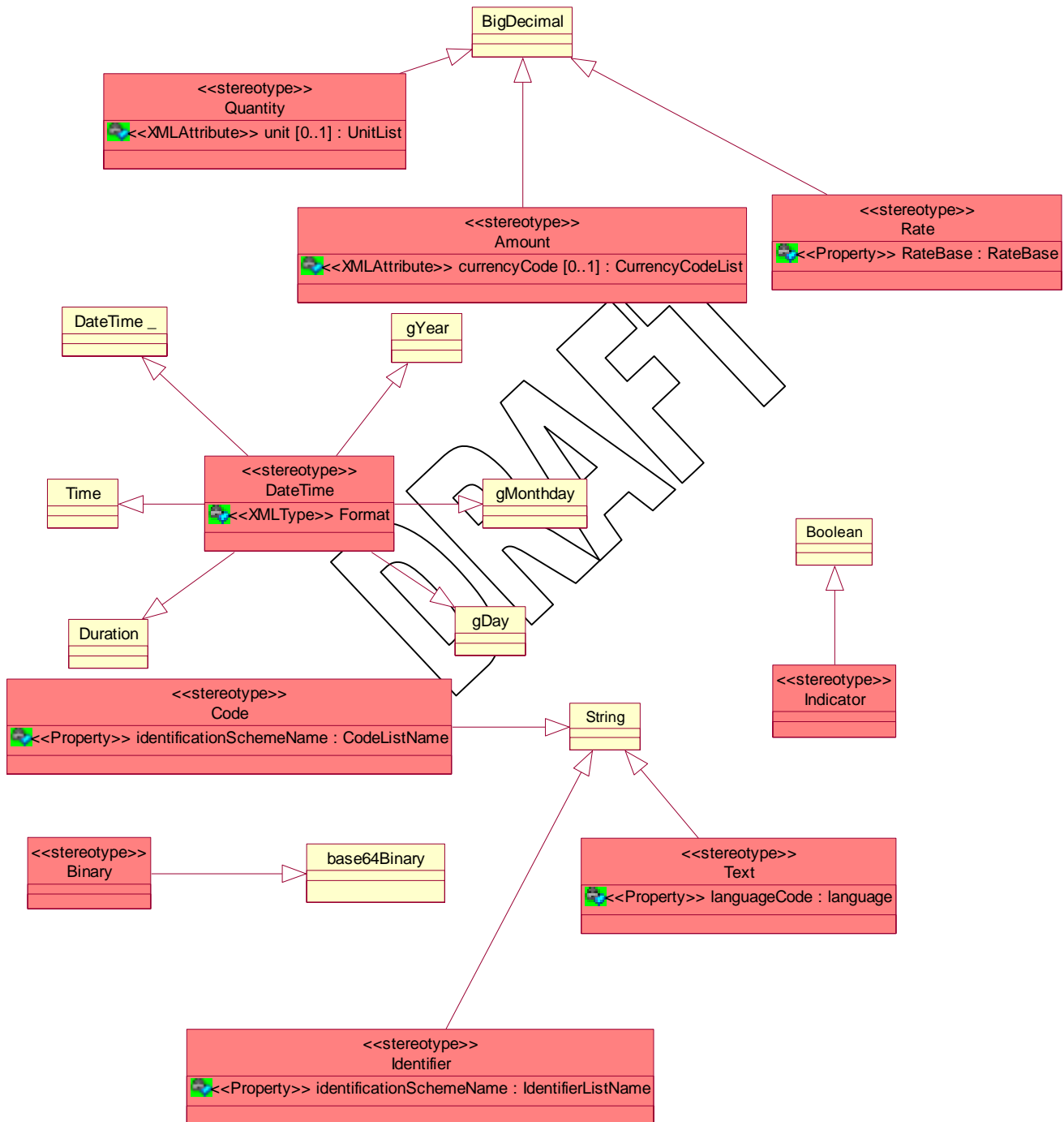
105 **2.3.1 Data types**

106 In a message model, all class attributes have a type, which we call **data types** for the
107 purpose of this chapter. Data types define the structure of a class attribute.

DRAFT

108

109 **2.3.1.1 Representation class meta-model**



Notes:

Each data type is identified by a class diagram and stereotyped by a representation class. A representation class has a number of characteristics that are passed on ('inherited by') all data types that are using that representation class. In this way, characteristics common to a number of datatypes are grouped together.

Stereotype <<XMLAttribute>> indicates that the values this attribute can be declared in the XML Schema in case of ambiguity, and will appear in the XML instance.

Stereotype <<Property>> indicates that the values this attribute will NOT be declared in the XML Schema, but is a property inherent to this datatype.

Stereotype <<XMLType>> (only used in representation class DateTime) indicates that any user defined data type will have to declare the primitive datatype (Time, gDay, gMonth,...) it will use.

122

123

2.3.1.2 Primitive Data types

ISO XML primitive data types are encoded as defined by W3C, defined at <http://www.w3.org/TR/xmlschema-2/#dt-encoding>. Following XML primitive types are supported:

128

UML Name	XML Name	Description
String	string	Set of finite sequences of UTF-8 characters
Boolean	boolean	Has the value space of boolean constants "True" or "False"
Integer	integer	Corresponds to 32 bits integer type
BigDecimal	decimal	Arbitrary precision decimal numbers
Date	date	Corresponds to a date. See ISO 8601. Format CCYY-MM-DD
Time	time	Corresponds to a time. See ISO8601. Format HH:MM:SS +- offset to UTC
DateTime	dateTime	Corresponds to a date and time. See ISO8601. Format CCYY-MM-DDTHH:MM:SS +- offset to UTC
Duration	duration	Corresponds to a period in time. See ISO8601. Format PnYnMnDTnHnMnS
gDay	gDay	It is a set of one-day long, annually periodic instances. The time zone must be UTC. Lexical representation:--MM-DD.
gMonth	gMonth	Represents a time period that starts at midnight on the first day of the month and lasts until the midnight that ends the last day of the month. Lexical representation: --MM--.

gYear	gYear	Represents a time period that starts at the midnight that starts the first day of the year and ends at the midnight that ends the last day of the year. It is a set of one-year long, non-periodic instances. Lexical representation: CCYY
gMonthday	gMonthday	It is a set of one-day long, monthly periodic instances. Lexical representation: ---DD. The time zone must be UTC.
base64Binary	base64Binary	represents Base64-encoded arbitrary binary data

129

130 2.3.1.3 User-defined data types

131 It is possible to define non-primitive data types by deriving either from a primitive type or
 132 from another non-primitive data type. Remark that in UML neither primitive nor non-
 133 primitive data types may have attributes. Those non-primitive datatypes can be used as
 134 UML types for UML attributes with the added benefit that the value space of the original
 135 primitive type (e.g. String) can be constrained by introducing invariants on the non-
 136 primitive data type. Those invariants will be mapped to facets when generating XML
 137 Schemas.

138 In order to apply facets, the XML types that are generated for those data types must be
 139 simpleTypes or complexTypes with simpleContent, and not complexTypes³.

140 A user-defined data type maps to an XML SimpleType. This SimpleType restricts an XML
 141 primitive type.

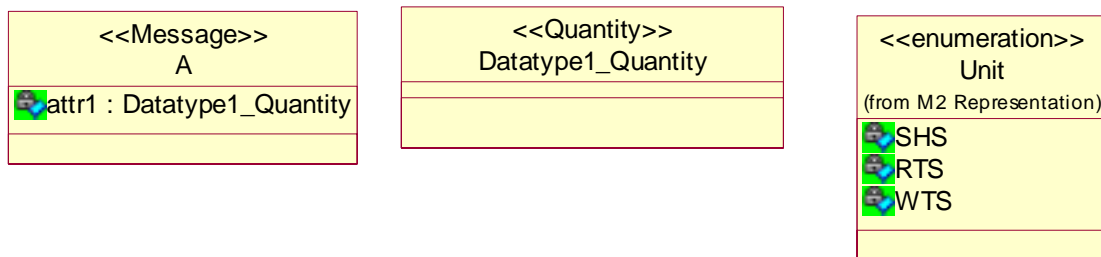
142 Where necessary (in case of ambiguity), the representation class attribute maps to an XML
 143 attribute.

144

145

146 2.3.1.3.1 Data type using representation class <<Quantity>>

147



³ XML Schema validation constraint: Facets cannot be applied to complexTypes without simpleContent.

