Track&Trace 시스템의 XML 스키마와 XSL 설계

유정순, 하수철
대전대학교 컴퓨터정보통신공학부

Design of XML Schema and XSL for the Track&Trace System

Joung-Soon Yoo, Soo-Cheol Ha
Dept. of Computer Engineering, Daejeon University
E-mail : jsyoo@zeus.dju.ac.kr, soocha@dju.ac.kr

요 약

본 논문은 UML로 모델링 된 Track&Trace 시스템[1]을 확장성과 유연성이 뛰어나며, 웹 상에서 안전하고 신뢰성 있으며 다양하고 광범위한 데이터 교환을 위한 XML로의 설계에 관한 연구이다. 이를 위해 UML 클래스 다이어그램의 클래스, 멤버 데이터와 데이터 타입을 이용해 Well-Formed XML에서 요소와 컨텐츠로 변환한다. 이러한 XML의 구조를 명확하게 정의하기 위해 클래스 다이어그램의 데이터 타입과 다중성을 이용해 DTD를 설계하고, 동시에 확장성을 갖춘 XML 스키마를 설계한다. 또한, XML 문서와 독립적으로 출력을 위한 스타일 지정인 CSS와 XSL도 설계한다.

1. 서론

물류 서비스가 차지하는 비용이 일본은 8.8%, 미국은 7.8%인데 비해 우리 나라는 매출액의 17% 이상을 차지하고 있으며, 이것은 국가 및 기업 경쟁력에서 매우 큰 장애가 되고 있다. Track&Trace 시스템은 실제로 전자상거래 등에서 배송되는 물류들에 대해 도착지까지의 경로 및 운송 상태 등을 실시간으로 모니터링 할 수 있는 기능 제공을 목표로 하고 있다[1].

XML은 웹, B2B와 같은 전자상거래, 디지털 도서관, 전자 출판 등에서 매우 광범위하게 사용되고 있으며, WWW, 인트라넷 등과 같이 데이터와 포맷 두 가지 모두를 공유하고자 할 때 유용하다.

HTML은 문서의 내용이나 구조, 스타일이 독립적지 않은 반면 XML은 데이터를 저장하는 XML 문서, 데이터의 구조를 정의하는 DTD 또는 XML 스키마, XML이 표시되는 방식에 대한 규정이며 데이터의 스타일을 지정하는 CSS 또는 XSL 문서들이 서로 독립적이다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 관련 연구로 XML에 관한 이론적 배경을 알아보고, 3장에서는 Track&Trace 시스템의 DTD를 설계한다. 4장에서는 XML 스키마와 XSL을 설계하며, 마지막 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련연구

2.1 Well-Formed XML 문서

Well-Formed XML 문서란 DTD나 XML 스키마 없이 XML 규약에 정의된 문서 양식에 따라 작성된 문서로 유효한 문서이다[2].

또한, XML 문서를 만들기 위한 최소한의 규칙으로 만들어진 문서이기 때문에 이 규칙이 맞으면 브라우저에 트리 형태로 보여지며, 맞지 않으면 XML 문서를 브라우저에서도 볼 수 없다[3].

2.2 Valid XML 문서

본 논문은 한국전자통신연구원(ETRI) 위탁과제(2001년) 지원을 받아 수행된 것입니다.

웹 상의 다양하고 광범위한 자료 교환에 중요한 역할을 하고 있는 XML 문서를 작성하려면 DTD나 XML 스키마가 필요하다. 특히, B2B와 같은 전자상거래에서는 XML 문서의 구조적인 정보가 더욱 중요하다. 이러한 DTD나 XML 스키마를 정의함으로써 개발자들은 쉽게 문서 안에 특정한 구조 방식으로 배치된 특정 정보를 뽑아서 처리하는 애플리케이션을 만들 수 있게 된다.

