[Abstract] With the development of applications of XML and the publication of XML schema recommendation by W3C, XML+XML schema-based EDIFACT is becoming a good solution to the electronic commerce applications. Now a lot of large-scale enterprises want to make the traditional electronic commerce systems migrated to XML schema-based EDIFACT. At the same time many small and medium-scale enterprises want to implement electronic commerce systems with XML schema-based EDIFACT. The design and implementation of EDIFACT message structures based on XML schema is the most important work and the base of applying XML +XML schema-based EDIFACT to electronic commerce systems and implementations of new electronic commerce systems. In this paper we present a method of design and implementation of EDIFACT message structures based on XML schema and illustrate it with examples.

[Key word] electronic commerce, XML, EDIFACT, XML schema, message structure, datatype

1. Introduction

In the past years a lot of electronic commerce systems were based on EDIFACT [1] (Electronic Data Interchange For Administration, Commerce and Transport), which is an international EDI (Electronic Data Interchange) standard developed by the United Nations. This international standard includes the rules on the application level for the structuring of user data and of associated service data in the interchange of messages in an open environment. Beside the syntax the EDIFACT standard [2] covers also the definition of data elements (the data information as basic component for message types), segments (functionally related sets of data elements) and message types (structured representation of the full information on an electronic commerce transaction).

Unfortunately, only those large-sized enterprises can afford the expensive electronic commerce systems based on EDIFACT. The small and medium scale enterprises cannot bear the cost of implementation and the difficulty of development. This cost includes the substantial investment in legacy information system, managements, maintenance and software etc [3]. At the same time EDIFACT has been criticized for poor design, confusing or absent semantics [4]. Those difficulties block the implementation and generalization of electronic commerce system based on EDIFACT.

The emergence of XML [5] (eXtensible Markup Language) resolves those problems. In 1996 W3C (World Wide Web Consortium) joined with SGML (Standard Generalized Markup Language) experts to form an SGML Working Group, which strategically pruned SGML into a refined subset
now known as XML, which, published in 1997, is a metalanguage and can be as the standard for self-describing data exchange in Internet applications. XML makes electronic commerce system developed rapidly and maintained easily. Now XML has become the first choice in the field of defining data interchange formats in electronic commerce. In May 2001 W3C published a recommendation of XML Schema [6,7,8]. XML schema can be used to describe the structure of XML document and define the semantics of element.

Thus we can use XML+XML schema-based EDIFACT as the solution to electronic commerce applications. In order to develop electronic commerce system based on XML schema-based EDIFACT, the first work is to use the XML schema to describe the EDIFACT message structure, which is a very important work and the base of applying XML to electronic commerce system based on EDIFACT and of implementing new electronic commerce systems.

This paper is organized as follows. In section 2, we discuss the related work. In section 3, we describe the message structure of EDIFACT. In section 4, we present how to specify EDIFACT message structure with XML schema. The section 5 concludes our work.

2. Related work

There have been a lot of papers and projects discussing the XML/EDI owing to the birth of XML and XML schema. But until now there is no discussion on using XML schema to define EDIFACT.

The CEN/ISSS Electronic Commerce Workshop published Ref No CWA 14162: Datatyping for Electronic Data Interchange in March 2000 [9]. This document concentrates on techniques for defining and constraining data or code set values used within B2B (business-to-business) electronic data interchange messages. The document only discusses the several datatyping for Electronic Data Interchange and don’t give the EDIFACT message structure defined by XML schema.

Multek Sweden AB developed IGML [10] in February 2000. It uses DTD to describe the EDIFACT message structure.

Michael Koehne also used the DTD to describe the EDIFACT in 2000[11].

Due to the shortage of DTD, the DTD-specified EDIFACT is not completely consistent to EDIFACT. For example, we cannot use DTD to define datatype we need and constrain the element content according to the special application.