148

149 Properties:

- 150 • Since the representation class Quantity (see metamodel) has an attribute with a type
151 named Unit which is stereotyped as being a <<XMLAttribute>>, the corresponding
152 Schema defines for element <attr1> an attribute named 'unit' with a enumerated list of
153 values a specified in the Class 'Unit'.
- 154 • An enumerated value is constrained within a list of possible values.
- 155 • The values for the enumerated items are taken from the UML initial value given to each
156 of the UML enumerated attributes.

157

158 Suppose this data type has an additional constraint (=XML facet) that the maximum
159 quantity may not exceed 20000 units.

160

161 Instance:

```
162 <A>
163   <attr1 unit="SHS">1000</attr1>
164 </A>
```

165

166 Schema:

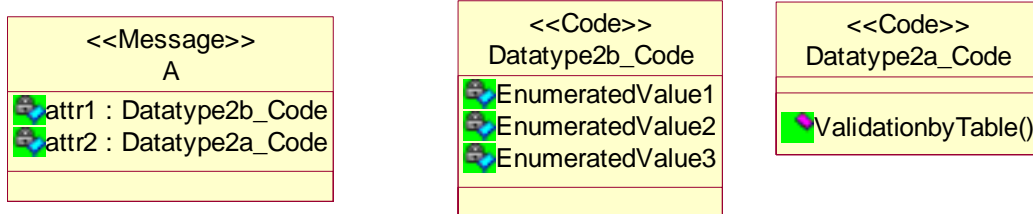
```
<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype1_Quantity"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Datatype1_Quantity">
  <xs:simpleContent>
    <xs:restriction base="xs:decimal">
      <xs:maxInclusive value="20000">
        <xs:attribute name="unit" type="Unit"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>

<xs:simpleType name="Unit">
  <xs:restriction base="xs:string">
    <xs:enumeration value="SHS"/>
    <xs:enumeration value="RTS"/>
    <xs:enumeration value="WTS"/>
  </xs:restriction>
</xs:simpleType>
```

167

168 **2.3.1.3.2 Data type using representation class <<Code>>**

169

170

171 Properties:

- 172 • Each user-defined datatype using <<Code> can indicate whether the list is an internal
173 list (i.e. specified in the schema), or external (i.e. not specified in the schema). This is
174 done using the invariant 'ValidationbyTable'. Datatype2b_Code is an enumeration of
175 which one of the Enumerated Values has to be chosen in the instance.
- 176 • An enumerated value is constrained within a list of possible values.
- 177 • The values for the enumerated items are taken from the UML initial value given to each
178 of the UML enumerated attributes.

179

UML	ISO XML instance
Class contains an enumeration of possible values	ISO XML element contains the chosen value

180

181 Instance:

```

182 <A>
183   <attr1>EnumeratedValue2</attr1>
184   <attr2>AnythingGoesHere</attr2>
185 </A>
  
```

186

187 Schema:

```

<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype2b_Code"/>
    <xs:element name="attr2" type="xs:Datatype2a_Code"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="Datatype2b_Code">
  <xs:restriction base="xs:string">
    <xs:enumeration value="EnumeratedValue1"/>
    <xs:enumeration value="EnumeratedValue2"/>
    <xs:enumeration value="EnumeratedValue3"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Datatype2a_Code">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

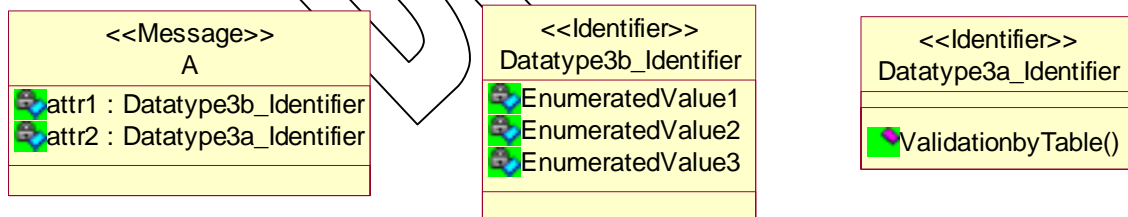
```

188

189

190 **2.3.1.3.3 Data type using representation class <<Identifier>>**

191



192

193 **Properties:**

- 194 • Each user-defined datatype using <<Identifier> can indicate whether the list is an
195 internal list (i.e. specified in the schema), or external (i.e. not specified in the schema).
196 This is done using the invariant 'ValidationbyTable'. Datatype3b_Identifier is an
197 enumeration of which one of the Enumerated Values has to be chosen in the instance.
- 198 • An enumerated value is constrained within a list of possible values.
- 199 • The values for the enumerated items are taken from the UML initial value given to each
200 of the UML enumerated attributes.

201

UML	ISO XML instance
Class contains an enumeration of possible values	ISO XML element contains the chosen value

202

203 Instance:

204

205

206

207

```
<A>
  <attr1>EnumeratedValue2</attr1>
  <attr2>AnythingGoesHere</attr2>
</A>
```

208

209 Schema:

```
<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype3b_Identifier"/>
    <xs:element name="attr2" type="xs:Datatype3a_Identifier"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="Datatype3b_Identifier">
  <xs:restriction base="xs:string">
    <xs:enumeration value="EnumeratedValue1"/>
    <xs:enumeration value="EnumeratedValue2"/>
    <xs:enumeration value="EnumeratedValue3"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Datatype3a_Identifier">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>
```

210

211

212

213

214

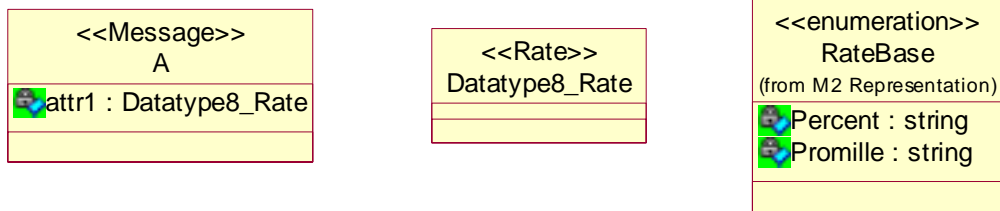
215

216

217

218 **2.3.1.3.4 Data type using representation class <<Rate>>**

219



220

221 Properties:

- 222 • Since the representation class Rate (see metamodel) has a meta-attribute with a type
223 named RateBase which is stereotyped as being a <<Property>>, the corresponding
224 Schema will not declare this attribute.
- 225 • An enumerated value is constrained within a list of possible values.
- 226 • The values for the enumerated items are taken from the UML initial value given to each
227 of the UML enumerated attributes.

228

229

230 Instance:

```

231 <A>
232   <attr1>95.6</attr1>
233 </A>
  
```

234

235 Schema:

```

<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype8_Rate"/>
  </xs:sequence>
</xs:complexType>

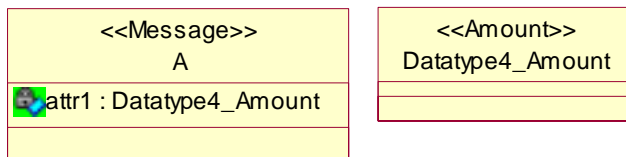
<xs:simpleType name="Datatype8_Rate">
  <xs:restriction base="xs:decimal">
    </xs:restriction>
</xs:simpleType>
  
```

236

237

238 **2.3.1.3.5 Data type using representation class <<Amount>>**

239



240

241 Properties:

- 242 • Since the representation class Amount (see metamodel) has an attribute with a type
243 named CurrencyCode which is stereotyped as being a <<XMLAttribute>>, the
244 corresponding Schema should define for element <attr1> an attribute named
245 'currencyCode' with a enumerated list of values as specified in the Class
246 'CurrencyCode'. However in this case, since we do not own this list (owned by ISO), it
247 is considered to be an external list to avoid having to update the standard each time one
248 of the values of the code list changes. Hence the XML attribute must appear in the
249 instance (to avoid ambiguity), but the content is NOT validated by Schema.

250

251

252 Instance:

```

253 <A>
254   <attr1 currencyCode="USD">95.6</attr1>
255 </A>
  
```

256

257 Schema:

```

<!-- <message>> A -->
<xs:element name="A" type="A" />

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype4_Amount" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Datatype4_Amount">
  <xs:simpleContent>
    <xs:restriction base="xs:decimal">
      <xs:attribute name="currencyCode" type="CurrencyCode" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="CurrencyCode">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

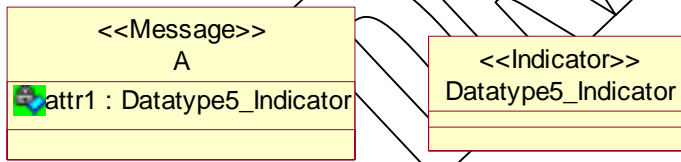
```

258

259

260 **2.3.1.3.6 Data type using representation class <<Indicator>>**

261



262

263 **Properties:**

- 264 • A datatype stereotyped by representation class <<Indicator>> indicates that the attribute
265 must have a Boolean value (true or false).