① DTD(Document Type Definition)

DTD는 문서 안의 데이터들에 대한 구조를 표현하는 규칙들의 집합으로 XML 문서에서 사용하고 있는 용어들을 정의(요소, 속성 등)하고, 이 용어들 간의 관계(포함, 순서, 수량, 필수 유무 등)를 정의하여 XML 문서의 자체적인 포맷을 정의하는 것이다[4].

DTD는 외부의 파일로서 참조되거나 문서 내부에 포함될 수 있으며, 우선 순위는 문서 내부부터 참조된다.

② XML 스키마

스키마 모델로는 W3C XML 스키마와 Microsoft XML 스키마가 있지만[8], Track&Trace 시스템에 대한 설계로는 W3C XML 스키마를 사용한다.

③ Namespace

Namespace는 요소의 이름과 속성 이름의 집합이며 이들이 중복되는 것을 해결하기 위해 사용한다. 이러한 요소의 이름이나 속성 이름 앞에 접두어를 붙여서 각각의 요소 이름과 속성 이름을 구별한다.

또한, Namespace에는 URI(Uniform Resource Identifier)가 사용되는데, 이것은 사용자 정의에 의한 URI와 스키마 등의 요소가 정의되어 있는 것을 재사용하기 위한 URI 두 가지가 있다.

2.3 스타일 시트

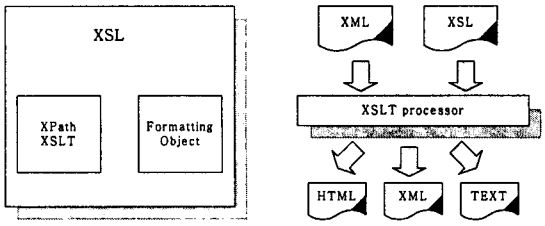
XML 문서는 문서의 구조적인 정보만을 저장하고 문서의 스타일에 대한 정보를 가지고 있지 않으므로 브라우저를 통해 보기 위해서는 스타일시트가 필요하다.이러한 스타일시트에는 CSS와 XSL이 있다. CSS와 XSL은 여러 가지 차이점이 존재하지만 최근 W3C에서는 이 둘을 함께 사용하는 방법에 대한 표준안을 마련하고 있다[6].

① CSS(Cascading Style Sheets)

HTML 뿐만 아니라 XML에서도 사용 가능하며, 문서를 웹 환경에 출력하기 위한 스타일 정보를 정의한 것으로 글꼴, 색상, 행간 등의 정보를 표현할 수 있다. 또한, 페이지 기능, 표, 위치 지정 등의 더 정밀한 스타일 정보 표현이 가능하다[5].

② XSL(eXtensible Stylesheet Language)

XSL과 XSLT 프로세서의 구조는 그림1과 같다. XSLT는 XML 문서를 다른 형태의 문서로 변환해 주는 웹 언어이며[5], XML 문서 내용의 일부만을 추출, 추가할 수 있다. 또한, FO(Formating Object)는 페이징, 레이아웃 등 XML 문서의 외양에 관한 스타일 정보를 부여하는 단계이다. 그러나, 아직까지 대부분의 브라우저나 프로세서에서는 XSLT만을 지원하고 있다[3].

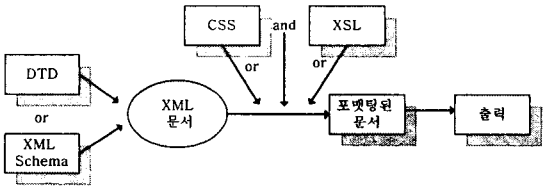


[그림1] XSL과 XSLT 프로세서의 구조

3. Track&Trace 시스템의 DTD 설계

3.1 개요

전체적인 XML 문서의 출력 과정은 그림3과 같다. DTD나 XML 스키마에 따라 작성된 문서를 Valid 문서라 하며, DTD나 XML 스키마 없이 작성된 문서를 Well-Formed 문서라고 한다. 또한, CSS나 XSL을 이용해서 XML 문서의 스타일을 지정해 화면에 출력한다.



