Part 6 – UN/CEFACT Naming and Design Rules

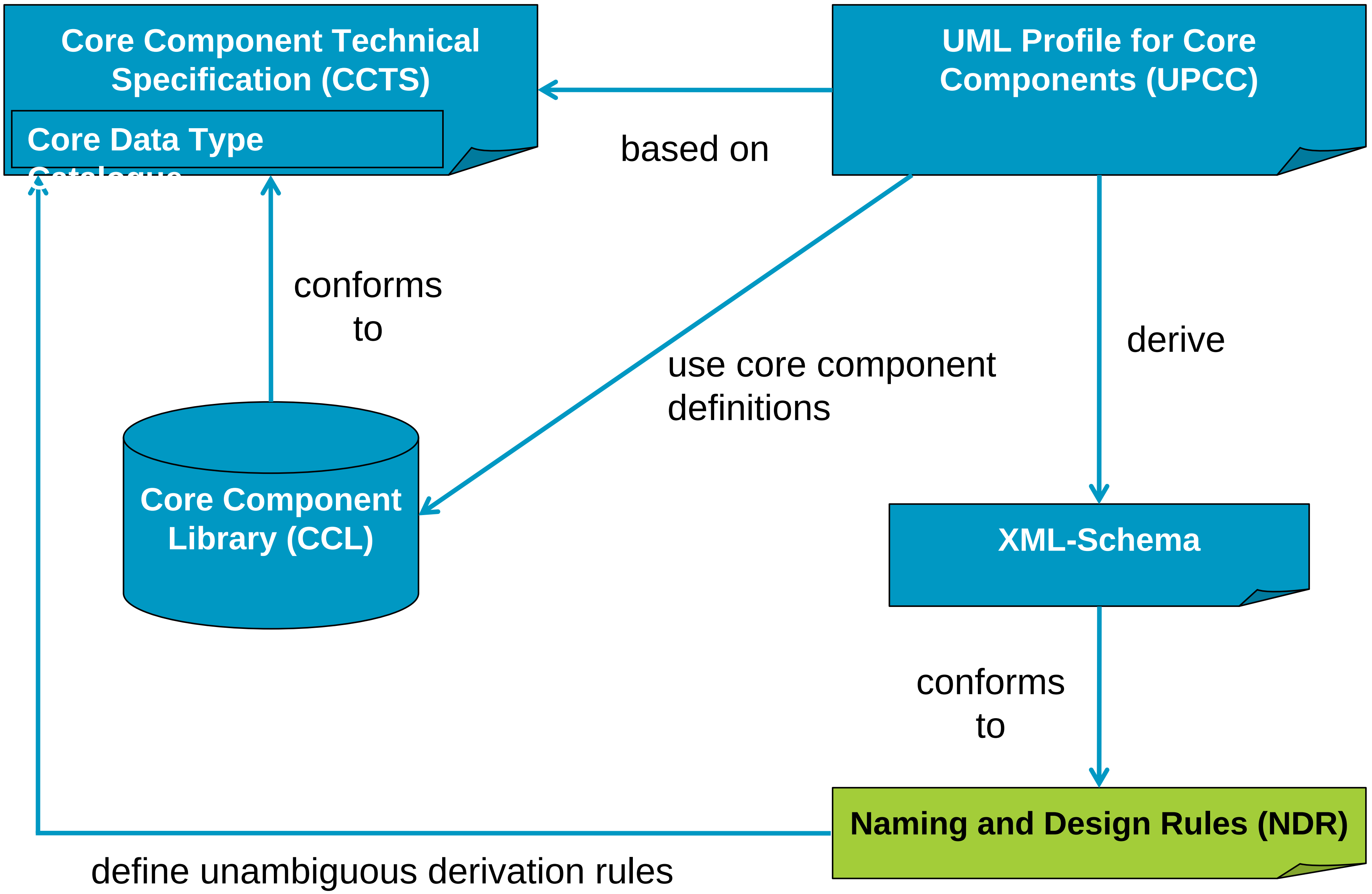
From Core Component Models to XML Schema artifacts

Research Studio Inter-Organisational Systems
Project Public Private Interoperability

Agenda

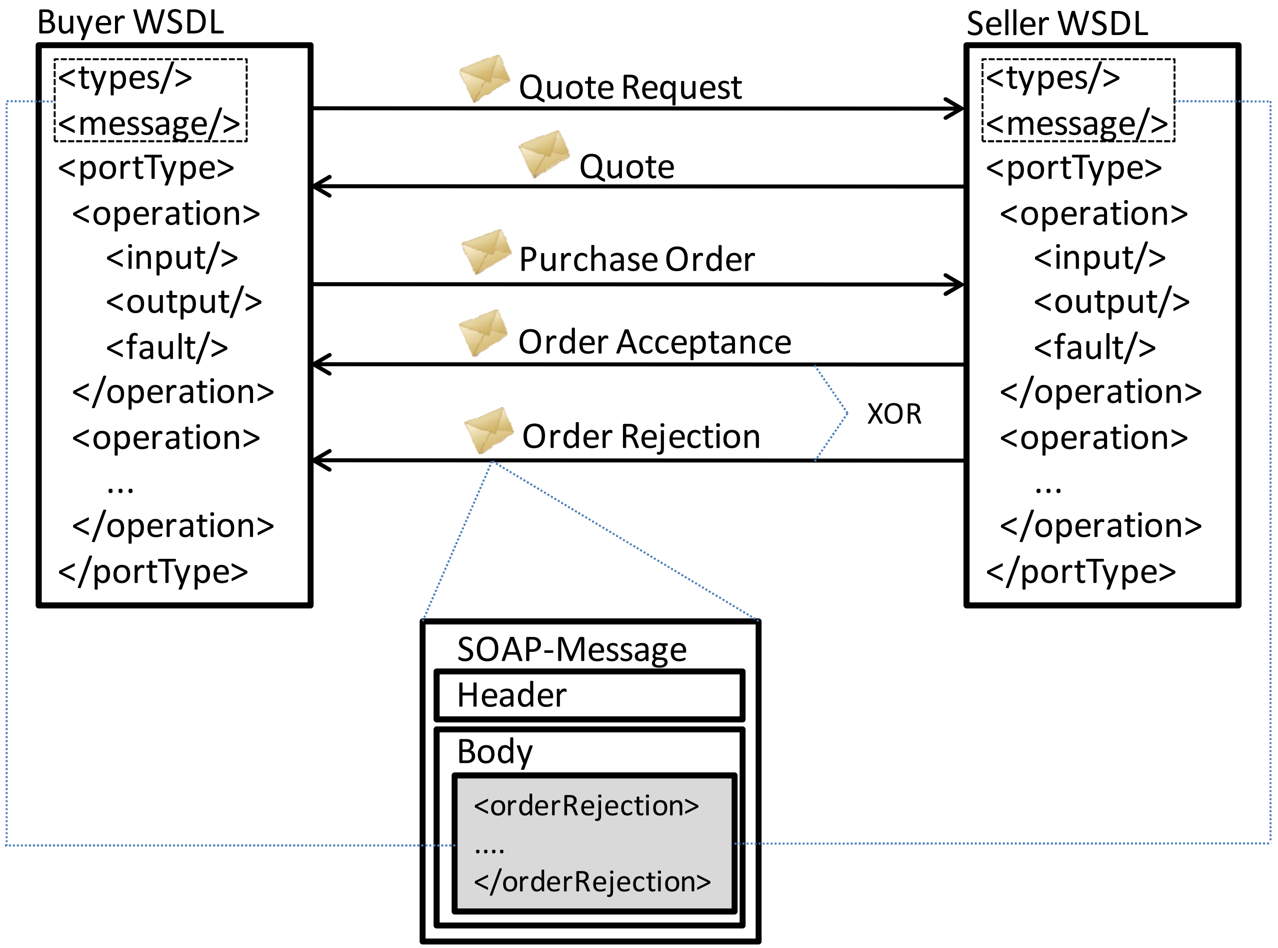
- Motivation
- Introduction to XML Schema
- Naming and Design Rules

