TECHNICAL SPECIFICATION

ISO/TS 20022-4

First edition
2004-12-15

Financial services — UNIversal Financial Industry message scheme —

Part 4: ISO 20022 XML design rules

Services financiers — Schéma universel de messages pour l'industrie financière —

Partie 4: Règles conceptuelles ISO 20022 XML

Reference number
ISO/TS 20022-4:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

(Blank page)

Contents

Foreword

1 Introduction 4

2 Normative references 4

3 ISO 20022 conversion from UML to XML 5

3.1 Assumptions 5
3.2 Terminology and conventions 5
3.2.1 Modelling terminology and conventions 5
3.2.2 XML naming conventions 6
3.3 UML to XML schema and XML instance conversion rules 7
3.3.1 Relationship between XML and UML artefacts 7
3.3.2 Data Types 7
3.3.3 Conversion rules for UML patterns 20

4 Adopted XML schema features 36

4.1 Assumptions 36
4.2 Features 36
4.2.1 Namespaces in XML schema and XML instances 36
4.2.2 XML facets on simpleTypes 37
4.3 Granularity of schemas 40
4.4 Summary of UML operations using <<Format>> related to schema production 40
4.5 Character set 41

Annex A: XML abbreviations 43

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/TS 20022-4 was prepared by Technical Committee ISO/TC 68 to complement ISO 20022-1, Overall methodology and format specifications for inputs to and outputs from the ISO 20022 Repository, with the XML syntax design rules to be applied by the ISO 20022 Registration Authority to translate an ISO 20022 compliant message definition into an ISO 20022 XML message schema for the production of ISO 20022 XML message instances. This Technical Specification should be reviewed and considered for publication as an International Standard once further experience has been gained in using these guidelines and the use of the underlying technology has further stabilized.

ISO 20022 consists of the following parts, under the general title *Financial services — UNIversal Financial Industry message scheme*:

— *Part 1: Overall methodology and format specifications for inputs to and outputs from the ISO 20022 Repository*

— *Part 2: Roles and responsibilities of the registration bodies*

— *Part 3: ISO 20022 modelling guidelines* [Technical Specification]

— *Part 4: ISO 20022 XML design rules* [Technical Specification]

— *Part 5: ISO 20022 reverse engineering* [Technical Specification]

1 Introduction

XML is a technical standard defined by W3C (the World Wide Web Consortium) that can be used for the physical representation (i.e. the syntax) of standardized ISO 20022 Messages. XML leaves a lot of freedom for the exact way it is used in a particular application. Therefore, merely stating that XML is used is not sufficient to guarantee predictability; one must also explain HOW it will be used.

This Technical Specification contains a set of XML design rules, called ISO 20022 XML. These design rules define how a standardized Message – described by a Message Definition in UML[1] according to the Modelling Guidelines of ISO/TS 20022-3 must be represented as a valid ISO 20022 compliant XML document.

A **valid XML document** (referred to hereafter as 'XML instance' or 'instance') as defined by W3C is any XML document that has an associated description and that complies with the constraints expressed in that description. The associated description in this case is derived from the Message Definition, which is originally described in UML.

This Technical Specification also describes how (a part of) the UML Message Definition can be converted into a W3C XML Schema. This XML schema will then make it possible to use a validating XML schema parser to automatically verify that a given XML instance complies with (a subset of) the constraints described in the Message Definition.

DTDs (Document Type Definitions) could also be used to validate partial compliance of an XML instance to its corresponding Message Definition. However, because of the limited validation functionality DTDs offer, this document does NOT cover XML DTDs.

Note that this document merely explains how a given Message Definition Diagram will be mapped into XML. It doesn't explain how to create a Message Definition Diagram. This information can be found in ISO/TS 20022-3 Modelling guidelines.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 20022-1, *Financial services — UNIversal Financial Industry message scheme — Part 1: Overall methodology and format specifications for inputs to and outputs from the ISO 20022 Repository*

URN namespace for ISO documents.

[1] More information about UML (Unified Modelling Language) is available on the Object Management Group website at: http://www.omg.org/uml

3 ISO 20022 conversion from UML to XML

3.1 Assumptions

Design rules to convert a Message Definition defined in UML into ISO 20022 XML are governed by the following design choices:

- The current work is based on following W3C Recommendations: the XML specification of October 2000 (http://www.w3c.org/TR/2000/REC-xml-20001006) and W3C's XML Schema specification of May, 2001 (http://www.w3c.org/TR/xmlschema-0/), (http://www.w3c.org/TR/xmlschema-1/) and (http://www.w3c.org/TR/xmlschema-2/)
- ISO 20022 XML representation must be as systematic as possible:
  - Business information is expressed as XML elements/values;
  - Metadata information is expressed as XML attributes. XML attributes are not to be conveyed 'on the wire' in the XML instance, unless required to remove ambiguity.
- Each ISO 20022 XML element, attribute, simpleType or complexType has a corresponding UML model element.
- Currently only ISO 20022 XML runtime (=validation) schemas are generated. Runtime schemas only contain information required to validate XML instances. No documentation or implementation information (e.g. elementID, version, definition, etc.) is mentioned.

3.2 Terminology and conventions

3.2.1 Modelling terminology and conventions[2]

- A **Message Definition** is represented in UML by a hierarchical Class Diagram (the **Message Definition Diagram**).
- The root class of the **Message Definition Diagram** is a UML class without attributes, containing the stereotype <<Message>>. It represents a **Message**.
- A **Message** is composed of **Message Components.** UML aggregations (with their UML role) are used to represent the parent-child relationship between the class representing the **Message** and its composing **Message Components.**
- A **Message Component** is represented as a UML class with an appropriate stereotype (<<MessageComponent>> or <<ChoiceComponent>>). A **Message Component**

[2] See also "ISO/TS 20022-3: ISO 20022 modelling guidelines" for more details on the modelling of a Message Definition.

contains **Message Elements**. A **Message Element** is either represented as a UML class attribute (where the class represents a **Message Component)** or as the UML role of a UML aggregation between two UML classes (where each class represents a **Message Component)**.

- Each **Message Element** has a type. This type is either represented as the UML type of a UML class attribute (where the attribute represents the **Message Element**) or by the target class of a UML aggregation (where the aggregation carries the role that represents the **Message Element**).
- The type of a **Message Element** is either a **Message Component** or a **Data Type**. A **Data Type** is represented as a UML class with an appropriate stereotype (such as <<Code>>, <<Identifier>> or <<Text>>[3]). A **Data Type** may have metadata, which is then represented as a stereotyped attribute in the UML class that represents the **Data Type**.