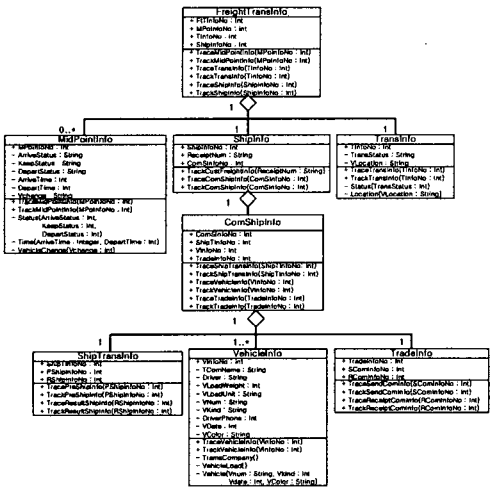
[그림3] XML 문서의 출력 과정

3.2 Well-Formed XML 문서 설계

UML로 모델링 된 Track&Trace 시스템의 클래스 다이어그램은 그림2와 같다. 클래스 다이어그램에서 최상위 클래스는 FreightTransInfo이며 자식 클래스는 MidPointInfo, ShipInfo, TransInfo로 구성된다. 그리고, ShipInfo의 자식 클래스는 ComShipInfo이며, ComShipInfo의 자식 클래스는 ShipTransInfo, VehicleInfo, TradeInfo로 각각의 클래스들은 연관 관계를

갖고 있다.

이러한 각각의 클래스들은 멤버 데이터와 데이터 타입에 따른 컨텐츠를 갖는다.



[그림2] Track&Trace 클래스 다이어그램

XML의 기본 설계인 Track&Trace의 Well-Formed XML 문서는 그림4와 같다. 클래스 다이어그램의 연관 관계에 따라 각각의 클래스들은 Well-Formed XML 문서에서 요소(엘리먼트)가 되며, 멤버 데이터들은 각각의 요소에 대한 자식 요소와 컨텐츠를 나타낸다.

```
<?xml version="1.0" encoding="euc-kr"?>
<FreightTransInfo>
    <MidPointInfo>
            <ArriveStatus>도착</ArriveStatus>
            <KeepStatus>보관중</KeepStatus>
            <DepartStatus>출발중</DepartStatus>
            <ArriveTime>13시 40분</ArriveTime>
            <DepartTime>13시 50분</DepartTime>
            <Vchange>무</Vchange>
    </MidPointInfo>
    <ShipInfo>
            <ComShipInfo>
                    <ShipTransInfo>

                    </ShipTransInfo>
                    <VehicleInfo>
                            <TComName>현대백배</TComName>
                            <Driver insurance="유">김인수</Driver>
                            <VLoadWeight>100</VLoadWeight>
                            <VLoadUnit>kg</VLoadUnit>
                            <VNum>부산 80가 5122</VNum>
                            <VKind>트럭</VKind>
                            <DriverPhone>011-211-4066</DriverPhone>
                            <DriverPhone>042-248-2278</DriverPhone>
                            <VDate>1997-08</VDate>
                            <VColor>blue</VColor>
                    </VehicleInfo>
                    <TradeInfo>

                    </TradeInfo>
            </ComShipInfo>
    </ShipInfo>
    <TransInfo>
            <TransStatus>출발대기중</TransStatus>
            <VLocation>서울</VLocation>
    </TransInfo>
</FreightTransInfo>
```

[그림4] Well-Formed XML 문서 설계

3.3 DTD 설계

DTD로 문서의 구조를 정의함으로써 문서의 오류를 줄일 수 있으며 정보 교환이 용이하다[4]. XML 문서 내에서 문서의 검증을 쉽게 하는 외부 DTD의 선언은 그림5와 같으며, DTD의 종류를 선언하는 명령인 SYSTEM을 사용하였다.

```
<!DOCTYPE FreightTransInfo SYSTEM "FreightTransInfo.dtd">
```

[그림5] XML 문서내에서의 외부 DTD 선언

Track&Trace 시스템의 DTD 설계는 그림6과 같다. 클래스 다이어그램의 멤버 데이터에서 지정한 데이터 형식이나 다중성(0. 1...*) 등을 이용하여 구조를 정의한다.