Overview of Core Component related UN/CEFACT specifications

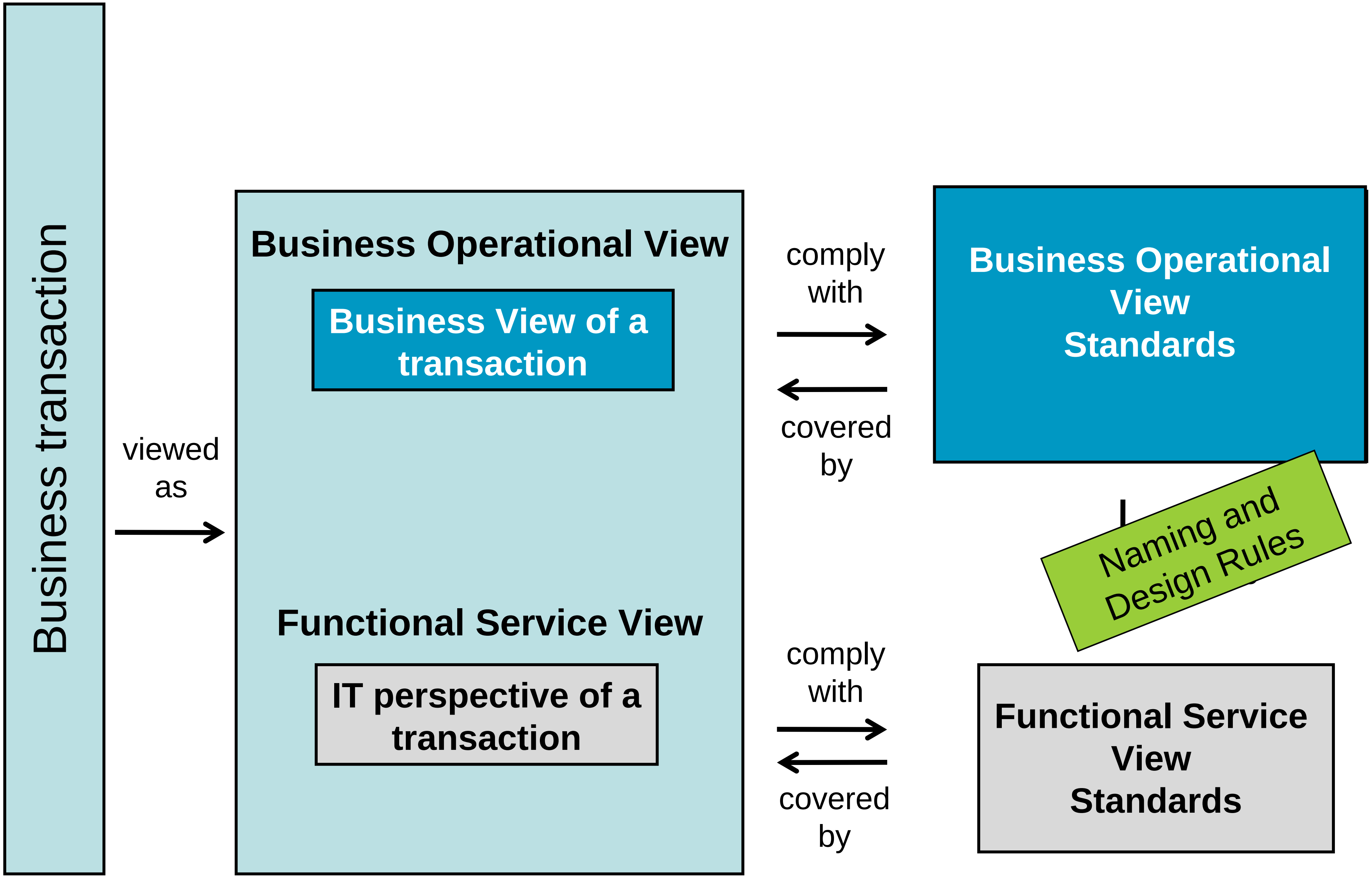


Core Component Technical Specification (CCTS)
Core Data Type Catalogue
UML Profile for Core Components (UPCC)
based on
conforms to
use core component definitions
derive
Core Component Library (CCL)
XML-Schema
conforms to
Naming and Design Rules (NDR)
define unambiguous derivation rules

Motivation: Business documents in a service oriented world



Open-edi Reference Model – IO 14662


Business transaction
viewed as
Business Operational View
Business View of a transaction
Functional Service View
IT perspective of a transaction
comply with
covered by
Business Operational View Standards
Naming and Design Rules
comply with
covered by
Functional Service View Standards

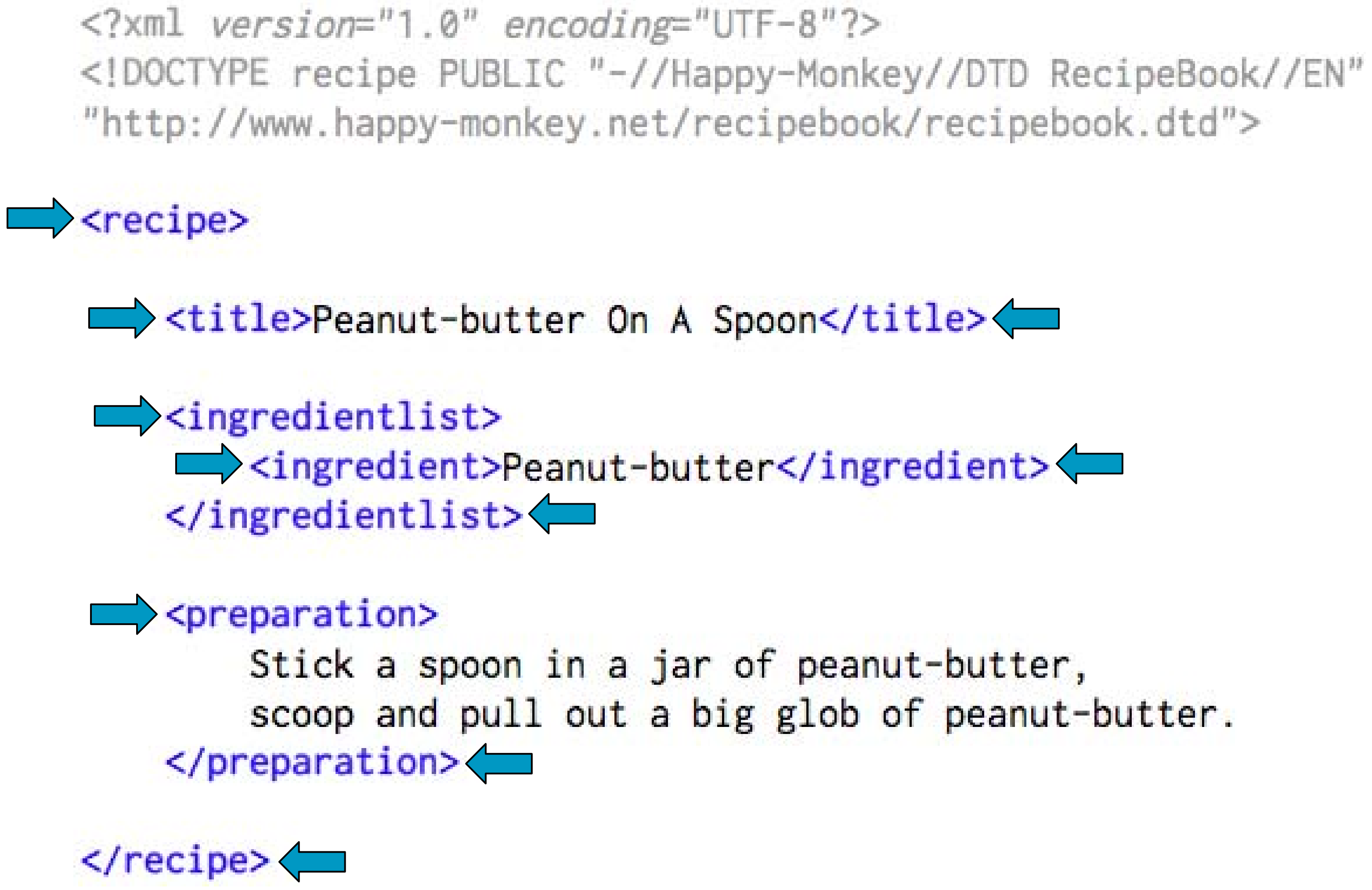
What is XML (eXtensible Markup Lanuage)?

- eXtensible Markup Language is a language, used to create custom markup languages
- XML is a W3C Recommendation
- Subset of SGML (Standard Generalized Markup Language)
- The primary purpose of XML is to share data in a structured manner
- XML is an extensive concept – thus it is not confined to a predefined set of elements/attributes etc
- XML is a meta-language. It is used to create new XML-based languages (although I prefer the term “formats”)

Difference between XML and HTML

- XML was designed to carry data, not displaying data
- Thus, the argument that XML data is both, machine process able and human-readable is not false, but in fact non-sense
- Different goals
  - **XML** was designed to **describe** data and to focus on **what data is**
  - **HTML** was designed to **display** data and to focus on **how data looks**
- HTML is about displaying information, XML is about describing information.