3.2.2 XML naming conventions

All names that are used for XML elements, XML attributes, XML simpleTypes and XML complexTypes are based on the names of the corresponding UML artefacts:

- XML simpleTypes and XML complexTypes use directly the names of their corresponding UML classes.
- For reasons of optimization[4], XML elements and XML attributes that may appear in XML instances use an abbreviated version of the names of the corresponding UML artefacts. The abbreviation is based on a ma.3pping table. Since this mapping table will be continuously updated with new abbreviations, it is put on http://www.iso20022.org.
- The Message is given a **Message Identifier**, defined by the ISO 20022 Registration Authority. The Message Identifier uniquely identifies the message and has the following structure: "**xxxx.nnn.aaa.bb**", whereby
  - **xxxx** is an alphabetic code in four positions (fixed length) identifying the Business Process
  - **nnn** is an alphanumeric code in three positions (fixed length) identifying the Message Functionality
  - **aaa** is a numeric code in three positions (fixed length) identifying a particular flavour (variant) of Message Functionality
  - **bb** is a numeric code in two positions (fixed length) identifying the version
  - '.' character as delimiter between elements.

[3] See section 3.3.2 Data Types for more information on Data Types.

[4] In Annex there is a list of guidelines to be followed to optimize Messages. These guidelines aim at improving both the validation performance as well as the throughput.

3.3 UML to XML schema and XML instance conversion rules

A Message Definition is composed of a limited number of distinct UML patterns.

By defining the conversion rules from those patterns to ISO 20022 XML, we can convert any Message Definition Diagram into its corresponding ISO 20022 XML schema and any Message into its corresponding ISO 20022 XML instance.

3.3.1 Relationship between XML and UML artefacts

An ISO 20022 **XML element** can represent the following UML artefacts:

- a Message
- a Message Element

An ISO 20022 **XML attribute** can represent the following UML artefacts:

- metadata of a Data Type

An ISO 20022 **XML simpleType** or an ISO 20022 **XML complexType with simpleContent** can represent the following UML artefacts:

- a Data Type

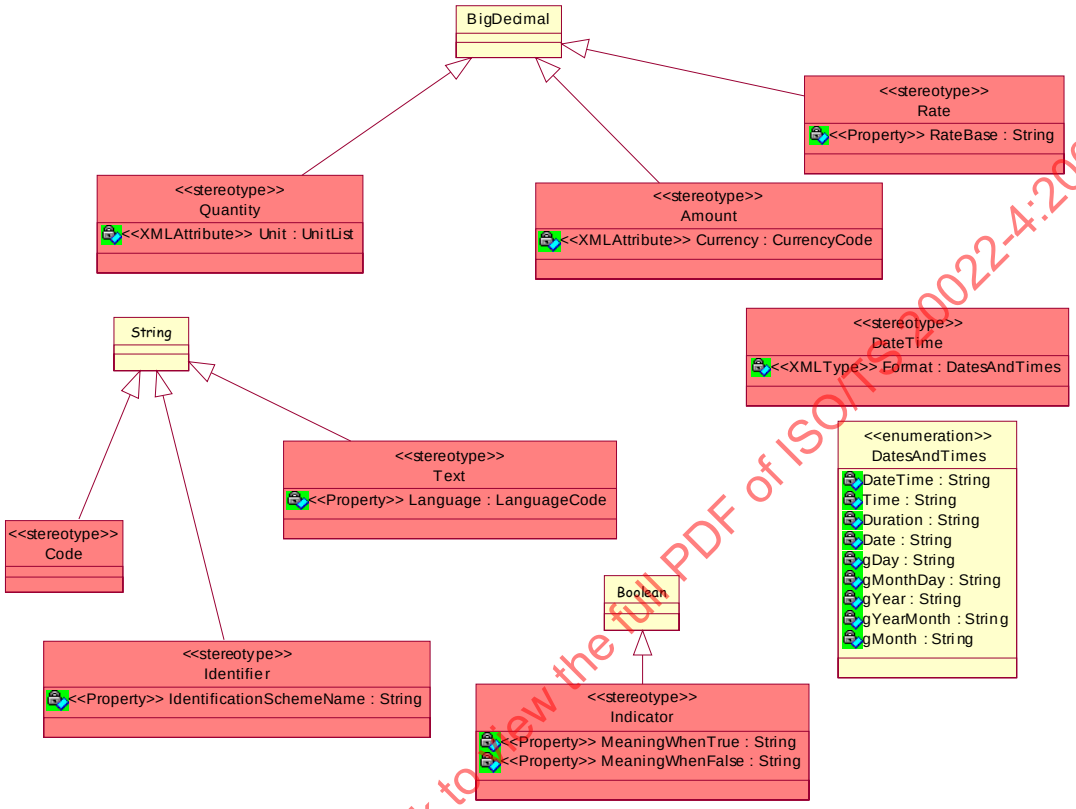
An ISO 20022 **XML complexType** can represent the following UML artefacts:

- a Message
- a Message Component

3.3.2 Data Types

All Message Elements that are used in a Message Definition have a type. This type may be an ISO 20022 **Data Type**. ISO 20022 Data Types are always based on a Data Type Representation.

3.3.2.1 Data Type Representations



Notes:

Each ISO 20022 Data Type is represented as a UML class and is stereotyped by one of the allowed Data Type Representations. A Data Type Representation has a number of characteristics that are passed on to ('inherited by') all Data Types that are using that Data Type Representation. In this way, characteristics common to a number of Data Types are grouped together.

Some characteristics are represented as UML class attributes (stereotyped appropriately as <<XMLAttribute>>, <<Property>> or <<XMLType>> to allow the correct conversion into XML schema and XML instance) in the class that represents the ISO 20022 Data Type.

Most information that is carried by these attributes is static information that only needs to be used for the documentation and/or implementation of the ISO 20022 Data Type. In some cases however the attribute will have an impact on the definition of the corresponding simpleType in the XML schema.

Some information that is carried by these attributes may be dynamic information that needs to be included in each XML instance that uses this ISO 20022 Data Type.

Chapter 2.3.2.3 gives a detailed description for ISO 20022 Data Types per allowed Data Type Representation. This description includes the impact of the attributes on the XML schema and/or XML instance.