3. EDIFACT

Since 1988 the United Nations has been developing the EDIFACT to meet the requirements of an internationally valid general business standard. This international standard includes the rules on the application level for the structuring of user data and of associated service data in the interchange of messages in an open environment. Beside the syntax, the EDIFACT standard covers also the definition of data elements (the data information as basic component for message types), segments (functionally related sets of data elements), and message types (structured representation of the full information on an electronic business transaction).

EDIFACT can be considered as a dynamic standard since new message types are developed and definitions of existing message types are be changing in the course of time. The complete documentation of the EDIFACT guidelines is included in an UN/EDIFACT directory (Fig.1 UN/EDIFACT directory), which comprises the message type directory, the segment type directory,
the composite data element type directory, the simple data element type directory, and the code list
directory.

EDIFACT message comprises an ordered set of segments. Segments may be grouped. A
segment group comprises an ordered set of segments: a trigger segment and at least one more
segment or segment group. The trigger segment shall be the first segment in the segment group,
shall have a status of mandatory and a maximum number of occurrences of one. A segment
comprises an ordered set of stand-alone data elements and/or composite data elements, each of
which are permitted to repeat, if so stated in the segment specification. A composite data element
comprises an ordered set of two or more component data elements. A simple data element contains
a single data element value. A simple data element is used either as a stand-alone data element or
as a component data element. A stand-alone data element occurs in a segment outside a composite
data element. A component data element occurs within a composite data element. The message
structure of CALINF is as follows:
<table>
<thead>
<tr>
<th>Segment Group</th>
<th>Description</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message header</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Beginning of message</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Free text</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Measurements</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Number of units</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Reference</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Name and address</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Contact information</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Communication contact</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Details of transport</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Place/location identification</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Reference</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>Quantity</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Free text</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 2 the message structure of CALINF
4. Designing EDIFACT message structures with XML schema

The EDIFACT message structures documentation (Fig.3 the documentation structure of EDIFACT message structures based on XML schema) we developed comprises simpleEDType.xsd, simpleEDeclaANDcompositeEDType.xsd, compositeEDeclaANDsegmnetEDtype.xsd, segmentEDecla.xsd, and the message documents that include the declarations of segment group element and message and the definitions of segment group element datatype and message datatype. simpleEDType.xsd includes the definitions of simple element datatype of EDIFACT. simpleEDeclaANDcompositeEDType.xsd includes the definitions of composite element datatype of EDIFACT and declarations of simple element of EDIFACT. compositeEDeclaANDsegmnetEDtype.xsd includes the declarations of composite element datatype of EDIFACT and the definitions of segment element datatype of EDIFACT. segmentEDecla.xsd includes the declarations of the segment element of EDIFACT. Each message structure is a document that includes the declarations of segment group element and message and the definitions of segment group element datatype and message datatype.

![Diagram of message structure documentation](CALINF.xsd -- segmentEDecla.xsd -- VESDEP.xsd)

Fig.3 the documentation structure of EDIFACT message structures based on XML schema

When we describe the EDIFACT message structure with XML schema, the most important issue is defining the element datatypes based on EDIFACT.