Content vs. Markup



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recipe PUBLIC "-//Happy-Monkey//DTD RecipeBook//EN"
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

   <title>Peanut-butter On A Spoon</title>

   <ingredientlist>
      <ingredient>Peanut-butter</ingredient>
   </ingredientlist>

   <preparation>
      Stick a spoon in a jar of peanut-butter,
      scoop and pull out a big glob of peanut-butter.
   </preparation>

</recipe>
```

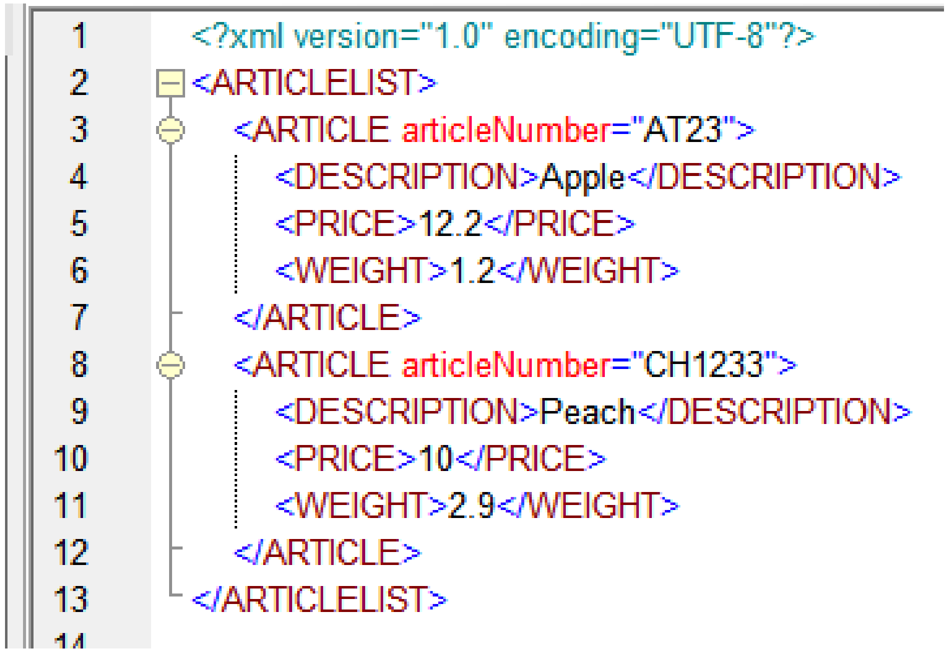
Every XML Document is text-based

- Easy sharing of data between different computers, operating systems, and applications
- Easy sharing of data over the Internet
- Overcoming limitations of binary formats
  - Platform dependent
  - Firewalls
  - Hard to debug
  - File inspection difficult (and often not wanted)
- XML overcomes these limitations
- What are the **disadvantages of text-based formats?**

Application scenarios of XML

- Generally: separate data from presentation
- Use XML to exchange data
- Use XML to store data
- Use XML to create new languages
  - MathML
  - RSS (Really Simple Syndication)

XML example document – List of fruit articles



```xml
<?xml version="1.0" encoding="UTF-8"?>
<ARTICLELIST>
    <ARTICLE articleNumber="AT23">
        <DESCRIPTION>Apple</DESCRIPTION>
        <PRICE>12.2</PRICE>
        <WEIGHT>1.2</WEIGHT>
    </ARTICLE>
    <ARTICLE articleNumber="CH1233">
        <DESCRIPTION>Peach</DESCRIPTION>
        <PRICE>10</PRICE>
        <WEIGHT>2.9</WEIGHT>
    </ARTICLE>
</ARTICLELIST>
```

Correctness of XML documents

- An XML document has two levels of correctness:
- **Well-formedness**
  - A well-formed document conforms to all of XML syntax rules
- **Validity**
  - A valid document additionally conforms to some semantic rules, specified in a Document Type Definition (DTD) or XML Schema
  - An XML document is valid if
    - It is well-formed
    - It follows some semantic rules

XML Syntax rules: Element vs. Tag vs. Attribute

- Element consists of start tag, optional content, and end tag
  - <ARTICLE>Apple</ARTICLE>
- Start tag
  - <ARTCLE>
- Content
  - Apple
- End tag
  - </ARTICLE>
- Attribute
  - <ARTICLE articleNumber="AT23">Apple</ARTICLE>

XML Syntax rules: Element rules

- There must be exactly one root element

- Every element contains of a start tag and an ending tag
  - The content of an element is optional and may be empty
  - <ARTICLE></ ARTICLE> is equivalent to
  - < ARTICLE />
- Tag names are case sensitive
  - < ARTICLE > </article>
- Elements must be correctly nested
  - <p> <b> a text </p> </b>

XML Syntax rules: Attribute rules

- XML elements may have attributes in the start tag
  - e.g. <ARTICLE articleNumber="AT23">….</ARTICLE>

- An attribute must be quoted
  - <ARTICLE articleNumber=AT23> …</ARTICLE> is wrong!

XML Syntax rules: Naming conventions for Tags

- Names may contain letters, number, and other characters
- Names must not start with a number of punctuation letter
- Names must not start with the letters xml, XML, Xml, etc.
- Names cannot contain space characters

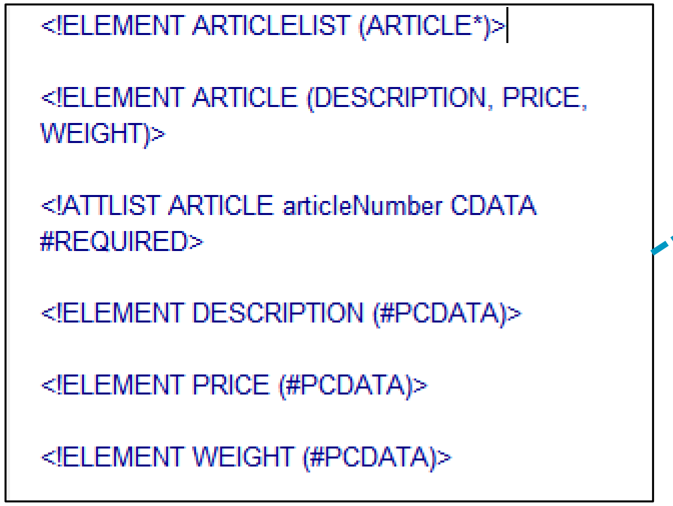
Summary XML Syntax

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML element must be property nested
- All XML document must have root tag
- Attribute values must always be quoted
- In XML white spaces are preserved
- In XML a new line is always stored as LF

An XML document is well-formed, if it follows all of the XML syntax rules!

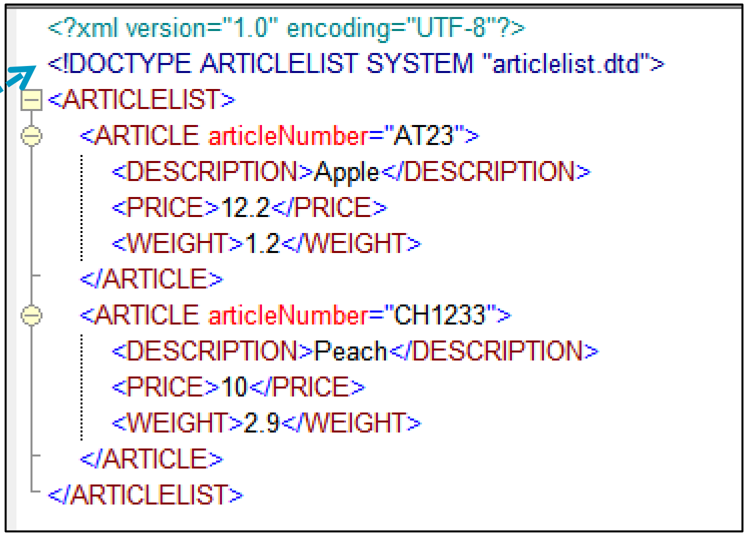
Ensuring validity of XML documents – Document Type Definitions (DTD)

- Specifying the structure and tag-names in XML documents



```
<!ELEMENT ARTICLELIST (ARTICLE*)>

<!ELEMENT ARTICLE (DESCRIPTION, PRICE,
WEIGHT)>

<!ATTLIST ARTICLE articleNumber CDATA
#REQUIRED>

<!ELEMENT DESCRIPTION (#PCDATA)>

<!ELEMENT PRICE (#PCDATA)>

<!ELEMENT WEIGHT (#PCDATA)>
```

Document Type Definition



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ARTICLELIST SYSTEM "articlelist.dtd">
<ARTICLELIST>
   <ARTICLE articleNumber="AT23">
      <DESCRIPTION>Apple</DESCRIPTION>
      <PRICE>12.2</PRICE>
      <WEIGHT>1.2</WEIGHT>
   </ARTICLE>
   <ARTICLE articleNumber="CH1233">
      <DESCRIPTION>Peach</DESCRIPTION>
      <PRICE>10</PRICE>
      <WEIGHT>2.9</WEIGHT>
   </ARTICLE>
</ARTICLELIST>
```

XML document

Shortcomings of Document Type Definitions (DTD)