3.3.2.2 Primitive Data Types

ISO 20022 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:

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 for further details. Format CCYY-MM-DD
Time	time	Corresponds to a time. See ISO 8601 for further details. Format HH:MM:SS +- offset to UTC
DateTime	dateTime	Corresponds to a date and time. See ISO 8601 for further details. Format CCYY-MM-DDTHH:MM:SS +- offset to UTC
Duration	duration	Corresponds to a period in time. See ISO 8601 for further details. Format PnYnMnDTnHnMnS
gDay	gDay	Corresponds to a set of one-day long, monthly periodic instances. The time zone must be UTC. See ISO 8601 for further details. Format: ---DD.
gMonth	gMonth	Corresponds to 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. See ISO 8601 for further details. Format: --MM--.
gYear	gYear	Corresponds to 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. See ISO 8601 for further details. Format: CCYY

gMonthday	gMonthday	Corresponds to a set of one-day long, annually periodic instances. The time zone must be UTC. See ISO 8601 for further details. Format:--MM-DD.
base64Binary	base64Binary	represents Base64-encoded arbitrary binary data

3.3.2.3 ISO 20022 Data Types

It is possible to define ISO 20022 Data Types by using one of the Data Type Representations.

By doing so, all characteristics of that Data Type representation are used by that ISO 20022 Data Type (the primitive type, any properties, etc).

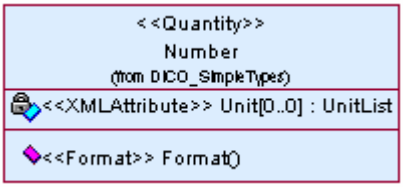
The value space of the original primitive type (e.g. String) and of the Data Type Representation (e.g. Identifier) is constrained by specifying the actual value of the relevant characteristics (e.g. the Identification Scheme to be used). It can be further constrained by introducing UML operations stereotyped by <<Format>>. Those operations will be converted to facets when generating XML schemas.

In order to apply facets, the XML types that are generated for those Data Types must be simpleTypes or complexTypes with simpleContent, and not complexTypes[5]. This is no problem as ISO 20022 Data Types map to an XML simpleType or complexType with simpleContent, which on their turn restrict an XML primitive type

The following sections describe the detailed conversion rules for Data Types of all allowed Data Type Representations.

[5] XML schema validation constraint: Facets cannot be applied to complexTypes without simpleContent.

3.3.2.3.1 Data Type using Data Type Representation <<Quantity>>



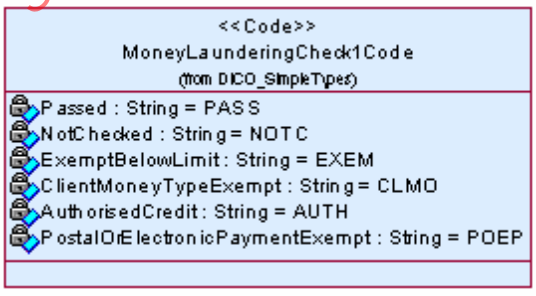
Properties:

The Data Type Representation Quantity (see metamodel) has an attribute called Unit (stereotyped as <<XMLAttribute>>). Any Data Type that is stereotyped by <<Quantity>> must specify whether there is a list of possible values for Unit.

- In case there is a list containing more than one value, the attribute will appear in the XML schema as an XML attribute and each instance will contain the actually used value. The list of valid values is defined in the Data Type "UnitList".
- In case there is only one or no value this fact will only be documented and the attribute will not appear in the XML schema or in the instance. The unit is then implied (either because it is fixed and hence documented or because it is present somewhere else in the Message).

3.3.2.3.2 Data Type using Data Type Representation <<Code>>

UML	ISO 20022 XML instance
Class contains an enumeration of possible values	ISO 20022 XML element contains the chosen value



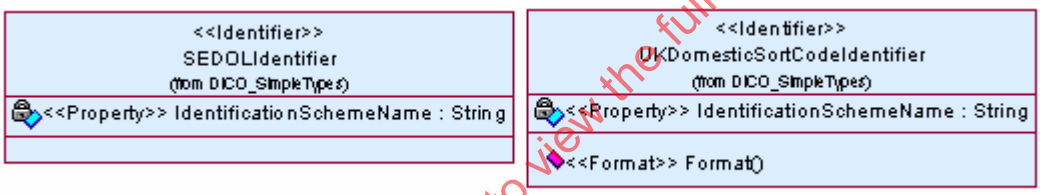
Properties:

This Data Type is used when the values of the list have a meaningful (i.e. semantic) value within the context of the message (e.g. the trade types). ISO 20022 Data Types using <<Code>> reference an internal list (i.e. a list specified in the schema). It is an enumeration of which one of the enumerated values has to be chosen in the instance.

An enumerated value is constrained within a list of possible values.

The values for the enumerated items are taken from the four-character initial value given to each of the UML enumerated attributes.

3.3.2.3.3 Data Type using Data Type Representation <<Identifier>>



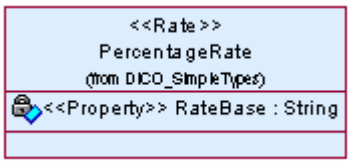
Properties:

ISO 20022 Data Types using <<Identifier>> refer to an external list (i.e. not specified in the schema).

The Data Type Representation Identifier (see metamodel) has an attribute called IdentificationSchemeName (stereotyped as <<Property>>). Any Data Type that is stereotyped by <<Identifier>> must specify the name of the actual identification scheme. This information only serves documentation purposes and will not be part of the XML schema or instance.

If required, a facet may be added for syntactical checking (using operation <<Format>>).

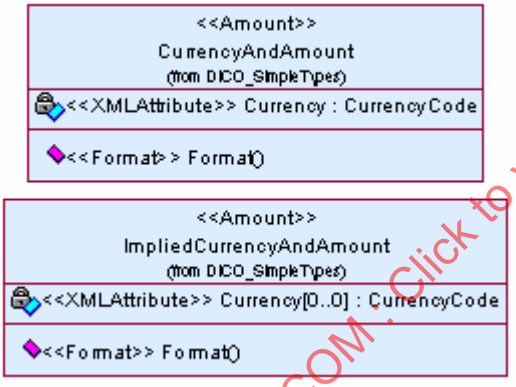
3.3.2.3.4 Data Type using Data Type Representation <<Rate>>



Properties:

The Data Type Representation Rate (see metamodel) has an attribute called RateBase (stereotyped as <<Property>>). Any Data Type that is stereotyped by <<Rate>> must specify the base that is actually used. This information only serves documentation purposes and will not be part of the XML schema or instance.

3.3.2.3.5 Data Type using Data Type Representation <<Amount>>

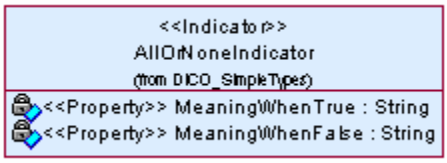


Properties:

The Data Type Representation Amount (see metamodel) has an attribute called Currency (stereotyped as <<XMLAttribute>>). Any Data Type that is stereotyped by <<Amount>> must specify whether there is a list of possible values for Currency.