The XML schema specification defines the following three kinds of datatypes: Primitive datatypes (string, binary, etc.), Derived datatypes (CDATA, token, etc), User-derived datatypes. The User-derived datatypes allow users to create complex datatypes that are composed of sets of primitive datatypes. User-derived datatypes can include enumerated lists of values, which can include values of different datatypes. For the string data type, users can define patterns that the string must conform to. For numeric values, maximum and minimum values can be specified (inclusively or exclusively), as scale and precision. Booleans can be represented as true or 0 and false or 1. Dates and time can be expressed using various ISO 8601-based formats. Datatypes can also be derived as the union of two other datatypes, as lists of values conforming to another datatype, or as restrictions on an existing datatype.
We take the simple element 5243 as an example to show the definition of element datatypes. We know that the value of element 5243 comprises A, B, C, D, E, F, K, M, N, Q, R and S. So we make the base datatype of the datatype of 5243 enumeration datatype. The dataType of 5243 is s5243simpleElementDataType, which is a complexType. At the same time we define the attributes of anno, flag, code, desc, repr and posi for every simple element in order to be understood easily by the developer. The element 5243 is defined with XML schema as follows:

```xml
- <xs:complexType name="s5243simpleElementDataType">
  - <xs:simpleContent>
    - <xs:extension base="s5243enumerationType">
      <xs:attributeGroup ref="SimpleElementAttributeGr" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

- <xs:simpleType name="s5243enumerationType">
  - <xs:restriction base="xs:string">
    <xs:enumeration value="A" />
    <xs:enumeration value="B" />
    <xs:enumeration value="C" />
    <xs:enumeration value="D" />
    <xs:enumeration value="E" />
    <xs:enumeration value="F" />
    <xs:enumeration value="K" />
    <xs:enumeration value="M" />
    <xs:enumeration value="N" />
    <xs:enumeration value="Q" />
    <xs:enumeration value="R" />
    <xs:enumeration value="S" />
  </xs:restriction>
</xs:simpleType>

- <xs:attributeGroup name="SimpleElementAttributeGr">
  <xs:attribute name="anno" type="xs:string" use="required" />
  <xs:attribute name="flag" type="xs:string" use="optional" />
  <xs:attribute name="code" type="xs:integer" use="required" />
  <xs:attribute name="desc" type="xs:string" use="optional" />
  <xs:attribute name="repr" type="xs:string" use="required" />
  <xs:attribute name="posi" type="xs:integer" use="optional" />
</xs:attributeGroup>

Now we can declare 5243 as follows:
```
```
<xs:element name="s5243" type="s5243simpleElementDataType" />
```

In succession we show how to define the datatypes of composite elements and declare composite elements. We take an example for the composite element c082 to show the definition of element datatypes. C082 is described according to the definition of EDIFACT as follows:

C082 PARTY IDENTIFICATION DETAILS
Desc: Identification of a transaction party by code.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>Party identifier</td>
<td>M</td>
<td>an..35</td>
</tr>
<tr>
<td>020</td>
<td>Code list identification code</td>
<td>C</td>
<td>an..17</td>
</tr>
<tr>
<td>030</td>
<td>Code list responsible agency code</td>
<td>C</td>
<td>an..3</td>
</tr>
</tbody>
</table>

First we define its datatype:

- `<xs:complexType name="c082compositeElementDataType">
-   `<xs:sequence>
-     `<xs:element ref="s3039" />
-     `<xs:element ref="s1131" minOccurs="0" />
-     `<xs:element ref="s3055" minOccurs="0" />
-   </xs:sequence>
-   `<xs:attributeGroup ref="CompositeElementAttributeGr" />
- </xs:complexType>

We define attributes to describe segment element:

- `<xs:attributeGroup name="SegmentAttributeGr">
-     `<xs:attribute name="abbr" type="xs:string" use="optional" />
-     `<xs:attribute name="anno" type="xs:integer" use="required" />
-     `<xs:attribute name="desc" type="xs:string" use="optional" />
-   </xs:attributeGroup>

We can declare `c082` as follows:

`<xs:element name="c082" type="c082compositeElementDataType" />

Then we can declare `RFF` as follows:

`<xs:element name="RFF" type="RFFsegmentDataType" />

Then we can declare `RFF` as follows:

We take an example for segment element `RFF` to show the definition of segment element datatypes. `RFF` is described according to the definition of EDIFACT as follows:

**RFF**

Function: To specify a reference.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>C506 REFERENCE</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

First we define its datatype:

```
<xs:complexType name="RFFsegmentDataType">
  <xs:sequence>
    <xs:element ref="c506"/>
  </xs:sequence>
  <xs:attributeGroup ref="SegmentAttributeGr"/>
</xs:complexType>
```

We define attributes to describe segment element:

```
- `<xs:attributeGroup name="SegmentAttributeGr">
-     `<xs:attribute name="abbr" type="xs:string" use="optional" />
-     `<xs:attribute name="anno" type="xs:integer" use="required" />
-     `<xs:attribute name="desc" type="xs:string" use="optional" />
-   </xs:attributeGroup>
```

Then we can declare `RFF` as follows:

```
<xs:element name="RFF" type="RFFsegmentDataType" />
```
We can define other simple element, composite element, segment element and segment group element of EDIFACT with the method we introduced before.