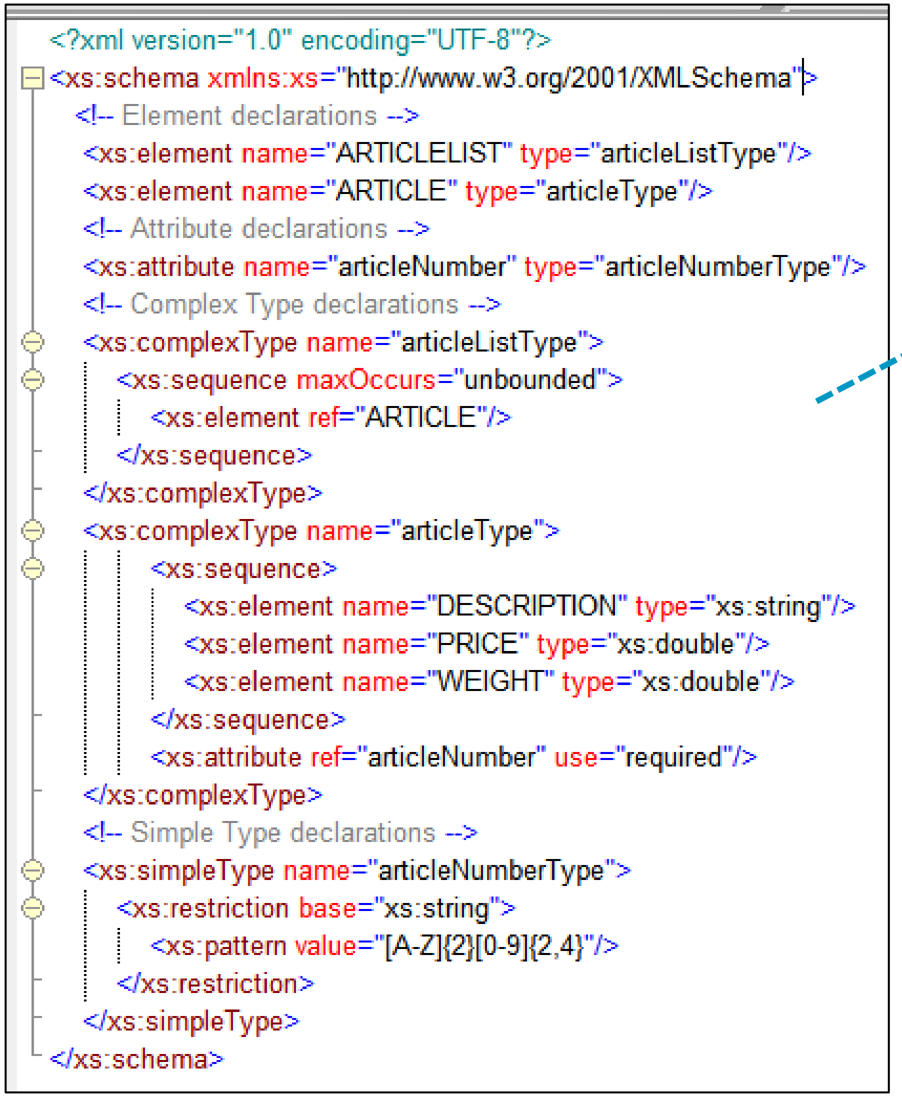
- DTDs use their own, non XML based syntax
- DTDs cannot define restrictions on a granular level like a number range or a string pattern – they only define text or non-text
  - e.g. our article number requires the following pattern: 2 upper case letters followed by minimum 2 to maximum 4 digits
- DTDs do not support namespaces
  - Namespaces are used to unambiguously distinguish between two elements having the same name.

DTDs are rarely used nowadays – XML Schema has become state-of-the-art

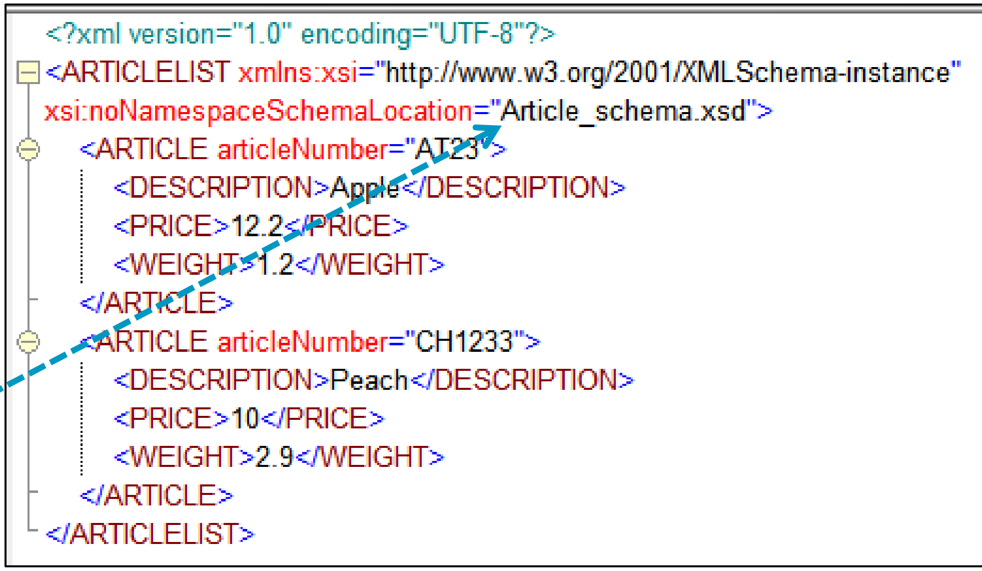
Ensuring validity of XML documents XML Schema (XSD)

- XML Schema (XSD) unambiguously defines which elements and attributes are allowed in an XML document (and in which combination)
- Several advantages over Document Type Definitions (DTD)
    - XSD is defined in XML
    - XSD enables to define you own data types
    - XSD enables to define restrictions on data (e.g. article number pattern)
    - XSD support namespaces

XML Schema example



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Element declarations -->
   <xs:element name="ARTICLELIST" type="articleListType"/>
   <xs:element name="ARTICLE" type="articleType"/>
   <!-- Attribute declarations -->
   <xs:attribute name="articleNumber" type="articleNumberType"/>
   <!-- Complex Type declarations -->
   <xs:complexType name="articleListType">
     <xs:sequence maxOccurs="unbounded">
       <xs:element ref="ARTICLE"/>
     </xs:sequence>
   </xs:complexType>
   <xs:complexType name="articleType">
       <xs:sequence>
         <xs:element name="DESCRIPTION" type="xs:string"/>
         <xs:element name="PRICE" type="xs:double"/>
         <xs:element name="WEIGHT" type="xs:double"/>
       </xs:sequence>
       <xs:attribute ref="articleNumber" use="required"/>
   </xs:complexType>
   <!-- Simple Type declarations -->
   <xs:simpleType name="articleNumberType">
     <xs:restriction base="xs:string">
       <xs:pattern value="[A-Z]{2}[0-9]{2,4}"/>
     </xs:restriction>
   </xs:simpleType>
</xs:schema>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<ARTICLELIST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Article_schema.xsd">
  <ARTICLE articleNumber="AT23">
    <DESCRIPTION>Apple</DESCRIPTION>
    <PRICE>12.2</PRICE>
    <WEIGHT>1.2</WEIGHT>
  </ARTICLE>
  <ARTICLE articleNumber="CH1233">
    <DESCRIPTION>Peach</DESCRIPTION>
    <PRICE>10</PRICE>
    <WEIGHT>2.9</WEIGHT>
  </ARTICLE>
</ARTICLELIST>
```

Elements in XML Schema

- There are two different elements types in XML Schema – **simple** and **complex** elements

- **Simple element**
  - A simple element does not contain child elements
  - A simple element does not contain attributes

- Example Schema:

```
<xs:element name="AddressLine" type="xs:string"/>
```

- Example XML Instance

```
<AddressLine> An address Text </AddressLine>
```

Complex elements – complex Type

- A complex element contains other elements and attributes

- Example Schema

```
<xs:element name="ARTICLE" type="articleType"/>
<xs:complexType name="articleType">
        <xs:sequence>
                <xs:element name="DESCRIPTION" type="xs:string"/>
                <xs:element name="PRICE" type="xs:double"/>
                <xs:element name="WEIGHT" type="xs:double"/>
        </xs:sequence>
        <xs:attribute ref="articleNumber" use="required"/>
</xs:complexType>
```

- Example XML Instance

```
<ARTICLE articleNumber="CH1233">
    <DESCRIPTION>Peach</DESCRIPTION>
    <PRICE>10</PRICE>
    <WEIGHT>2.9</WEIGHT>
</ARTICLE>
```

XML Schema simpleTypes

- A simple type is a direct or indirect derivation from XML Schema built-in data types
- A simple Type may be used to set the value domain of an attribute or an element

- Example

```
<xs:simpleType name="articleNumberType">
        <xs:restriction base="xs:string">
                <xs:pattern value="[A-Z]{2}[0-9]{2,4}"/>
        </xs:restriction>
</xs:simpleType>
```

XML Schema complexTypes

- A complex type supports adding attributes to simple types
- A complex type support nested element types

- Example for nested element types

  -
    ```
    <xsd:complexType name="addressType">
      <xsd:sequence>
              <xsd:element name="address" type="xsd:token"/>
             <xsd:element name="city" type="xsd:token"/>
             <xsd:element name="state" type="xsd:token"/>
             <xsd:element name="country" type="xsd:token"/>
             <xsd:element name="zip" type="xsd:token"/>
           </xsd:sequence>
    </xsd:complexType>
    ```

XML Namespaces

- In XML element names are defined by the developer
- This often results in a conflict when trying to mix XML documents from different XML applications
- Example

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

Domain A

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Domain B

- If both fragments are added together – a naming conflict occurs.
- Table has a different meaning and content in Domain A and Domain B.