266

267

268 **Instance:**

```

269 <A>
270   <attr1>true</attr1>
271 </A>

```

272

273 **Schema:**


```

<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype5_Indicator"/>
  </xs:sequence>
</xs:complexType>

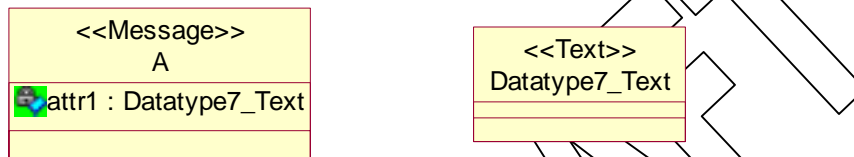
<xs:simpleType name="Datatype5_Indicator">
  <xs:restriction base="xs:boolean">
  </xs:restriction>
</xs:simpleType>

```

274

275 **2.3.1.3.7 Data type using representation class <<Text>>**

276



277

278 Instance:

```

279 <A>
280   <attr1>any narrative text</attr1>
281 </A>

```

282

283 Schema:

```

<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype7_Text"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="Datatype7_Text">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

```

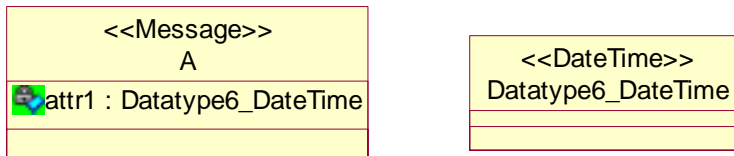
284

285

286 **2.3.1.3.8 Data type using representation class <<DateTime>>**

287

288



289

290

291 Properties:

- 292 • Representation class 'DateTime' has a meta attribute Format which is stereotyped
293 <<XMLType>>. This means that any datatype that is using representation class
294 <<DateTime>> has to indicate from which XML primitive datatype it is restricting.
- 295 • Suppose an additional constraint is added namely that the date should be equal or later
296 than January first, 2002.

297

298

299 Instance:

```
300 <A>
301   <attr1>2002-11-23</attr1>
302 </A>
```

303

304 Schema:

```
<!-- <<message>> A -->
<xs:element name="A" type="A"/>

<!-- class: A -->
<xs:complexType name="A">
  <xs:sequence>
    <xs:element name="attr1" type="xs:Datatype6_DateTime"/>
  </xs:sequence>
</xs:complexType>

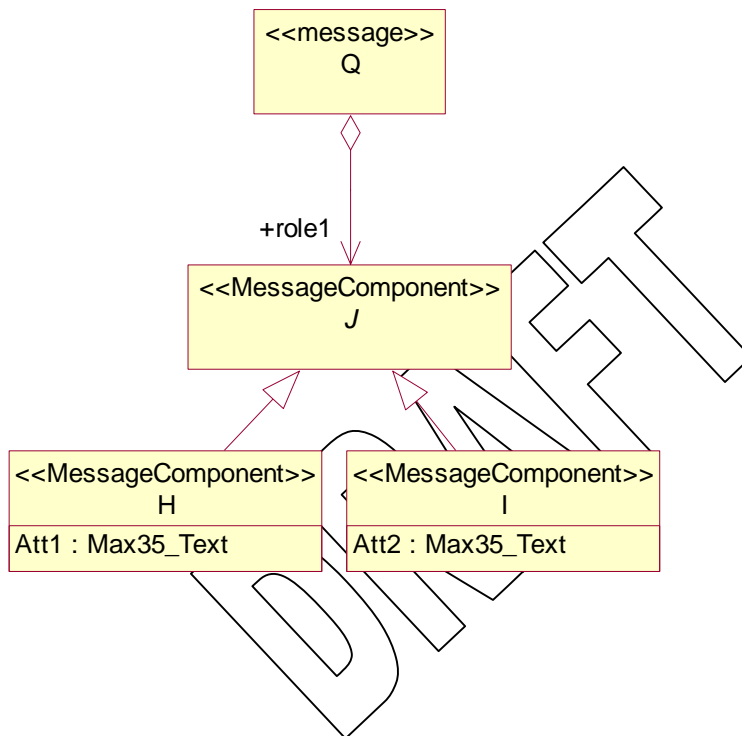
<xs:simpleType name="Datatype6_DateTime">
  <xs:restriction base="xs:dateTime">
    <xs:minInclusive value="2002-01-01T00:00:00"/>
  </xs:restriction>
</xs:simpleType>
```

305

306 **2.3.1.4 Enumerated types**307 **2.3.1.4.1 Basic pattern**

- 308
- In the example below, two different types can play role1: either Att1 or Att2.
- 309
- In the ISO XML representation, a ISO XML attribute is introduced to express the
- 310 actual type.

311



312

313

314 **Instance:**

```

315 <Q xmlns="urn:ISO:xsd:$Q" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
316 instance">
317   <role1 xsi:type="H">
318     <Att1>data1</Att1>
319   </role1>
320 </Q>

```

321 **or**

```

322 <Q xmlns="urn:ISO:xsd:$Q" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
323 instance">
324   <role1 xsi:type="I">
325     <Att2>data2</Att2>
326   </role1>
327 </Q>
328

```

329

330 Schema:

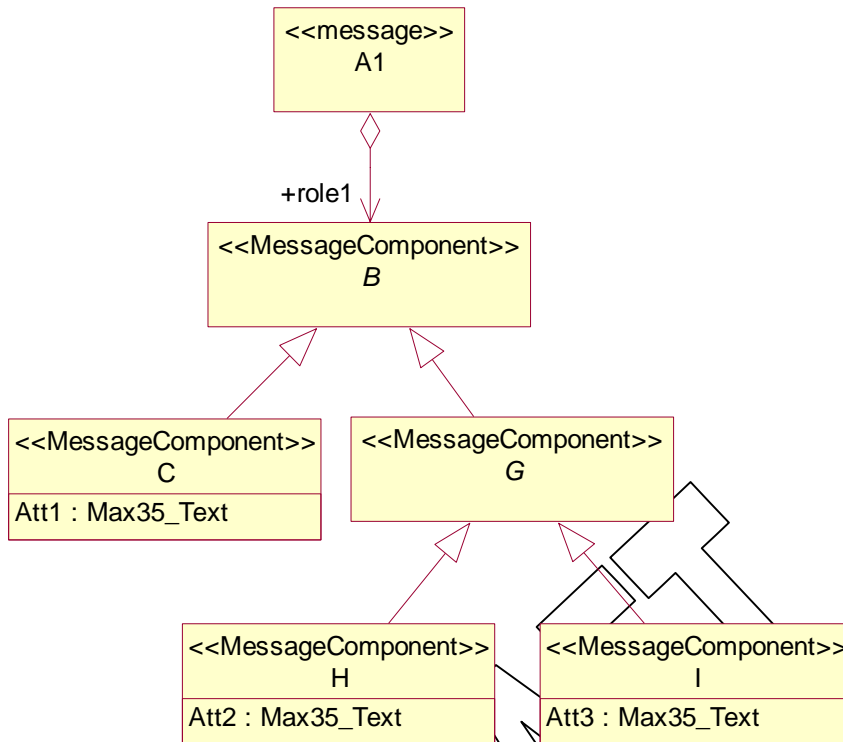
```
331 <?xml version="1.0" encoding="UTF-8"?>
332 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
333 on Sep 07 15:58:10-->
334 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
335 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$Q"
336 targetNamespace="urn:ISO:xsd:$Q">
337
338 <xs:element name="Q" type="Q"/>
339
340 <xs:complexType name="Q">
341 <xs:sequence>
342 <xs:element name="role1" type="J"/>
343 </xs:sequence>
344 </xs:complexType>
345
346 <xs:complexType name="H">
347 <xs:complexContent>
348 <xs:extension base="J">
349 <xs:sequence>
350 <xs:element name="Att1" type="Max35_Text"/>
351 </xs:sequence>
352 </xs:extension>
353 </xs:complexContent>
354 </xs:complexType>
355
356 <xs:complexType name="J" abstract="true"/>
357
358 <xs:complexType name="I">
359 <xs:complexContent>
360 <xs:extension base="J">
361 <xs:sequence>
362 <xs:element name="Att2" type="Max35_Text"/>
363 </xs:sequence>
364 </xs:extension>
365 </xs:complexContent>
366 </xs:complexType>
367
368 <xs:simpleType name="Max35_Text">
369 <xs:restriction base="xs:string">
370 <xs:length value="35"/>
371 </xs:restriction>
372 </xs:simpleType>
373
374 </xs:schema>
375
```