In the last we can get the EDIFACT message structure based on XML schema. For example, the CALINF message structure is in CALINF.xsd. Its content is partly shown as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 (http://www.xmlspy.com) by myh (company) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:include schemaLocation="http://xml.whut.edu.cn/xmledi/segmentDeclaration.xsd"/>
  <xs:element name="CALINF" type="MessageCalinfDataType"/>
  <xs:element name="SegmentGroup1" type="CalinfSegmentGr1DataType"/>
  <xs:element name="SegmentGroup2" type="CalinfSegmentGr2DataType"/>
  <xs:element name="SegmentGroup3" type="CalinfSegmentGr3DataType"/>
  <xs:element name="SegmentGroup4" type="CalinfSegmentGr4DataType"/>
  <xs:element name="SegmentGroup5" type="CalinfSegmentGr5DataType"/>
  <xs:element name="SegmentGroup6" type="CalinfSegmentGr6DataType"/>
  <xs:element name="SegmentGroup7" type="CalinfSegmentGr7DataType"/>
  <xs:complexType name="MessageCalinfDataType">
    <xs:sequence>
      <xs:element ref="UNH"/>
      <xs:element ref="BGM"/>
      <xs:element ref="DTM" minOccurs="0" maxOccurs="9"/>
      <xs:element ref="SegmentGroup1" minOccurs="0" maxOccurs="99"/>
      <xs:element ref="SegmentGroup2" minOccurs="0" maxOccurs="9"/>
      <xs:element ref="SegmentGroup3" maxOccurs="9"/>
      <xs:element ref="SegmentGroup5"/>
      <xs:element ref="SegmentGroup7" minOccurs="0" maxOccurs="9"/>
      <xs:element ref="UNT"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

5 conclusions

With the development of electronic commerce and application of XML and the publication of XML schema recommendation, a lot of large scale enterprises want to make electronic commerce systems migrate from traditional EDIFACT to XML schema-based EDIFACT. At the same time many SMEs want to implement electronic commerce system with XML schema-based EDIFACT. XML+XML schema-based EDIFACT is a good solution to the electronic commerce applications. The XML document includes the message information while the XML schema can be used to define the message structure and datatypes of elements according to EDIFACT, so we can use the XML schema to validate the XML documents and make them conform to the specification of the XML schema –based EDIFACT. In this paper we use XML schema to define EDIFACT, which is the base of generalization of electronic commerce systems based on XML schema-based EDIFACT.

The documents we developed comprise simpleEDType.xsd,
simpleEDeclaANDcompositeEDType.xsd, compositeEDeclaANDsegmnetEDtype.xsd, segmentEDecla.xsd, and the documents that include the declarations of segment group elements and message and the definitions of segment group element datatypes and message datatype. The documentation structure is consistent to the EDIFACT directory and is easy to be understood. Dividing definitions of simple elements, composite elements, segments, segment group into different documents makes them reusable and easy to apply.

Till now we have defined about twenty message structures with the method presented in our paper. We will go on this work until all EDIFACT message structures be specified with XML schema.

Reference

2. ISO 9735--1: Electronic data interchange for administration, commerce and transport
   (EDIFACT) — Application level syntax rules (Syntax version number: 4) —Part 1:Syntax
   rules common to all parts, together with syntax service directories for each of the parts.
4. Nabil R.Adam and Yesha, editors, Electronic Commerce: Current Research Issues and
   Applications, Volume 1028 of Lecture Notes in computer Science. Springer, Berlin, Germany,
   1996.
5. Extensible Markup Language (XML) 1.0 (Second Edition),
   W3C Recommendation, 2 May 2001
9. CEN/ISSS Workshop Agreement: CWA 14162 Datatyping for Electronic Data Interchange,
   March 2001
    IN, October 23, 1999