Soliving the name conflict using a prefix

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Namespaces cont'd

- When using prefixes in XML so called **namespaces** for the prefixes must be defined
- The namespace is defined by the **xmlns attribute** in the start tag of an element

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

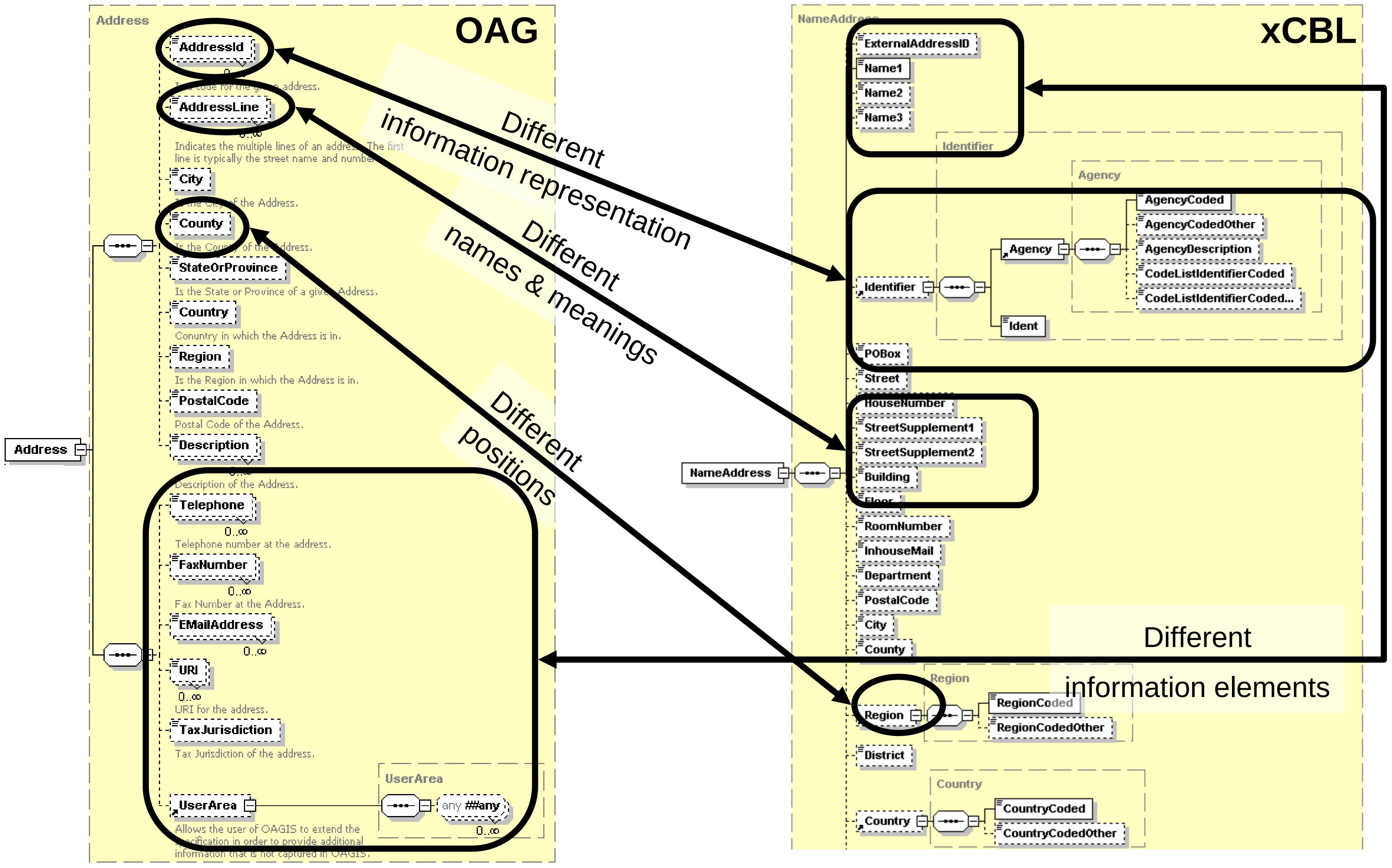
Namespaces may also be defined in the root element

```
<root
   xmlns:h="http://www.w3.org/TR/html4/"
   xmlns:f="http://www.w3schools.com/furniture">
   <h:table>
     <h:tr>
       <h:td>Apples</h:td>
       <h:td>Bananas</h:td>
     </h:tr>
   </h:table>
   <f:table>
     <f:name>African Coffee Table</f:name>
     <f:width>80</f:width>
     <f:length>120</f:length>
   </f:table>
</root>
```

Default namespaces: xmlns="namespaceURI"

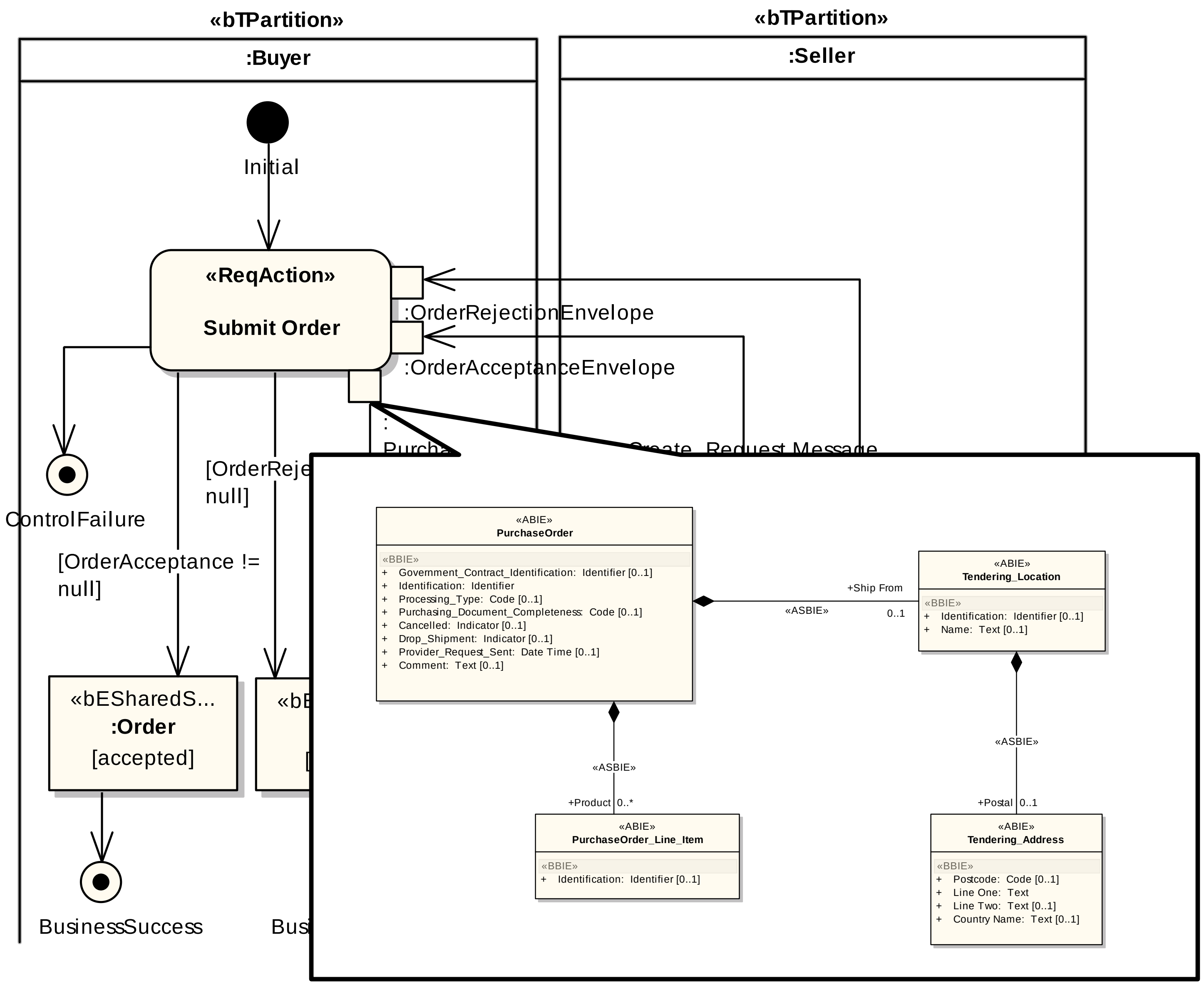
```
<table
    xmlns="http://www.w3schools.com/furniture">
      <name>African Coffee Table</name>
      <width>80</width>
      <length>120</length>
</table>
```