376

377

378 **2.3.1.4.2 Re-use pattern**

379



380

381

382

383

384 Instance:

```

385 <A1 xmlns="urn:ISO:xsd:$A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
386 instance">
387   <role1 xsi:type="C">
388     <Att1>data1</Att1>
389   </role1>
390 </A1>
  
```

391 or

```

392 <A1 xmlns="urn:ISO:xsd:$A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
393 instance">
394   <role1 xsi:type="H">
395     <Att2>data2</Att2>
396   </role1>
397 </A1>
  
```

398 or

```

399 <A1 xmlns="urn:ISO:xsd:$A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
400 instance">
401   <role1 xsi:type="I">
  
```

```
402     <Att3>data3</Att3>
403   </role1>
404 </A1>
405
```

406

407 **Schema:**

```
408 <?xml version="1.0" encoding="UTF-8"?>
409 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
410 on Sep 07 15:58:10-->
411 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
412   elementFormDefault="qualified" xmlns="urn:ISO:xsd:$A1"
413   targetNamespace="urn:ISO:xsd:$A1">
414
415   <xs:element name="A1" type="A1"/>
416
417   <xs:complexType name="A1">
418     <xs:sequence>
419       <xs:element name="role1" type="B"/>
420     </xs:sequence>
421   </xs:complexType>
422
423   <xs:complexType name="G" abstract="true">
424     <xs:complexContent>
425       <xs:extension base="B"/>
426     </xs:complexContent>
427   </xs:complexType>
428
429   <xs:complexType name="B" abstract="true"/>
430
431   <xs:complexType name="C">
432     <xs:complexContent>
433       <xs:extension base="B">
434         <xs:sequence>
435           <xs:element name="Att1" type="Max35_Text"/>
436         </xs:sequence>
437       </xs:extension>
438     </xs:complexContent>
439   </xs:complexType>
440
441   <xs:complexType name="I">
442     <xs:complexContent>
443       <xs:extension base="G">
444         <xs:sequence>
445           <xs:element name="Att3" type="Max35_Text"/>
446         </xs:sequence>
447       </xs:extension>
448     </xs:complexContent>
449   </xs:complexType>
450
451   <xs:complexType name="H">
452     <xs:complexContent>
453       <xs:extension base="G">
454         <xs:sequence>
455           <xs:element name="Att2" type="Max35_Text"/>
```

```
456     </xs:sequence
457     </xs:extension>
458 </xs:complexContent>
459 </xs:complexType>
460
461 <xs:simpleType name="Max35_Text">
462   <xs:restriction base="xs:string">
463     <xs:length value="35"/>
464   </xs:restriction>
465 </xs:simpleType>
466
467 </xs:schema>
468
```

DRAFT

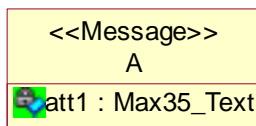
469

470 **2.3.2 Class**

471

UML	XML instance
Class name with a role name	Role becomes an element. The class itself has no corresponding ISO XML element.
Class name without a role name: <ul style="list-style-type: none"> The class is aggregated but the role name is not given; or The class has the stereotype <<message>> 	The class name becomes the ISO XML element name

472



473

474 Instance:

```

475 <A xmlns="urn:ISO:xsd:$A" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
476 instance">
477   <att1>data</att1>
478 </A>
  
```

479

480 Schema:

```

481 <?xml version="1.0" encoding="UTF-8"?>
482 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
483 on Sep 05 16:21:43-->
484 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
485 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$A"
486 targetNamespace="urn:ISO:xsd:$A">
487   <xs:element name="A" type="A"/>
488
489   <xs:complexType name="A">
490     <xs:sequence>
491       <xs:element name="att1" type="Max35_Text"/>
492     </xs:sequence>
493   </xs:complexType>
494
495   <xs:simpleType name="Max35_Text">
496     <xs:restriction base="xs:string">
497       <xs:length value="35"/>
498     </xs:restriction>
499   </xs:simpleType>
  
```


500
501 </xs:schema>

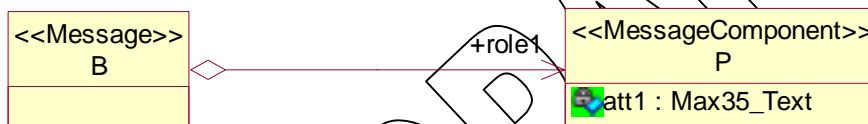
502 2.3.3 Simple composition

- 503 ▪ A parent-child relationship between two classes is expressed by a role;
- 504 ▪ The parent-class maps to a ISO XML element with its name as the tag (see pattern
505 “[class name without a role](#)”);
- 506 ▪ The role of the child-class maps to a ISO XML element tag. The child class is not
507 mapped.

508

UML	ISO XML instance
Parent class	See “ Class ” pattern
Child class	ISO XML element with role name as tag. This element is contained within the parent element

509



510 Instance:

```

511 <B xmlns="urn:ISO:xsd:$B" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
512 instance">
513   <role1>
514     <att1>data</att1>
515   </role1>
516 </B>
  
```

517

518 Schema:

```

519 <?xml version="1.0" encoding="UTF-8"?>
520 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
521 on Sep 05 16:21:43-->
522 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
523 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$B"
524 targetNamespace="urn:ISO:xsd:$B">
525
526
527 <xs:element name="B" type="B"/>
528
529 <xs:complexType name="B">
530   <xs:sequence>
531     <xs:element name="role1" type="P"/>
532   </xs:sequence>
  
```

```

533 </xs:complexType>
534
535 <xs:complexType name="P">
536   <xs:sequence>
537     <xs:element name="att1" type="Max35_Text" />
538   </xs:sequence>
539 </xs:complexType>
540
541 <xs:simpleType name="Max35_Text">
542   <xs:restriction base="xs:string">
543     <xs:length value="35" />
544   </xs:restriction>
545 </xs:simpleType>
546
547 </xs:schema>
548

```

549

550 2.3.4 Class attributes

- 551 ▪ A class can also contain attributes;
- 552 ▪ A class attribute is described using a name and a type;
- 553 ▪ By default ,the first ISO XML child elements within its parents are the attributes,
554 followed by the roles. However, you can define [the sequence of all the child](#)
555 [elements](#) belonging to a class.

UML	ISO XML instance
Parent class	See " Class " pattern
Child class	ISO XML element with role name as tag. This element is contained within the parent element.
Class containing attributes	ISO XML elements with attribute name as tag. This element is contained within the parent element.

556



557

558 Instance:

```

559 <D xmlns="urn:ISO:xsd:$D" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
560 instance">
561   <att1>>false</att1>
562   <role1>
563     <att2>data2</att2>
564   </role1>

```

565 </D>

566

567 Schema:

```
568 <?xml version="1.0" encoding="UTF-8"?>
569 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
570 on Sep 05 16:21:43-->
571 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
572 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$D"
573 targetNamespace="urn:ISO:xsd:$D">
574
575 <xs:element name="D" type="D"/>
576
577 <xs:complexType name="D">
578   <xs:sequence>
579     <xs:element name="att1" type="TrueFalse_Indicator"/>
580     <xs:element name="role1" type="E"/>
581   </xs:sequence>
582 </xs:complexType>
583
584 <xs:complexType name="E">
585   <xs:sequence>
586     <xs:element name="att2" type="Max35_Text"/>
587   </xs:sequence>
588 </xs:complexType>
589
590 <xs:simpleType name="Max35_Text">
591   <xs:restriction base="xs:string">
592     <xs:length value="35"/>
593   </xs:restriction>
594 </xs:simpleType>
595
596 <xs:simpleType name="TrueFalse_Indicator">
597   <xs:restriction base="xs:boolean"/>
598 </xs:simpleType>
599
600 </xs:schema>
```

601

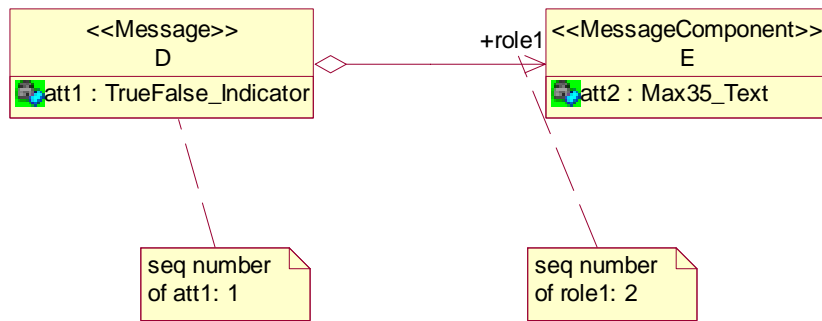
602

603 2.3.4.1 Element order

604

605 To manage the order in which XML elements are generated from a given UML model, each
606 UML attribute and role (automatically or manually) gets assigned a sequence number (see
607 previous schema and instance).