- In case there is a list containing more than one value, the attribute will appear in the XML schema as an XML attribute and each instance will contain the actually used value. The list of valid values is defined in the Data Type "CurrencyCode".
- In case there is only one or no value this fact will only be documented and the attribute will not appear in the XML schema or in the instance. The currency is then implied (either because it is fixed and hence documented or because it is present somewhere else in the Message).

3.3.2.3.6 Data Type using Data Type Representation <<Indicator>>

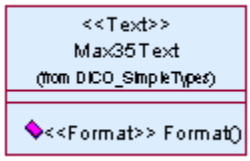


Properties:

A Data Type stereotyped by Data Type Representation <<Indicator>> indicates that the Message Element must have a Boolean value (true or false).

The Data Type Representation Indicator (see metamodel) has two attributes called MeaningWhenTrue and MeaningWhenFalse (both stereotyped as <<Property>>). Any Data Type that is stereotyped by <<Indicator>> must specify the actual meaning that is implied when the value is true and when the value is false. This information only serves documentation purposes and will not be part of the XML schema or instance.

3.3.2.3.7 Data Type using Data Type Representation <<Text>>

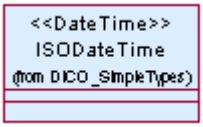


Properties:

A Data Type stereotyped by Data Type Representation <<Text>> indicates that the Message Element contains textual information.

The "Format" operation contains the XML facet for this data type.

3.3.2.3.8 Data Type using Data Type Representation <<DateTime>>



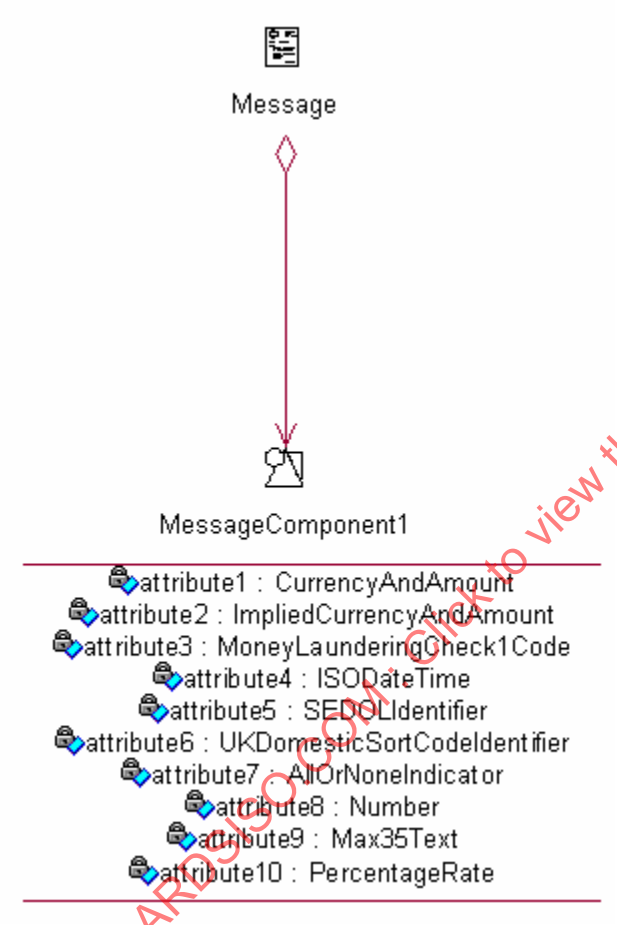
Properties:

The Data Type Representation DateTime (see metamodel) has an attribute called Format (stereotyped as <<XMLType>>). Any Data Type that is stereotyped by <<DateTime>> must specify the actual primitive Data Type (date, dateTime, …) being used. This information will appear in the XML schema as the restriction base for this ISO 20022 Data Type.

3.3.2.4 Example

Following example contains all of the above specified data types in one message. First the Message Definition Diagram is shown, from which the XML schema is derived.

3.3.2.4.1 Message Definition Diagram



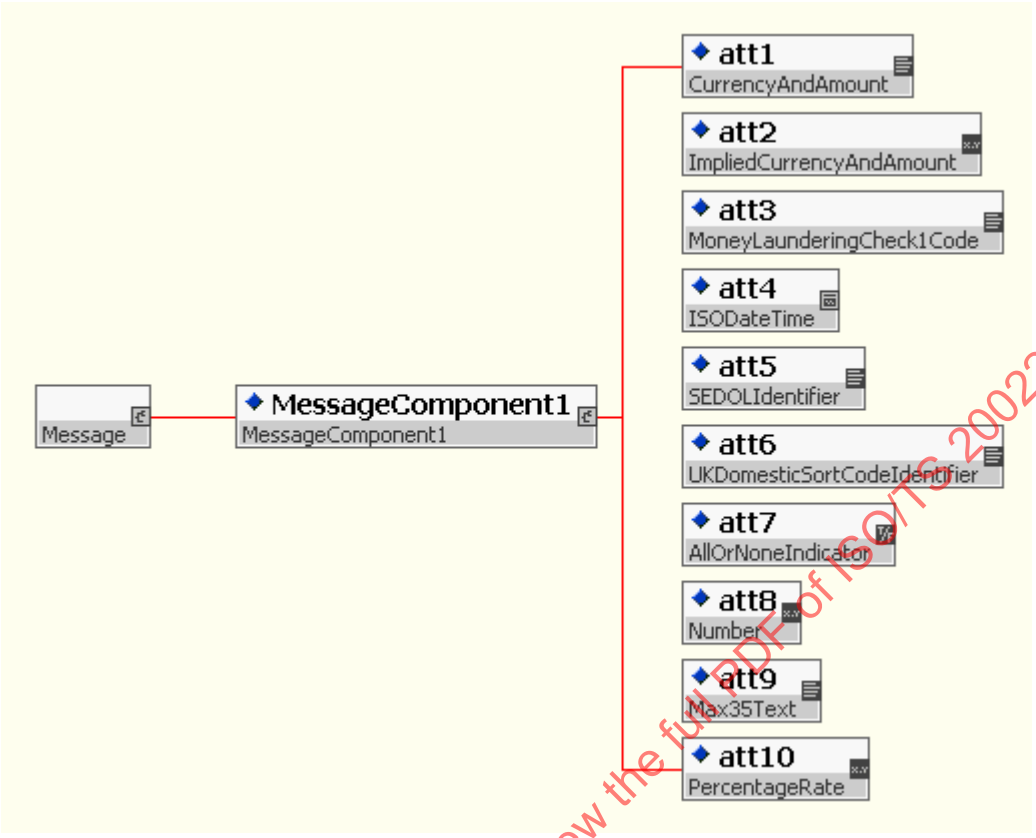
3.3.2.4.2 Schema source[6]