Different Structure and Semantic are currently the biggest issues of XML standards

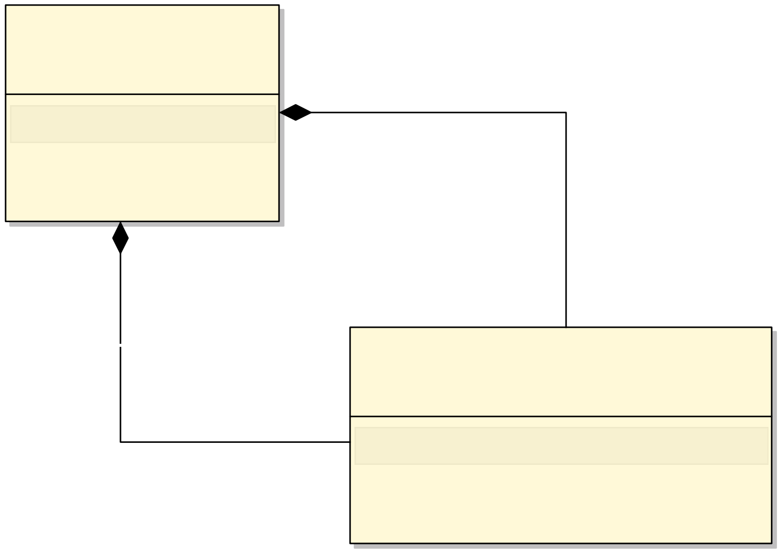


OAG
xCBL
Address
AddressId
The code for the given address.
AddressLine
Indicates the multiple lines of an address. The first line is typically the street name and number.
City
The City of the Address.
County
Is the County of the address.
StateOrProvince
Is the State or Province of a given Address.
Country
Conuntry in which the Address is in.
Region
Is the Region in which the Address is in.
PostalCode
Postal Code of the Address.
Description
Description of the Address.
Telephone
Telephone number at the address.
FaxNumber
Fax Number at the Address.
EMailAddress
URI
URI for the address.
TaxJurisdiction
Tax Jurisdiction of the address.
UserArea
any ##any
Allows the user of OAGIS to extend the specification in order to provide additional information that is not captured in OAGIS.
NameAddress
ExternalAddressID
Name1
Name2
Name3
Identifier
Agency
AgencyCoded
AgencyCodedOther
AgencyDescription
CodeListIdentifierCoded
CodeListIdentifierCoded...
Ident
POBox
Street
HouseNumber
StreetSupplement1
StreetSupplement2
Building
Floor
RoomNumber
InhouseMail
Department
PostalCode
City
County
Region
RegionCoded
RegionCodedOther
District
Country
CountryCoded
CountryCodedOther
Different information representation
Different names & meanings
Different positions
Different information elements