608



609

610

611 2.3.5 Composition of vectorial attributes (Collections)

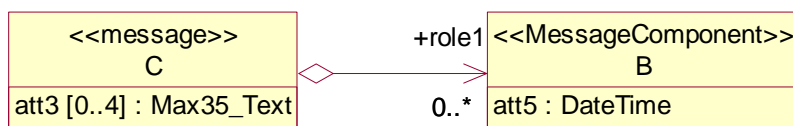
- 612 • The cardinality expresses the number of occurrences of elements. The default value is
613 1, in which case it can be omitted; else it is represented as a **range** e.g **0..***.
- 614 • Use a range-cardinality to express a collection of elements, which can be represented
615 either as a collection of attributes or roles. In the example below, C contains a
616 collection of A's expressed as attributes (att3) and a collection of Bs expressed as roles
617 (role1).
- 618 • Schemas can validate exactly the cardinality.

619

Cardinality	Description	Schema representation
1	Exactly one	Element name="A"
0..1	Optional	Element name="A" minOccurs="0" maxOccurs="1"
0..n	Any number of occurrences	Element name="A" minOccurs="0" maxOccurs="unbounded"
1..n	At least one	Element name="A" minOccurs="1" maxOccurs="unbounded"
1..4	From 1 to 4	Element name="A" minOccurs="1" maxOccurs="4"
0..3	From 0 to 3	Element name="A" minOccurs="0" maxOccurs="3"

620

621



622

623 Instance :

```
624 <C xmlns="urn:ISO:xsd:$C" xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
625 instance">  
626   <att3>data1a</att3>  
627   <att3>data1b</att3>  
628   <role1>  
629     <att5>2001-01-01</att5>  
630   </role1>  
631 </C>
```

632

633 Schema:

```
634 <?xml version="1.0" encoding="UTF-8"?>  
635 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)  
636 on Sep 07 13:40:40-->  
637 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
638 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$C"  
639 targetNamespace="urn:ISO:xsd:$C">  
640  
641 <xs:element name="C" type="C"/>  
642  
643 <xs:complexType name="C">  
644 <xs:sequence>  
645 <xs:element name="att3" type="Max35_Text" minOccurs="0" maxOccurs="4"/>  
646 <xs:element name="role1" type="B" minOccurs="0" maxOccurs="unbounded"/>  
647 </xs:sequence>  
648 </xs:complexType>  
649  
650 <xs:complexType name="B">  
651 <xs:sequence>  
652 <xs:element name="att5" type="xs:dateTime"/>  
653 </xs:sequence>  
654 </xs:complexType>  
655  
656 <xs:simpleType name="Max35_Text">  
657 <xs:restriction base="xs:string">  
658 <xs:length value="35"/>  
659 </xs:restriction>  
660 </xs:simpleType>  
661  
662 </xs:schema>  
663
```

664

665

666

667

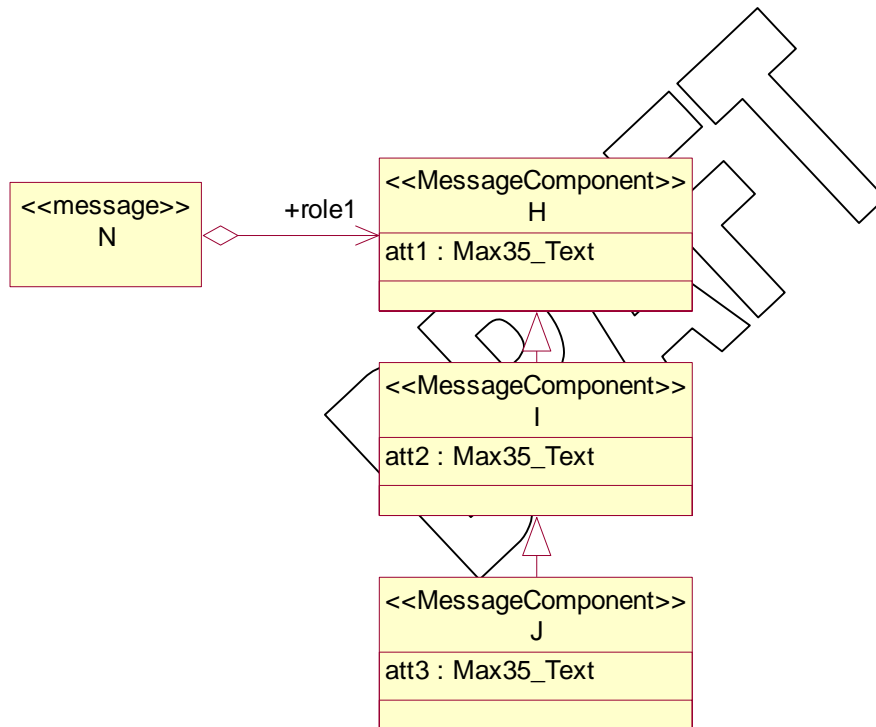
668 **2.3.6 Inheritance**

669 It is possible to re-use business elements by specializing existing elements. This process -
670 also called virtual containment - impacts element order and generated Schemas.

- 671 ▪ In the example below the business element H contains an attribute att1. The
672 business element I, which re-uses H, contains att2 and att1; the latter attribute is
673 inherited from H. The business element J, which re-uses I, contains att3, att2 and
674 att1; the last two attributes being inherited from I respectively H.
- 675 ▪ This means that a container N containing H, can also contain I, as I “is-a” H; etc...
676 This process is

677

678



679

680 **Instance:**

```

681 <N xmlns="urn:ISO:xsd:$N" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
682 instance">
683   <role1 xsi:type="H">
684     <att1>data1</att1>
685   </role1>
686 </N>

```

687 or

```

688 <N xmlns="urn:ISO:xsd:$N" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
689 instance">
690   <role1 xsi:type="I">
691     <att1>data1</att1>
692     <att2>data2</att2>
693   </role1>
694 </N>

```

695 or

```

696 <N xmlns="urn:ISO:xsd:$N" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
697 instance">
698   <role1 xsi:type="J">
699     <att1>data1</att1>
700     <att2>data2</att2>
701     <att3>data3</att3>
702   </role1>
703 </N>

```

704

705 Schema:

```

706 <?xml version="1.0" encoding="UTF-8"?>
707 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
708 on Sep 07 13:40:40-->
709 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
710 elementFormDefault="qualified" xmlns="urn:ISO:xsd:$N"
711 targetNamespace="urn:ISO:xsd:$N">
712
713   <xs:element name="N" type="N"/>
714
715   <xs:complexType name="N">
716     <xs:sequence>
717       <xs:element name="role1" type="H"/>
718     </xs:sequence>
719   </xs:complexType>
720
721   <xs:complexType name="I">
722     <xs:complexContent>
723       <xs:extension base="H">
724         <xs:sequence>
725           <xs:element name="att2" type="Max35_Text"/>
726         </xs:sequence>
727       </xs:extension>
728     </xs:complexContent>
729   </xs:complexType>
730
731   <xs:complexType name="H">
732     <xs:sequence>
733       <xs:element name="att1" type="Max35_Text"/>
734     </xs:sequence>
735   </xs:complexType>
736
737   <xs:complexType name="J">
738     <xs:complexContent>
739       <xs:extension base="I">
740         <xs:sequence>

```

```

741     <xs:element name="att3" type="Max35_Text" />
742   </xs:sequence>
743 </xs:extension>
744 </xs:complexContent>
745 </xs:complexType>
746
747 <xs:simpleType name="Max35_Text">
748   <xs:restriction base="xs:string">
749     <xs:length value="35" />
750   </xs:restriction>
751 </xs:simpleType>
752
753 </xs:schema>