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns = "urn:iso:std:iso:20022:xsd:$Message"
      targetNamespace = "urn:iso:std:iso:20022:xsd:$Message"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified">
      <xs:element name = "Document" type = "Document"/>
      <xs:complexType name = "Document">
            <xs:sequence>
                  <xs:element name = "Message" type = "Message"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Message">
            <xs:sequence>
                  <xs:element name = "MessageComponent1" type = "MessageComponent1"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "MessageComponent1">
            <xs:sequence>
                  <xs:element name = "att1" type = "CurrencyAndAmount"/>
                  <xs:element name = "att2" type = "ImpliedCurrencyAndAmount"/>
                  <xs:element name = "att3" type = "MoneyLaunderingCheck1Code"/>
                  <xs:element name = "att4" type = "ISODateTime"/>
                  <xs:element name = "att5" type = "SEDOLIdentifier"/>
                  <xs:element name = "att6" type = "UKDomesticSortCodeIdentifier"/>
                  <xs:element name = "att7" type = "AllOrNoneIndicator"/>
                  <xs:element name = "att8" type = "Number"/>
                  <xs:element name = "att9" type = "Max35Text"/>
                  <xs:element name = "att10" type = "PercentageRate"/>
            </xs:sequence>
      </xs:complexType>
      <xs:simpleType name = "PercentageRate">
            <xs:restriction base = "xs:decimal"/>
      </xs:simpleType>
      <xs:simpleType name = "Max35Text">
            <xs:restriction base = "xs:string">
                  <xs:maxLength value = "35"/>
                  <xs:minLength value = "1"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name = "Number">
            <xs:restriction base = "xs:decimal">
                  <xs:totalDigits value = "18"/>
                  <xs:fractionDigits value = "0"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name = "AllOrNoneIndicator">
            <xs:restriction base = "xs:boolean"/>
      </xs:simpleType>
```


[6] XML naming conventions (namely abbreviations) cannot be seen directly in the Message Definition Diagram. See also Annex A for more information on abbreviations

```
    <xs:simpleType name = "UKDomesticSortCodeIdentifier">
        <xs:restriction base = "xs:string">
            <xs:pattern value = "SC[0-9]{6,6}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name = "SEDOLIdentifier">
        <xs:restriction base = "xs:string"/>
    </xs:simpleType>
    <xs:simpleType name = "ISODateTime">
        <xs:restriction base = "xs:dateTime"/>
    </xs:simpleType>
    <xs:simpleType name = "MoneyLaunderingCheck1Code">
        <xs:restriction base = "xs:string">
            <xs:enumeration value = "PASS"/>
            <xs:enumeration value = "NOTC"/>
            <xs:enumeration value = "EXEM"/>
            <xs:enumeration value = "CLMO"/>
            <xs:enumeration value = "AUTH"/>
            <xs:enumeration value = "POEP"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name = "ImpliedCurrencyAndAmount">
        <xs:restriction base = "xs:decimal">
            <xs:minInclusive value = "0"/>
            <xs:totalDigits value = "18"/>
            <xs:fractionDigits value = "5"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name = "CurrencyAndAmount_SimpleType">
        <xs:restriction base = "xs:decimal">
            <xs:minInclusive value = "0"/>
            <xs:totalDigits value = "18"/>
            <xs:fractionDigits value = "5"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name = "CurrencyCode">
        <xs:restriction base = "xs:string"/>
    </xs:simpleType>
    <xs:complexType name = "CurrencyAndAmount">
        <xs:simpleContent>
            <xs:extension base = "CurrencyAndAmount_SimpleType">
                <xs:attribute name = "Ccy" use = "required" type = "CurrencyCode"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