Business documents in UN/CEFACT's Modeling Methodology



From conceptual models to deployment artifacts

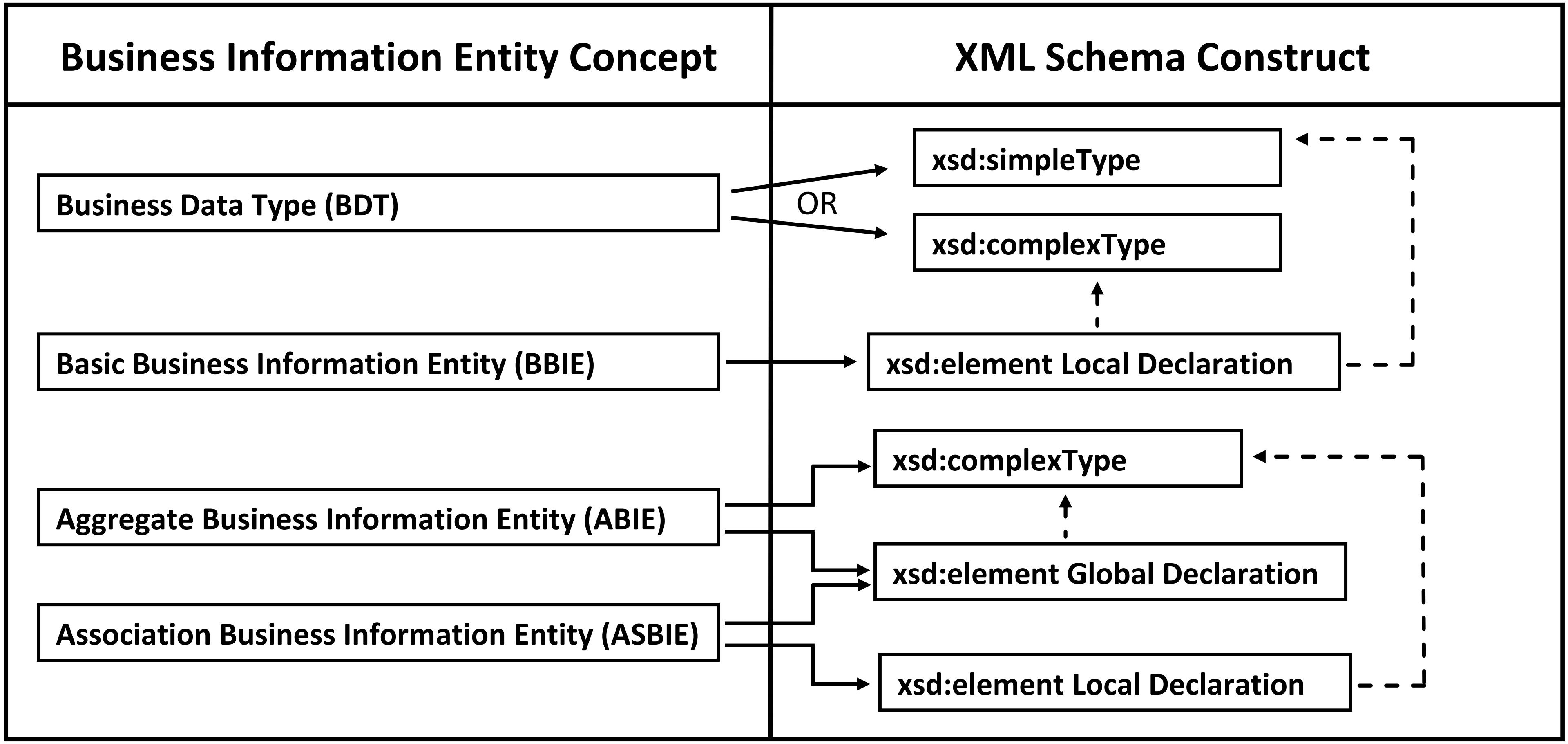


Naming and Design Rules

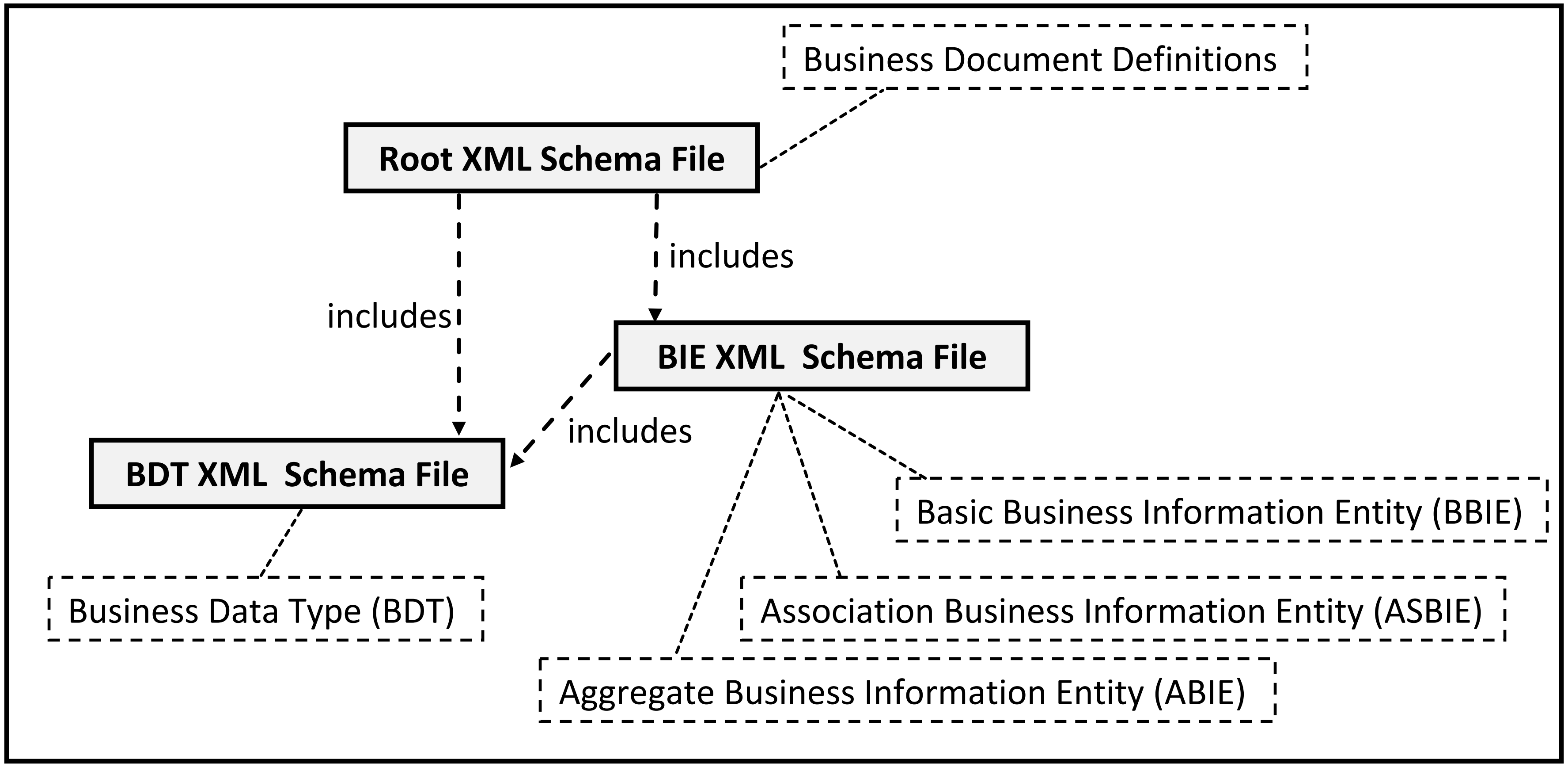
VIENNA **AddIn**

```
<xsd:complexType name="US_PersonType">
		<xsd:sequence>
				<xsd:element name= "DateofBirth" type="udt1:DateType">
				<xsd:element name="FirstName" type="udt1:TextType"/>
				<xsd:element name="US_Work" type="bie1:US_AddressType"/>
				<xsd:element name="US_Private" type="bie1:US_AddressType"/>
		<xsd:sequence>
</xsd:complexType>

<xsd:complexType name="US_AddressType">
 […]
</xsd:complexType>
```

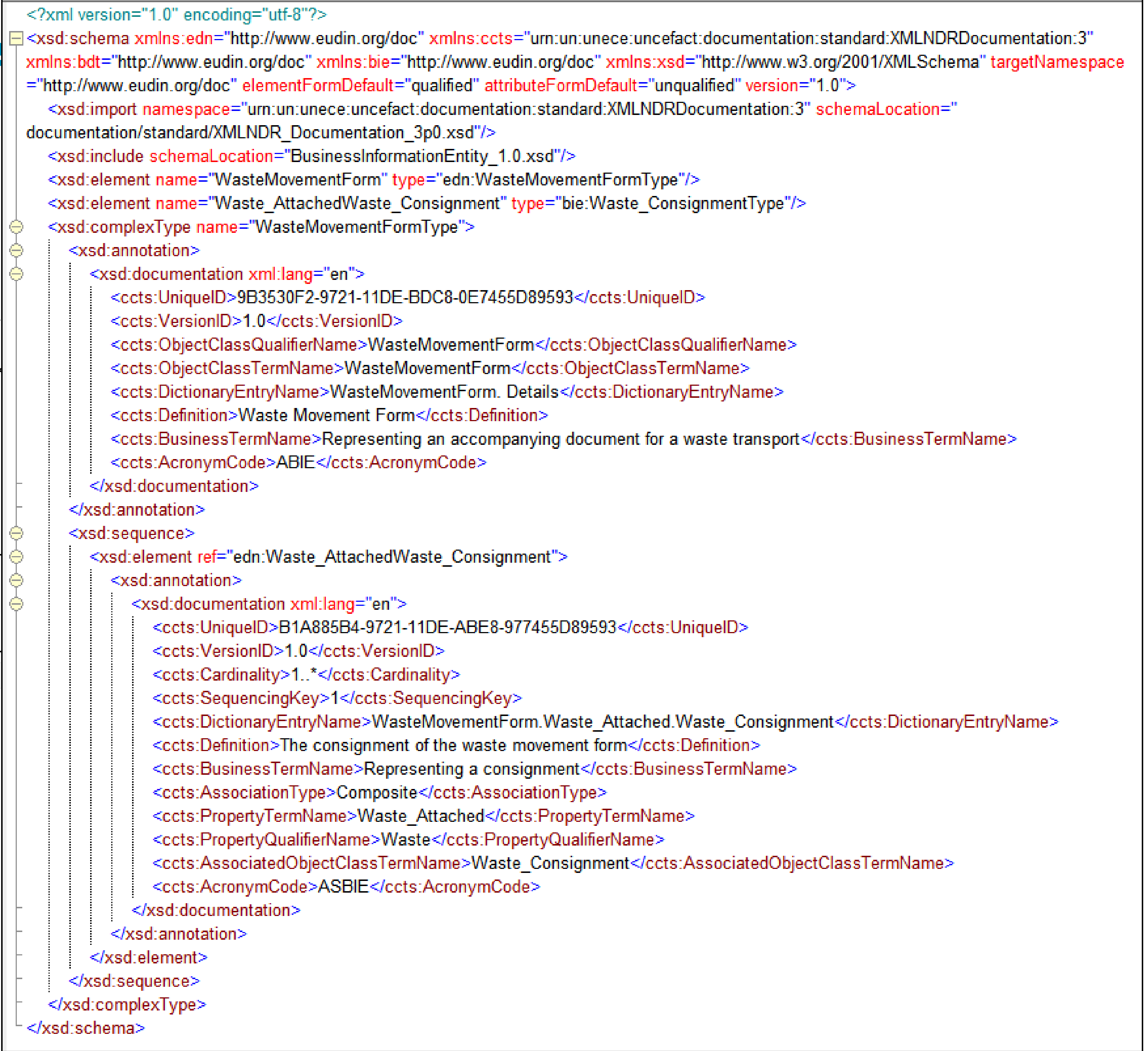
Transformation concepts


Business Information Entity Concept
XML Schema Construct
Business Data Type (BDT)
OR
xsd:simpleType
xsd:complexType
Basic Business Information Entity (BBIE)
xsd:element Local Declaration
Aggregate Business Information Entity (ABIE)
Association Business Information Entity (ASBIE)
xsd:complexType
xsd:element Global Declaration
xsd:element Local Declaration

Resulting XML files

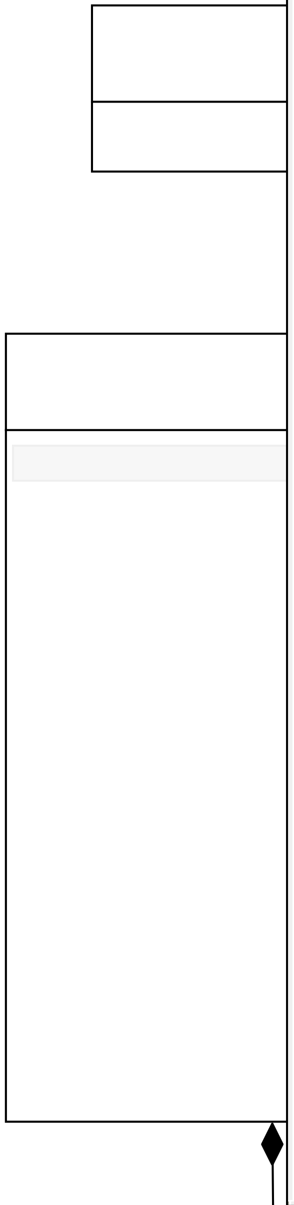

Business Document Definitions
Root XML Schema File
includes
includes
BIE XML Schema File
includes
BDT XML Schema File
Basic Business Information Entity (BBIE)
Business Data Type (BDT)
Association Business Information Entity (ASBIE)
Aggregate Business Information Entity (ABIE)