그러나, 순서나 발생 횟수(MidPointInfo:0번 또는 한 번 이상) 등의 지정이 제한적이며, 모든 요소들의 내용을 문자열(#PCDATA)로 나타내므로 숫자 등의 데이터 타입이 제한적(string, name tokens, ID 등)이다. 또한, Namespace를 사용할 수 없으므로 여러 DTD로부터 태그를 가져올 수 없다.

그림6은 다른 요소들은 속성을 가지고 있지 않지만, Driver는 보험 가입 여부(Insurance)는 반드시 필요, 성별(Sex)은 선택 사항인 요소의 특성을 나타내는 속성을 보이고 있다.

```
<?xml version="1.0" encoding="euc-kr"?>
<!ELEMENT FreightTransInfo (MidPointInfo+,ShipInfo,TransInfo)>
<!ELEMENT MidPointInfo (ArriveStatus,KeepStatus,DepartStatus,ArriveTime,DepartTime,Vchange+)>
<!ELEMENT ArriveStatus (#PCDATA)>
<!ELEMENT KeepStatus (#PCDATA)>
<!ELEMENT DepartStatus (#PCDATA)>
<!ELEMENT ArriveTime (#PCDATA)>
<!ELEMENT DepartTime (#PCDATA)>
<!ELEMENT Vchange (#PCDATA)>
<!ELEMENT ShipInfo (ComShipInfo)>
<!ELEMENT ComShipInfo (ShipTransInfo,VehicleInfo+,TradeInfo)>
<!ELEMENT ShipTransInfo (..)>
<!ELEMENT VehicleInfo (TComName,Driver,VLoadWeight,VLoadUnit,VNum,VKind,DriverPhone+,VDate+,VColor+)>
<!ELEMENT TComName (#PCDATA)>
<!ELEMENT Driver (#PCDATA)>
<!ATTLIST Driver
        Insurance CDATA #REQUIRED
        Sex CDATA #IMPLIED
>
<!ELEMENT VLoadWeight (#PCDATA)>
<!ELEMENT VLoadUnit (#PCDATA)>
<!ELEMENT VNum (#PCDATA)>
<!ELEMENT VKind (#PCDATA)>
<!ELEMENT DriverPhone (#PCDATA)>
<!ELEMENT VDate (#PCDATA)>
<!ELEMENT VColor (#PCDATA)>
<!ELEMENT TradeInfo (..)>
<!ELEMENT TransInfo (TransStatus,VLocation)>
<!ELEMENT TransStatus (#PCDATA)>
<!ELEMENT VLocation (#PCDATA)>
```

[그림6] Track&Trace DTD 설계

4. Track&Trace 시스템의 XML 스키마와 XSL의 설계

4.1 XML 스키마 설계

XML 스키마의 선언은 그림7과 같으며, "http://www.tnt.org"를 기본 Namespace로 가정하고 xsi를 사용하기 위한 Namespace를 선언하였다. 또한, 스키마 문서와의 연결을 위해 그 요소인 스키마 Location으로 스키마가 선언되어 있는 것을 불러와서 사용할 수 있는 확장성을 가지고 있다.

```
<FreightTransInfo xmlns ="http://www.tnt.org"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.tnt.org/FreightTransInfo.xsd">
```

[그림7] Namespace를 이용한 XML 스키마 선언

Track&Trace 시스템의 XML 스키마 설계는 그림8과 같다. XML 스키마로 전자상거래 등과 같은 복잡한 내용 모델과 상세하고 견고한 내용 모델을 만들 수 있다.

다양한 데이터 타입을 지원하는 XML 스키마에서는 DTD에서 숫자, 문자 모두를 문자로 표현한 것을 보안하기 위해 simpleType을 이용하여 사용자가 타입을 정의하며, complexType 아래 자식 요소들을 선언하고 순서나 발생 횟수 등을 설정한다.