3.3.2.5 Schema graphical representation



3.3.2.5.1 XML instance

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Doc:Document xmlns:Doc = "urn:iso:std:iso:20022:xsd:$Message" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
    <Doc:Message>
        <Doc:MessageComponent1>
            <Doc:att1 Ccy = "EUR">100000</Doc:att1>
            <Doc:att2>500000</Doc:att2>
            <Doc:att3>PASS</Doc:att3>
            <Doc:att4>2002-07-21T08:35:30</Doc:att4>
            <Doc:att5>5719210</Doc:att5>
            <Doc:att6>SC123456</Doc:att6>
            <Doc:att7>true</Doc:att7>
            <Doc:att8>12345678901234567</Doc:att8>
            <Doc:att9>ABCDEFGHIJKLMNOPQRST123456789012345</Doc:att9>
            <Doc:att10>35</Doc:att10>
        </Doc:MessageComponent1>
    </Doc:Message>
</Doc:Document>
```

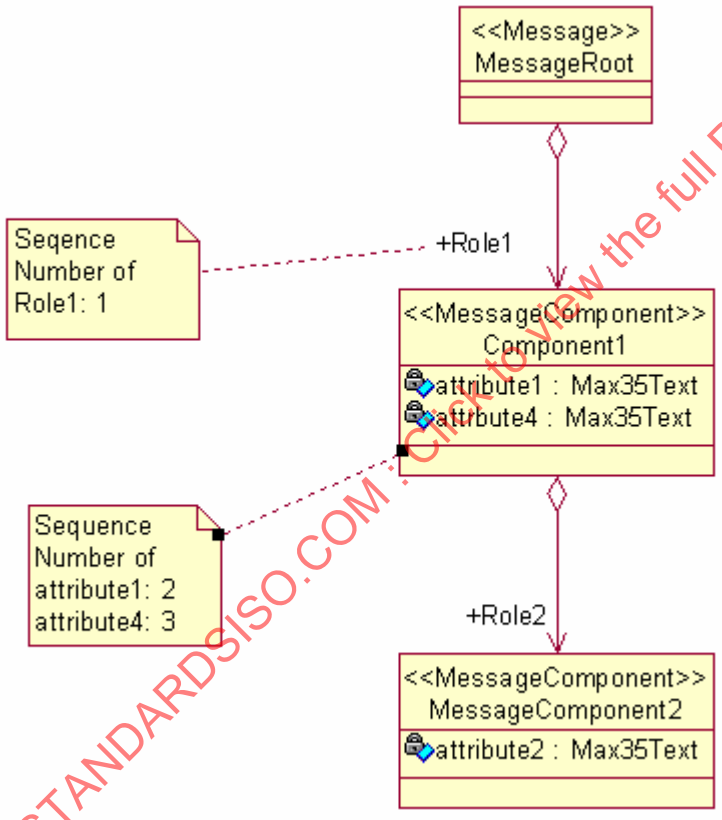
3.3.3 Conversion rules for UML patterns

The following sections describe the detailed conversion rules for all UML patterns that can appear in a Message Definition.

3.3.3.1 Element order

By default in ISO 20022 XML, the order of XML elements is determined as follows: Message Elements represented as class attributes come first, followed by Message Elements represented as aggregation roles. However, this can be overridden.

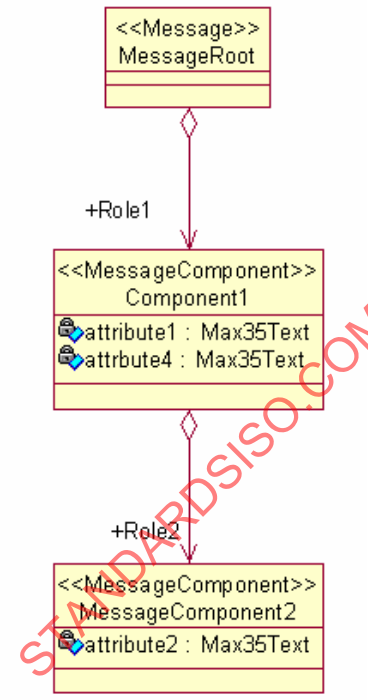
To manage the order in which XML elements are generated from a given Message Definition Diagram, each Message Element gets assigned a sequence number.



3.3.3.2 Simple composition

Message Definition Diagram artefact	XML schema	XML instance
Message Component	complexType.	
Message[7]	complexType. The name becomes the XML root element that is typed by that complexType	XML Root tag
Message Element[8] of a Message Component	XML elements of a complexType.	XML tag

3.3.3.2.1 Message Definition Diagram



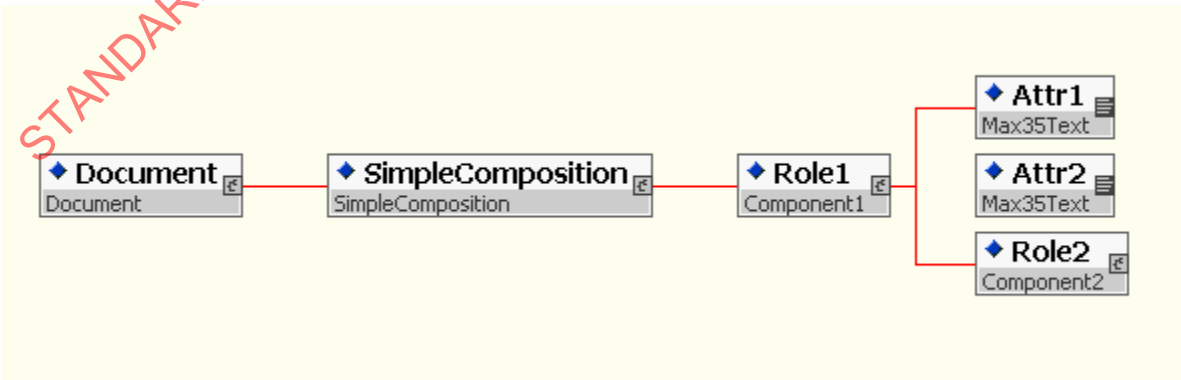
[7] There is only one such element in a message definition diagram, and it is the root element of that message.

[8] Message Elements are either UML Aggregation Roles or UML Class Attributes.

3.3.3.2.2 Schema source

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns = "urn:iso:std:iso:20022:xsd:$SimpleComposition"
      targetNamespace = "urn:iso:std:iso:20022:xsd:$SimpleComposition"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified">
     <xs:element name = "Document" type = "Document"/>
     <xs:complexType name = "Document">
          <xs:sequence>
               <xs:element name = "SimpleComposition" type = "SimpleComposition"/>
          </xs:sequence>
     </xs:complexType>
     <xs:complexType name = "SimpleComposition">
          <xs:sequence>
               <xs:element name = "Role1" type = "Component1"/>
          </xs:sequence>
     </xs:complexType>
     <xs:complexType name = "Component1">
          <xs:sequence>
               <xs:element name = "Attr1" type = "Max35Text"/>
               <xs:element name = "Attr2" type = "Max35Text"/>
               <xs:element name = "Role2" type = "Component2"/>
          </xs:sequence>
     </xs:complexType>
     <xs:complexType name = "Component2">
          <xs:sequence>
               <xs:element name = "Attr1" type = "Max35Text"/>
          </xs:sequence>
     </xs:complexType>
     <xs:simpleType name = "Max35Text">
          <xs:restriction base = "xs:string">
               <xs:maxLength value = "35"/>
               <xs:minLength value = "1"/>
          </xs:restriction>
     </xs:simpleType>
</xs:schema>
```

3.3.3.2.3 Schema graphical representation



3.3.3.2.4 XML instance

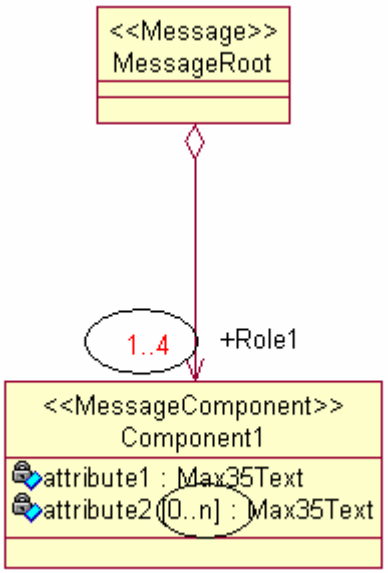
```
<?xml version = "1.0" encoding = "UTF-8"?>
<Doc:Document xmlns:Doc = "urn:iso:std:iso:20022:xsd:$SimpleComposition" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
    <Doc:SimpleComposition>
        <Doc:Role1>
            <Doc:Attr1>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr1>
            <Doc:Attr2>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr2>
            <Doc:Role2>
                <Doc:Attr1>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr1>
            </Doc:Role2>
        </Doc:Role1>
    </Doc:SimpleComposition>
</Doc:Document>
```

3.3.3.3 Composition of vectorial attributes (Collections)

- The multiplicity expresses the number of occurrences of elements. The default value is 1, in which case it can be omitted; else it is represented as a **range** e.g. **`0..n`** .
- Use a range-multiplicity to express a collection of elements, which can represent a collection of Message Elements (expressed as attributes or as roles).
- Schemas can validate exactly the multiplicity.