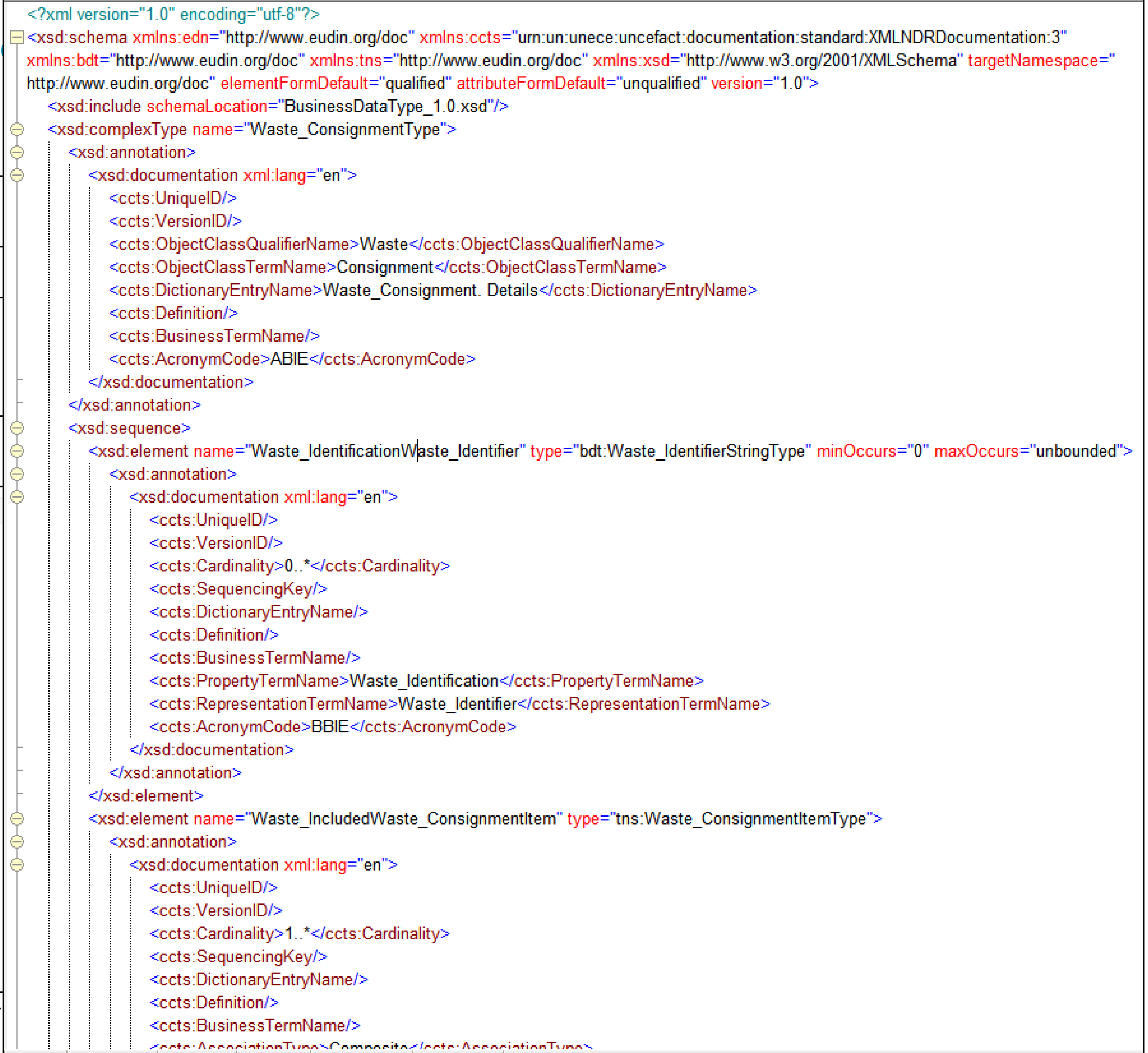
Example



```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:edn="http://www.eudin.org/doc" xmlns:ccts="urn:un:unece:uncefact:documentation:standard:XMLNDRDocumentation:3"
xmlns:bdt="http://www.eudin.org/doc" xmlns:bie="http://www.eudin.org/doc" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace
="http://www.eudin.org/doc" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:un:unece:uncefact:documentation:standard:XMLNDRDocumentation:3" schemaLocation="
documentation/standard/XMLNDR_Documentation_3p0.xsd"/>
  <xsd:include schemaLocation="BusinessInformationEntity_1.0.xsd"/>
  <xsd:element name="WasteMovementForm" type="edn:WasteMovementFormType"/>
  <xsd:element name="Waste_AttachedWaste_Consignment" type="bie:Waste_ConsignmentType"/>
  <xsd:complexType name="WasteMovementFormType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        <ccts:UniqueID>9B3530F2-9721-11DE-BDC8-0E7455D89593</ccts:UniqueID>
        <ccts:VersionID>1.0</ccts:VersionID>
        <ccts:ObjectClassQualifierName>WasteMovementForm</ccts:ObjectClassQualifierName>
        <ccts:ObjectClassTermName>WasteMovementForm</ccts:ObjectClassTermName>
        <ccts:DictionaryEntryName>WasteMovementForm. Details</ccts:DictionaryEntryName>
        <ccts:Definition>Waste Movement Form</ccts:Definition>
        <ccts:BusinessTermName>Representing an accompanying document for a waste transport</ccts:BusinessTermName>
        <ccts:AcronymCode>ABIE</ccts:AcronymCode>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="edn:Waste_AttachedWaste_Consignment">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            <ccts:UniqueID>B1A885B4-9721-11DE-ABE8-977455D89593</ccts:UniqueID>
            <ccts:VersionID>1.0</ccts:VersionID>
            <ccts:Cardinality>1..*</ccts:Cardinality>
            <ccts:SequencingKey>1</ccts:SequencingKey>
            <ccts:DictionaryEntryName>WasteMovementForm.Waste_Attached.Waste_Consignment</ccts:DictionaryEntryName>
            <ccts:Definition>The consignment of the waste movement form</ccts:Definition>
            <ccts:BusinessTermName>Representing a consignment</ccts:BusinessTermName>
            <ccts:AssociationType>Composite</ccts:AssociationType>
            <ccts:PropertyTermName>Waste_Attached</ccts:PropertyTermName>
            <ccts:PropertyQualifierName>Waste</ccts:PropertyQualifierName>
            <ccts:AssociatedObjectClassTermName>Waste_Consignment</ccts:AssociatedObjectClassTermName>
            <ccts:AcronymCode>ASBIE</ccts:AcronymCode>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example f





```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:edn="http://www.eudin.org/doc" xmlns:ccts="urn:un:unece:uncefact:documentation:standard:XMLNDRDocumentation:3"
xmlns:bdt="http://www.eudin.org/doc" xmlns:tns="http://www.eudin.org/doc" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="
http://www.eudin.org/doc" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
  <xsd:include schemaLocation="BusinessDataType_1.0.xsd"/>
  <xsd:complexType name="Waste_ConsignmentType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        <ccts:UniqueID/>
        <ccts:VersionID/>
        <ccts:ObjectClassQualifierName>Waste</ccts:ObjectClassQualifierName>
        <ccts:ObjectClassTermName>Consignment</ccts:ObjectClassTermName>
        <ccts:DictionaryEntryName>Waste_Consignment. Details</ccts:DictionaryEntryName>
        <ccts:Definition/>
        <ccts:BusinessTermName/>
        <ccts:AcronymCode>ABIE</ccts:AcronymCode>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="Waste_IdentificationWaste_Identifier" type="bdt:Waste_IdentifierStringType" minOccurs="0" maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            <ccts:UniqueID/>
            <ccts:VersionID/>
            <ccts:Cardinality>0..*</ccts:Cardinality>
            <ccts:SequencingKey/>
            <ccts:DictionaryEntryName/>
            <ccts:Definition/>
            <ccts:BusinessTermName/>
            <ccts:PropertyTermName>Waste_Identification</ccts:PropertyTermName>
            <ccts:RepresentationTermName>Waste_Identifier</ccts:RepresentationTermName>
            <ccts:AcronymCode>BBIE</ccts:AcronymCode>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="Waste_IncludedWaste_ConsignmentItem" type="tns:Waste_ConsignmentItemType">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            <ccts:UniqueID/>
            <ccts:VersionID/>
            <ccts:Cardinality>1..*</ccts:Cardinality>
            <ccts:SequencingKey/>
            <ccts:DictionaryEntryName/>
            <ccts:Definition/>
            <ccts:BusinessTermName/>
            <ccts:AssociationType>Composite</ccts:AssociationType>
```

Thank you for your attention

```
<Lecturer>
    <Name>**Philipp Liegl**</Name>
    <Company>**Research Studios Austria**</Company>
    <Department>**Research Studio Inter-Organisational Systems**</Department>
    <Address>
        <Street>**Thurngasse 8/3/20**</Street>
        <ZIP>**1090**</ZIP><City>**Vienna**</City>
        <Country>**Austria**</Country>
    </Address>
    <Contact>
        <Email>**office.ios@resarchstudio.at**</Email>
        <Http>**http://www.researchstudio.at**</Http>
    </Contact>
    <? Presentation status=“**questions**” ?>
</Lecturer>
```