```

754

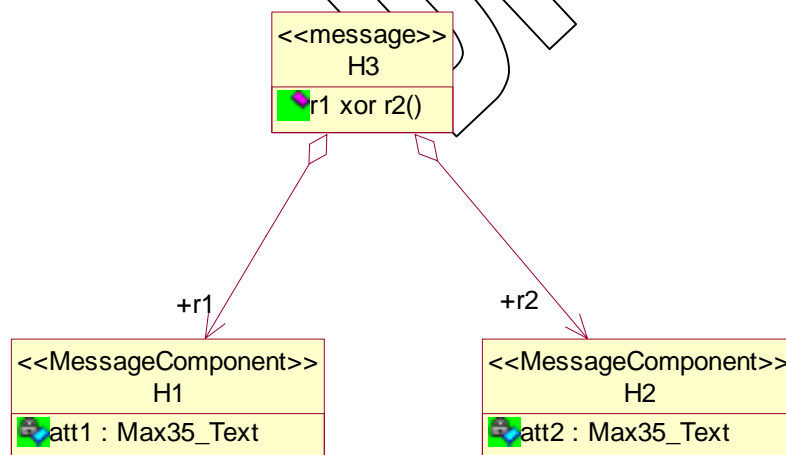
755 Notes:

- 756 ▪ Inherited attributes appear first;
- 757 ▪ Inheritance is cumulative: always add attributes, never remove them;
- 758 ▪ It is an error in the pattern to redefine an attribute that already exists in a base class.
- 759 ▪ XML schemas do not support multiple inheritance.

760

761 **2.3.7 Enumerated roles using XOR invariant**

762



763

764 Instance:

```

765 <H3 xmlns="urn:ISO:xsd:$H3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
766 instance">
767   <r1>
768     <att1>data1</att1>

```


769 </r1>
770 </H3>

771 or

772 <H3 xmlns="urn:ISO:xsd:\$H3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
773 instance">
774 <r2>
775 <att2>data2</att2>
776 </r2>
777 </H3>

778

779 Schema:

780 <?xml version="1.0" encoding="UTF-8"?>
781 <!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
782 on Sep 07 16:55:10-->
783 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
784 elementFormDefault="qualified" xmlns="urn:ISO:xsd:\$H3"
785 targetNamespace="urn:ISO:xsd:\$H3">
786
787 <xs:element name="H3" type="H3"/>
788
789 <xs:complexType name="H3">
790 <xs:sequence>
791 <xs:choice>
792 <xs:element name="r1" type="H1"/>
793 <xs:element name="r2" type="H2"/>
794 </xs:choice>
795 </xs:sequence>
796 </xs:complexType>
797
798 <xs:complexType name="H2">
799 <xs:sequence>
800 <xs:element name="att2" type="Max35_Text"/>
801 </xs:sequence>
802 </xs:complexType>
803
804 <xs:complexType name="H1">
805 <xs:sequence>
806 <xs:element name="att1" type="Max35_Text"/>
807 </xs:sequence>
808 </xs:complexType>
809
810 <xs:simpleType name="Max35_Text">
811 <xs:restriction base="xs:string">
812 <xs:length value="35"/>
813 </xs:restriction>
814 </xs:simpleType>
815
816 </xs:schema>

817

818

819 **Note:** multiplicity for enumerated roles is treated as follows:

820

UML notation	UML notation	Schema notation	means
r1 0..1	r2 0..1	minOccurs="0" maxOccurs="1"	r1 or r2 may be present, but not both. This means both may be absent as well.
r1 0..n	r2 0..n	minOccurs="0" maxOccurs="unbounded"	r1 or r2 may be present up to n times, but not both. This means both may be absent as well.
r1 1	r2 1	-	r1 or r2 must be present, but not both (= XOR).
r1 1..n	r2 1..n	minOccurs="1" maxOccurs="unbounded"	r1 or r2 must be present up to n times, but not both (= XOR).
r1 0..n	r2 1..n	A choice between <xsd:element name= « r1 » with minOccurs="0" maxOccurs="unbounded" and <xsd:element name= « r2 » minOccurs="1" maxOccurs="unbounded"	r1 or r2 may be present up to n times, but not both. This means both may be absent as well.

821

822 **Note:** some rules regarding the XOR in UML:

823

- Any XML name may be given to the operation

824

825

- the XOR operation has to be declared in a specific way in its "operation specification box".

826

827

- It is not allowed to make an XOR between a role of the current class and a role of a sub- or superclass.

828

829

830

- The XOR invariant only applies to the roles mentioned in the XOR. Consequently, some roles may not be part of the XOR. Hence when roles are added, they are not part of the XOR until they are also added in the XOR invariant.

831 **2.3.8 Enumerated attributes using XOR invariant**

832

833

```
<S xmlns="urn:ISO:xsd:$S" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

<<Message>> S
att1 : Max35_Text att2 : Max35_Text
<<inv>> att1 xor att2()

834

instance">

835

<att1>data1</att1>

836

</S>

837

or

838

```
<S xmlns="urn:ISO:xsd:$S" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

839

instance">

840

<att2>data2</att2>

841

</S>

842

843

Schema:

844

```
<?xml version="1.0" encoding="UTF-8"?>
```

845

```
<!--Schema version 2.2 - Generated by Swift workstation (build:R2.2.0.10)
```

846

```
on Sep 07 16:55:10-->
```

847

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

848

```
elementFormDefault="qualified" xmlns="urn:ISO:xsd:$S"
```

849

```
targetNamespace="urn:ISO:xsd:$S">
```

850

851

```
<xs:element name="S" type="S"/>
```

852

853

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

854

```
<xs:sequence>
```

855

```
<xs:choice>
```

856

```
<xs:element name="att1" type="Max35_Text"/>
```

857

```
<xs:element name="att2" type="Max35_Text"/>
```

858

```
</xs:choice>
```

859

```
</xs:sequence>
```

860

```
</xs:complexType>
```

861

862

```
<xs:simpleType name="Max35_Text">
```

863

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

864

```
<xs:length value="35"/>
```

865

```
</xs:restriction>
```

866

```
</xs:simpleType>
```

867

868

```
</xs:schema>
```

869

870

871

872 **Note:** some rules regarding the XOR in UML:

- 873 • Any valid XML name may be given to the operation
- 874 • the XOR operation has to be declared in a specific way in its “operation
875 specification box”.
- 876 • It is not allowed to make an XOR between an attribute of the current class and an
877 attribute of a sub- or superclass.
- 878 • The XOR invariant only applies to the attributes mentioned in the XOR.
879 Consequently, some attributes within the class may not be part of the XOR. Hence
880 when attributes are added to the class, they are not part of the XOR until they are
881 also added in the XOR invariant.

882

883 **2.3.9 Enumerated roles and attributes using <<choice>> stereotype**

884 This pattern models a choice between roles and/or attributes.

885 All roles between the superclass containing the <<choice>> stereotype and its subclasses
886 are part of the choice, as well as all attributes in the superclass. Consequently, when a role /
887 attribute is added, it becomes automatically part of the choice (as opposed to the XOR
888 invariant pattern where a new role / attribute does not automatically become part of the
889 choice). When a role / attribute is removed, it is automatically removed from the choice.

890

891

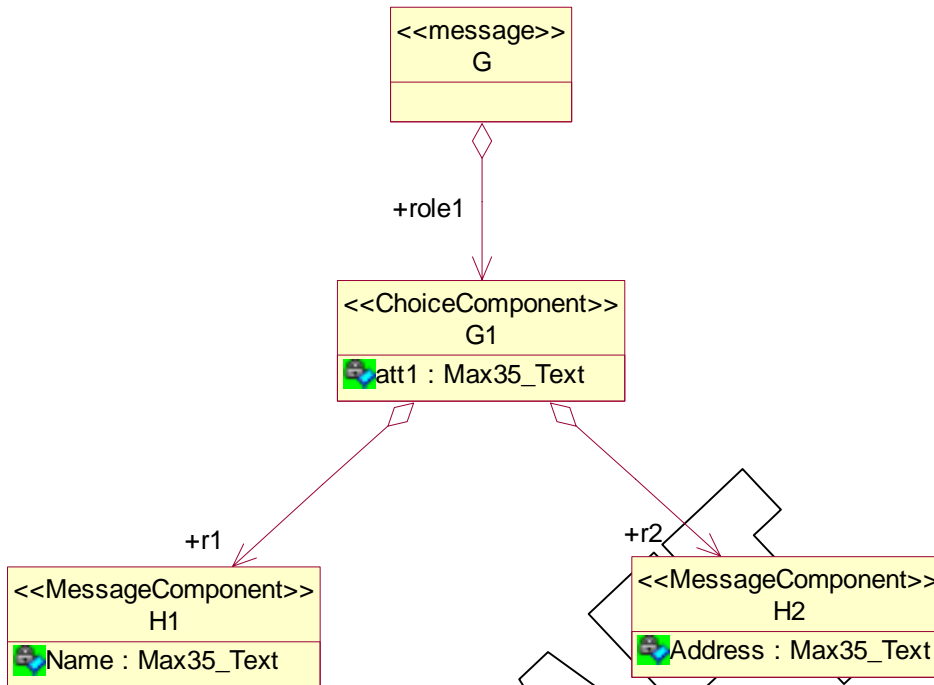
892

893

894

895

896



897 Instance:

```

898 G xmlns="urn:ISO:xsd:$G" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
899 instance">
900   <role1>
901     <att1>data</att1>
902   </role1>
903 </G>