Multiplicity	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”

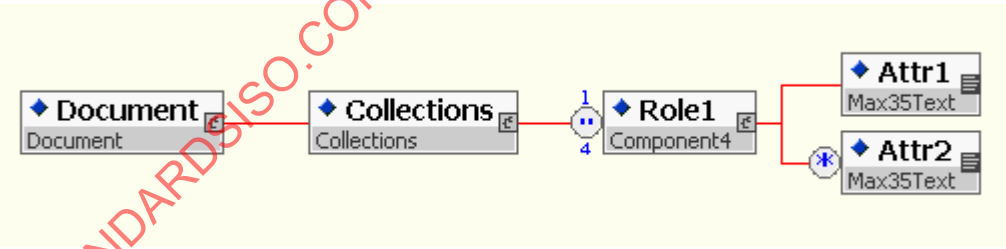
3.3.3.3.1 Message Definition Diagram



3.3.3.3.2 Schema source

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns = "urn:iso:std:iso:20022:xsd:$Collections"
      targetNamespace = "urn:iso:std:iso:20022:xsd:$Collections"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified">
      <xs:element name = "Document" type = "Document"/>
      <xs:complexType name = "Document">
            <xs:sequence>
                  <xs:element name = "Collections" type = "Collections"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Collections">
            <xs:sequence>
                  <xs:element name = "Role1" type = "Component4" maxOccurs = "4"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component4">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
                  <xs:element name = "Attr2" type = "Max35Text" minOccurs = "0" maxOccurs = "unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:simpleType name = "Max35Text">
            <xs:restriction base = "xs:string">
                  <xs:maxLength value = "35"/>
                  <xs:minLength value = "1"/>
            </xs:restriction>
      </xs:simpleType>
</xs:schema>
```

3.3.3.3.3 Schema graphical representation



3.3.3.3.4 XML instance

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Doc:Document xmlns:Doc = "urn:iso:std:iso:20022:xsd:$Collections" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
    <Doc:Collections>
        <Doc:Role1>
            <Doc:Attr1>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr1>
            <Doc:Attr2>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr2>
        </Doc:Role1>
    </Doc:Collections>
</Doc:Document>
```

3.3.3.4 Inheritance

It is possible –but rare- that Message Components are specialized. However, only the most specialized Message Component may be mentioned in the Message Definition Diagram. The generalization component(s) are not mentioned in the schema.

Therefore there is no need to support a specific pattern for inheritance, since it behaves –from a UML to XML point of view- like a simple composition pattern.

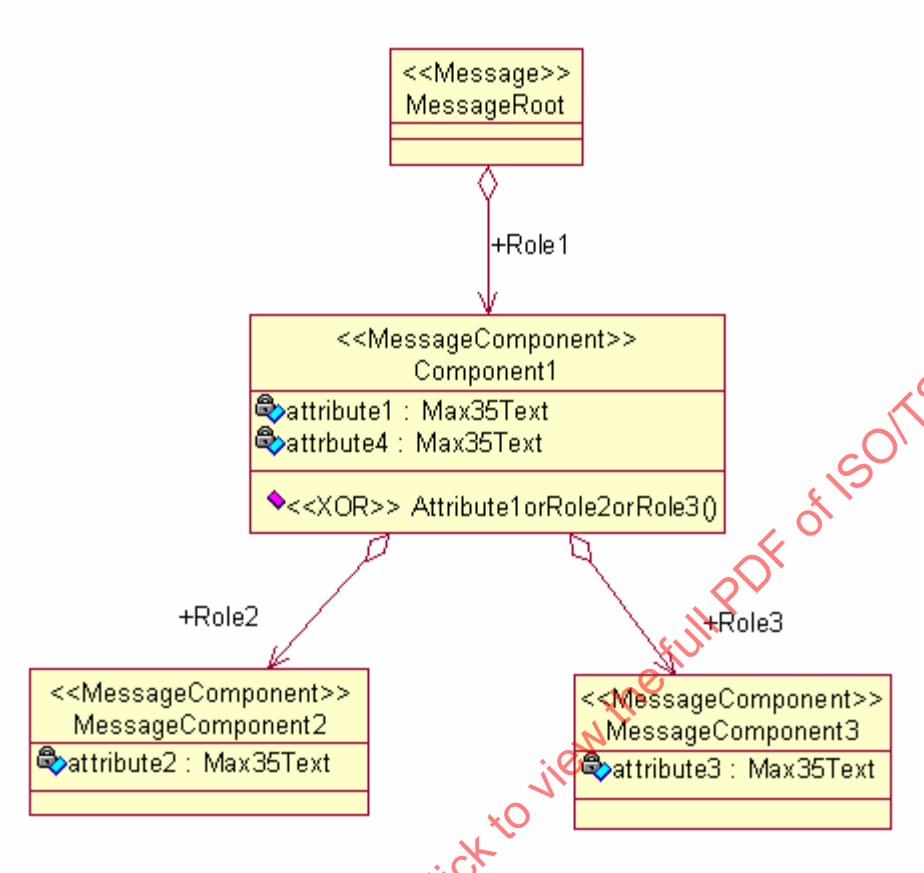
3.3.3.5 Choice between Message Elements using the XOR stereotype

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.

Note: some rules regarding the XOR in UML:

- Any name may be given to the operation
- An XOR must only apply to the Message Elements (i.e. class attributes and aggregation roles) pertaining to that Message Component. It is not allowed to make an XOR between a role of the current Message Component and a role of a child- or parent- Message Component.
- The XOR operation only applies to the Message Elements mentioned in the XOR. Consequently, some Message Elements may not be part of the XOR. Hence when Message Elements are added, they are not part of the XOR until they are also added in the XOR operation.