```

904 or

```

905 <G xmlns="urn:ISO:xsd:$G" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
906 instance">
907   <role1>
908     <r1>
909       <Name>data</Name>
910     </r1>
911   </role1>
912 </G>

```

913 or

```

914 <G xmlns="urn:ISO:xsd:$G" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
915 instance">
916   <role1>
917     <r2>
918       <Address>data</Address>
919     </r2>
920   </role1>
921 </G>

```

922

923 Schema:

```

924 <?xml version="1.0" encoding="UTF-8"?>
925 <!--Schema version 2.2 - Generated by SWIFTStandards Workstation
926 (build:R2.2.0.10) on Oct 18 18:40:07-->
927 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
928 elementFormDefault="qualified" xmlns="urn:swift:xsd:$G"
929 targetNamespace="urn:swift:xsd:$G">
930
931 <xs:element name="G" type="G"/>
932
933 <xs:complexType name="G">
934 <xs:sequence>
935 <xs:element name="role1" type="G1"/>
936 </xs:sequence>
937 </xs:complexType>
938
939 <xs:complexType name="G1">
940 <xs:sequence>
941 <xs:choice>
942 <xs:element name="att1" type="Max35_Text"/>
943 <xs:element name="r1" type="H1"/>
944 <xs:element name="r2" type="H2"/>
945 </xs:choice>
946 </xs:sequence>
947 </xs:complexType>
948
949 <xs:complexType name="H2">
950 <xs:sequence>
951 <xs:element name="Address" type="Max35_Text"/>
952 </xs:sequence>
953 </xs:complexType>
954
955 <xs:complexType name="H1">
956 <xs:sequence>
957 <xs:element name="Name" type="Max35_Text"/>
958 </xs:sequence>
959 </xs:complexType>
960
961 <xs:simpleType name="Max35_Text">
962 <xs:restriction base="xs:string">
963 <xs:length value="35"/>
964 </xs:restriction>
965 </xs:simpleType>
966 </xs:schema>

```

967

968

969 **Note:** the aggregation of a <<choice>> may not have a multiplicity. However the members
970 of a <<choice>> are allowed to have one. These multiplicities are treated as follows:

971

UML notation	UML notation	Schema notation	means

r1 0..1	r2 0..1	minOccurs="0" maxOccurs="1"	r1 or r2 may be present, but not both. This means both may be absent as well.
r1 0..n	r2 0..n	minOccurs="0" maxOccurs="unbounded"	r1 or r2 may be present up to n times, but not both. This means both may be absent as well.
r1 1	r2 1	-	r1 or r2 must be present, but not both (= XOR).
r1 1..n	r2 1..n	minOccurs="1" maxOccurs="unbounded"	r1 or r2 must be present up to n times, but not both (= XOR).
r1 0..n	r2 1..n	A choice between <xsd:element name= « r1 » with minOccurs="0" maxOccurs="unbounded" and <xsd:element name= « r2 » minOccurs="1" maxOccurs="unbounded"	r1 or r2 may be present up to n times, but not both. This means both may be absent as well.

972

973

974 **3 Schema design rules**975 **3.1 Common design rules and usage**

- 976 • Should only be used to validate the message (though this validation is limited if we
977 compare with pure software validation)
- 978 • Should not replace the UML model.
- 979

980 **3.2 Schema Design rules**981 **3.2.1 XML name clash support within the scope of a message**982 **3.2.1.1 General behaviour of ISO XML attributes**

983 The schema will be generated only for validation purposes.

984 **3.2.1.2 Case 1: 2 UML role names are the same and have the same content
985 model**

986 This is not an issue, as those role names will be defined in two different complexTypes.

987 **3.2.1.3 Case 2: 2 UML role or 2 attribute names are the same, and they have
988 a different content model**

989 This is not an issue for schemas as long as the roles or attributes belong to different classes.

990 **3.2.1.4 Case 3: 2 UML attribute names are the same, and their respective
991 UML types are the same.**

992 Same as 3.2.1.2

993 **3.2.1.5 Case 4: A UML role name and a UML attribute name are the same**

994 This is not an issue for schemas as long as the role and attribute belong to different classes.

995 3.2.1.6 Case 5: Two classes in different packages have the same name

996 As the name of the class will be used for naming the associated complexType in the
997 schema, this is NOT allowed.

998 3.2.2 XML schema features used in ISO XML

999 3.2.2.1 Namespaces in XML schema and XML instances

1000 ISO XML schema and XML instances use four name spaces:

- 1001 ▪ the default (non qualified) namespace. All schema have their own default
1002 namespace generated according to the following regular expression:
1003 “urn:ISO:xsd:\\$+”. Where the “+” must be replaced by the message name
1004 eventually prefixed by the collaboration name separated by a ‘.’.
- 1005 ▪ xs: W3C XML schema namespace (not used in instances)
- 1006 ▪ xsi: W3C XML schema-instance namespace
- 1007 ▪ a target namespace (for schema only) which is the same as the default namespace.

1008 Schema:

```
1009 <schema  
1010     xmlns="urn:ISO:xsd:$NoticeOfExecution"  
1011     xmlns:xs=" http://www.w3.org/2001/XMLSchema"  
1012     targetNamespace="urn:ISO:xsd:$NoticeOfExecution">
```

1013 Instance:

```
1014 <NoticeOfExecution  
1015     xmlns="urn:ISO:xsd:$NoticeOfExecution"  
1016     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

1017 3.2.2.2 Schema location in the XML instance

1018 The rootelement of the business payload carries the location (as an Universal Resource
1019 Information) of the XML Schema, in the form of the XML attribute
1020 `xsi:SchemaLocation`.

1021 It is not allowed to have `xsi:SchemaLocation` to appear in any element (much like
1022 `xmlns`) but only in the rootelement of the business payload.

1023 Instance:

```
1024 <NoticeOfExecution xsi:schemaLocation="file://file_path">
```

1025 3.2.2.3 XML facets on simple types

1026 The following sections describe the facets that will be introduced in the XML schema.

1027 Those XML schema facets are based on the UML invariants which have been specified in
1028 the UML model according to the section [3.4. Summary of UML invariants related to](#)
1029 [schema production](#).

1030 3.2.2.3.1 pattern

1031 Pattern matching allows lexical validation on strings, which syntax can be described using
1032 regular expressions, (commonly referred to as “[Perl](#) expressions”).

1033 This facet only applies to strings.

1034 The exact syntax of the allowed regular expressions is defined in appendix E of “XML
1035 Schema Part 2: Datatypes” (XML Schema’s W3C Recommendation May 2001).

1036 For instance:

```
1037 <xs:simpleType name='BIC'>  
1038   <xs:restriction base='string'>  
1039     <xs:pattern value='[a-zA-Z]{4,4}[a-zA-Z]{2,2}[a-zA-Z0-9]{2,2}[a-zA-Z0-9]{0,3}'/>  
1040   </xs:restriction>  
1041 </xs:simpleType>
```

1042

1043 3.2.2.3.2 length, minLength, maxLength

1044 XML schema allows restriction of the value space of any string value (i.e.: double, integer,
1045 date etc are not affected) by using the following constraining facets:

- 1046 ▪ length
- 1047 ▪ minLength
- 1048 ▪ maxLength

1049 Those facets only apply on strings, and their values must be positive integer values.

1050 For instance, a BankAddress is a string of 10 characters minimum and 40 characters
1051 maximum:

```
1052 <xs:simpleType name='BankAddress'>  
1053   <xs:restriction base='string'>  
1054     <xs:minLength value='10'/>  
1055     <xs:maxLength value='40'/>  
1056   </xs:restriction>  
1057 </xs:simpleType>
```

1058

1059 3.2.2.3.3 minInclusive, maxInclusive, minExclusive, maxExclusive

1060 XML schema allows restriction of the value space of any numerical value by using the
1061 following constraining facets:

- 1062 ▪ minInclusive
- 1063 ▪ minExclusive
- 1064 ▪ maxInclusive
- 1065 ▪ maxExclusive

1066 Those facets only apply to numerical values (Integer, Long, BigDecimal, Float, Double)
1067 and to time measurement related values (Date, Time,...) and their value must be constants
1068 of the same type than the numeric value they apply to.

1069 For instance, the financial instrument below must contain between 1 and 100000 securities:

```

1070 <xs:complexType name='SecuritiesInstrument'>
1071   <xs:sequence>
1072     <xs:element name='ISIN' type='string'/>
1073     <xs:element name='Quantity' >
1074       <xs:simpleType>
1075         <xs:restriction base='xs:decimal'>
1076           <xs:minInclusive value='1'/>
1077           <xs:maxInclusive value='100000'/>
1078         </xs:restriction>
1079       </xs:simpleType>
1080     </xs:element>
1081   </xs:sequence>
1082 </xs:complexType>
1083

```

1084 3.2.2.3.4 enumeration

1085 XML schema allows restriction of the value space of an enumeration by using the
1086 enumeration constraining facet.

1087 This facet only applies to enumerations, and their value must be part of the original
1088 enumeration from which they restrict.

1089 For instance, a class M containing an attribute b of type E1 with an XML Invariant
1090 restricting the enumerated value to Value2:

```

1091 <xs:complexType name="M">
1092   <xs:sequence>
1093     <xs:element name = "b">
1094       <xs:simpleType>
1095         <xs:restriction base="E1">

```

```
1096         <xs:enumeration value = "Value2"/>
1097     </xs:restriction>
1098 </xs:simpleType>
1099 </xs:element>
1100 </xs:sequence>
1101 </xs:complexType>
1102 <xs:simpleType name = "E1">
1103     <xs:restriction base = "xs:string">
1104         <xs:enumeration value = "Value1"/>
1105         <xs:enumeration value = "Value2"/>
1106     </xs:restriction>
1107 </xs:simpleType>
1108
```

1109 3.2.2.3.5 totalDigits, fractionDigits

1110 Fixed point decimal values need a totalDigits specification (i.e. the maximum number of
1111 decimal digits in values of datatypes derived from decimal: totalDigits), as well as a
1112 fractionDigits specification (i.e. the maximum number of decimal digits in the fractional
1113 part of values of datatypes derived from decimal: fractionDigits).

1114 The value of the totalDigits facet must be a positive integer.

1115 The value of the fractionDigits facet must be a non-negative integer.

1116 For instance, requiring a totalDigits of 8 digits with 2 digits after the decimal point on an
1117 amount would translate to the following instance:

```
1118 <xs:simpleType name='Amount'>
1119     <xs:restriction base='xs:decimal'>
1120         <xs:totalDigits value='8'>
1121             <xs:fractionDigits value='2'>
1122                 </xs:restriction>
1123     </xs:simpleType>
```

1124 3.2.2.4 Nillable

1125 To be used in conjunction with the XML-nil attribute. The Schema attribute `nillable`
1126 specifies whether the instance can carry a nil value. Default value is `false`.

1127 In the following schema:

```
1128     <xs:complexType name=' FinancialInstrument'>
1129         <xs:sequence>
1130             <xs:element name='ISIN' type='string' />
1131             <xs:element name='Quantity' type='xs:decimal' nillable='true' />
1132         </xs:sequence>
1133     </xs:complexType >
```

1134 Only the Quantity can carry a nil value.

1135

1136 It should be noted that nillable is not a facet, but an attribute (as abstract, minOccurs,
1137 maxOccurs, ...). This implies that, in the schema's context, nillable applies to an element
1138 (and not a type).

1139 Therefore the nillable option should consequently not be encoded as an invariant on a class
1140 in the UML model. It will thus be set either at the attribute or role level (in which case the
1141 corresponding element in the schema would be nillable).

1142 In the XML instance document, the XML attribute **nil** can be used to indicate that an
1143 element has no value.

1144 Assuming the following schema:

```
1145     <xs:complexType name='OrderOfBuy'>
1146         <xs:element Securities type='FinancialInstrument' />
1147     </xs:complexType >
1148     <xs:complexType name=' FinancialInstrument'>
1149         <xs:element name='ISIN' type='string' />
1150         <xs:element name='Quantity' type='xs:decimal' nillable='true' />
1151     </xs:complexType >
```

1152 An order-of-buy XML instance with no quantity of securities specified (as opposed to a
1153 value of zero) will be expressed as:

```
1154     <OrderOfSell>
1155         <Securities>
1156             <ISIN>BE1234567890 </ISIN>
1157             <Quantity xsi:nil='true' />
1158         </Securities>
1159     </ OrderOfSell >
```

1160 Note that an alternative to not using the 'nil' XML-attribute is to omit the nil element. By
1161 doing so we introduce an ambiguity between **not** specifying an optional element and
1162 specifying an optional element which value is **nil**.

1163 3.3 Granularity of Schemas

1164 There is one Schema per message.

1165 **3.4 Summary of UML invariants related to schema production**

1166 Those invariants will be defined as user properties on methods having the <<inv>>
1167 stereotypes, on the tab called XML Invariants.

1168

XML facet	Applies on UML type	Value of type	Schema example
pattern	String	Defined in Appendix E of “XML Schema Part 2: Datatypes”	<pre><xs:simpleType name='BIC'> <xs:restriction base='string'> <xs:pattern value='[a-z]{2,4}' /> </xs:restriction> </xs:simpleType></pre>
length	String	Non-negative integer	<pre><xs:simpleType name='BIC'> <xs:restriction base='string'> <xs:length value='12' /> </xs:restriction> </xs:simpleType></pre>
minLength	String	Non-negative integer	<pre><xs:simpleType name='BIC'> <xs:restriction base='string'> <xs:minLength value='8' /> </xs:restriction> </xs:simpleType></pre>
maxLength	String	Non-negative integer	<pre><xs:simpleType name='BIC'> <xs:restriction base='string'> <xs:maxLength value='12' /> </xs:restriction> </xs:simpleType></pre>
totalDigits	Integer, Long, Float, Double, BigDecimal	Positive integer	<pre><xs:simpleType name='BEF'> <xs:restriction base='xs:decimal'> <xs:totalDigits value='3' /> </xs:restriction> </xs:simpleType></pre>

fractionDigits	Float, Double, BigDecimal	Non-negative integer	<xs:simpleType name='USD' > <xs:restriction base='xs:decimal' > <xs:fractionDigits value='2'/> </xs:restriction> </xs:simpleType>
minInclusive	Integer, Long, Float, Double, BigDecimal	Constant of the same type as the UML type	<xs:simpleType name='Salary' > <xs:restriction base='xs:decimal' > <xs:minInclusive value='40000'/> </xs:restriction> </xs:simpleType>
minExclusive	Integer, Long, Float, Double, BigDecimal	Constant of the same type as the UML type	<xs:simpleType name='Salary' > <xs:restriction base='xs:decimal' > <xs:minExclusive value='40000'/> </xs:restriction> </xs:simpleType>
maxInclusive	Integer, Long, Float, Double, BigDecimal	Constant of the same type as the UML type	<xs:simpleType name='Taxes' > <xs:restriction base='xs:decimal' > <xs:maxInclusive value='90000'/> </xs:restriction> </xs:simpleType>
maxExclusive	Integer, Long, Float, Double, BigDecimal	Constant of the same type as the UML type	<xs:simpleType name='Taxes' > <xs:restriction base='xs:decimal' > <xs:maxExclusive value='90000'/> </xs:restriction> </xs:simpleType>

1169 4 Character set

1170 ISO XML uses UTF-8 as the (default) character encoding mechanism, for the following
1171 reasons:

- 1172 • It has the most efficient method of character representation:
 - 1173 • It is the shortest method to represent the characters which are currently the most
 - 1174 commonly used in a financial environment (ASCII and EBCDIC characters)
 - 1175 • It can still represent almost any known character
- 1176 • It is interoperable with many other encoding schemes through (automatable) conversion
1177 algorithms.

1178 Example:

1179 <?xml version="1.0" encoding="UTF-8"?>

DRAFT

1180

1181 **End of document**

1182

1183

DRAFT