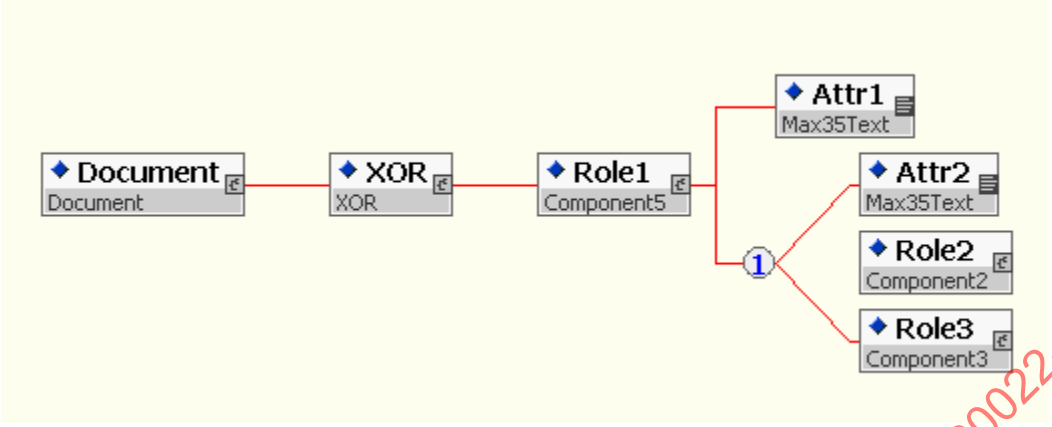
3.3.3.5.1 Message Definition Diagram



3.3.3.5.2 Schema source

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns = "urn:iso:std:iso:20022:xsd:$XOR"
      targetNamespace = "urn:iso:std:iso:20022:xsd:$XOR"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified">
      <xs:element name = "Document" type = "Document"/>
      <xs:complexType name = "Document">
            <xs:sequence>
                  <xs:element name = "XOR" type = "XOR"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "XOR">
            <xs:sequence>
                  <xs:element name = "Role1" type = "Component5"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component5">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
                  <xs:choice>
                        <xs:element name = "Attr2" type = "Max35Text"/>
                        <xs:element name = "Role2" type = "Component2"/>
                        <xs:element name = "Role3" type = "Component3"/>
                  </xs:choice>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component3">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component2">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
            </xs:sequence>
      </xs:complexType>
      <xs:simpleType name = "Max35Text">
            <xs:restriction base = "xs:string">
                  <xs:maxLength value = "35"/>
                  <xs:minLength value = "1"/>
            </xs:restriction>
      </xs:simpleType>
</xs:schema>
```

3.3.3.5.3 Schema graphical representation



3.3.3.5.4 XML instance

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Doc:Document xmlns:Doc = "urn:iso:std:iso:20022:xsd:$XOR" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
    <Doc:XOR>
        <Doc:Role1>
            <Doc:Attr1>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr1>
            <Doc:Attr2>ABCDEFGHIJKLMNOPQRST12345678901 2345</Doc:Attr2>
        </Doc:Role1>
    </Doc:XOR>
</Doc:Document>
```

3.3.3.6 Choice between Message Elements using <<choice>> stereotype

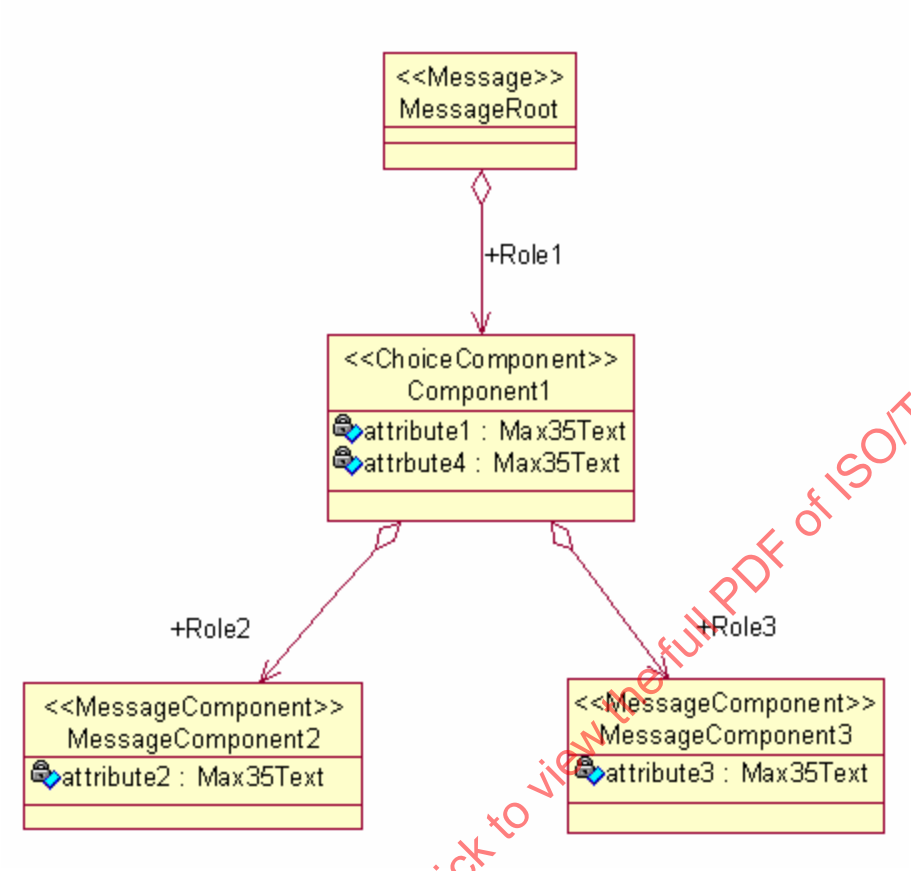
This pattern models a choice between Message Elements (i.e. UML class attributes and UML aggregation roles).

All Message Elements of the ChoiceComponent are part of the choice. Consequently, when a Message Element is added, it becomes automatically part of the choice (as opposed to the XOR pattern where a new Message Element does not automatically become part of the choice). When a Message Element is removed, it is automatically removed from the choice.

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

Msg. Def. Diagram notation	**Msg. Def. Diagram 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.

3.3.3.6.1 Message Definition Diagram



3.3.3.6.2 Schema source

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns = "urn:iso:std:iso:20022:xsd:$Choice"
      targetNamespace = "urn:iso:std:iso:20022:xsd:$Choice"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified">

      <xs:element name = "Document" type = "Document"/>
      <xs:complexType name = "Document">
            <xs:sequence>
                  <xs:element name = "Choice" type = "Choice"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Choice">
            <xs:sequence>
                  <xs:element name = "Role1" type = "Component6"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component6">
            <xs:sequence>
                  <xs:choice>
                        <xs:element name = "Attr1" type = "Max35Text"/>
                        <xs:element name = "Attr2" type = "Max35Text"/>
                        <xs:element name = "Role2" type = "Component2"/>
                        <xs:element name = "Role3" type = "Component3"/>
                  </xs:choice>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component3">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name = "Component2">
            <xs:sequence>
                  <xs:element name = "Attr1" type = "Max35Text"/>
            </xs:sequence>
      </xs:complexType>
      <xs:simpleType name = "Max35Text">
            <xs:restriction base = "xs:string">
                  <xs:maxLength value = "35"/>
                  <xs:minLength value = "1"/>
            </xs:restriction>
      </xs:simpleType>
</xs:schema>
```