그림8에서는 VKind(차량종류) 타입의 이름을 VK로 하고 simpleType을 이용해서 카고, 탑차 등으로 사용자가 정의한 후 하나를 선택하도록 하였다.

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.tnt.org"
            xmlns="http://www.tnt.org" elementFormDefault="qualified">
                .
                .
<xsd:simpleType name="VK">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="카고" />
         <xsd:enumeration value="탑차" />
         <xsd:enumeration value="냉동탑차" />
         <xsd:enumeration value="트랙" />
         <xsd:enumeration value="트레일러" />
         <xsd:enumeration value="시티벤" />
         <xsd:enumeration value="윙바디" />
         <xsd:enumeration value="보통캐리어" />
         <xsd:enumeration value="탱크로리" />
         <xsd:enumeration value="기타" />
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="DP">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d(3)-d(3)-d(4)|d(3)-d(4)-d(4)" />
   </xsd:restriction>
</xsd:simpleType>
                .
                .
<xsd:element name="shipinfo">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="ComShipInfo" maxOccurs="1" >
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="ShipTransInfo" type="xsd:string" maxOccurs="1" />
                  <xsd:element name="VehicleInfo"  minOccurs="1" maxOccurs="unbounded" />
                     <xsd:complexType>
                        <xsd:sequence>
                           <xsd:element name="TComName" type="xsd:string" maxOccurs="1"  />
                           <xsd:element name="Driver" type="xsd:string" maxOccurs="1" />
                           <xsd:element name="VLoadWeight" type="xsd:int" maxOccurs="1" />
                           <xsd:element name="VLoadUnit" type="VU" maxOccurs="1" />
                           <xsd:element name="VNum" type="VN" maxOccurs="1" />
                           <xsd:element name="VKind" type="VK" maxOccurs="1" />
                           <xsd:element name="DriverPhone" type="DP" maxOccurs="1" />
                           <xsd:element name="VDate" type="xsd:gYearMonth" maxOccurs="1" />
                           <xsd:element name="VColor" type="xsd:string" maxOccurs="1" />
                        </xsd:sequence>
                           <xsd:attribute name="Insurance" use="required">
                              <xsd:simpleType>
                                 <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="유" />
                                    <xsd:enumeration value="무" />
                                 </xsd:restriction>
                              </xsd:simpleType>
                           </xsd:attribute>
                           <xsd:attribute name="Sex" use="optional">
                              <xsd:simpleType>
                                 <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="남" />
                                    <xsd:enumeration value="여" />
                                 </xsd:restriction>
                              </xsd:simpleType>
                           </xsd:attribute>
                     </xsd:complexType>
                        .
                        .
```

[그림8] Track&Trace XML 스키마 설계

4.2 XSL의 설계

XML 문서내에서의 XSL 선언은 그림9와 같다. 이것은 XML 문서와 스타일시트 문서를 연결하기 위한 것으로 어떤 스타일시트(CSS 또는 XSL)를 사용할 것인지 선언해 주는 것으로 타입에 XSL 사용을 선언하였다.

```
<?xml-stylesheet type="text/xsl" href="FreightTransInfo.xsl"?>
```

[그림9] XML 문서내에서의 XSL 선언

XSL 문서내에서 XML 문서를 HTML이나 다른 형태의 문서(html,text,xml)로 변환[6] 가능한 XSLT 선언은 그림10과 같다. output을 이용해 변환 결과 파일을 html로, 컨텐츠가 한글이므로 인코딩 방식을 euc-kr로 지정해 주었으며, indent로 공백 줄바꿈 등을 이용해서 보기 좋게 작성되도록 선언하였다.

```
<?xml version="1.0" encoding="euc-kr"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" version="1.0"encoding="euc-kr"indent="yes"/>
<xsl:template match="/">
```

[그림 10] XSL 문서내에서의 XSLT 선언

XSL 문서내에서 XML 문서 외에 HTML에도 적용이 가능한 CSS는 그림11과 같으며, 여러 개의 선택자의 속성이 모두 같으므로 콤마로 분리 즉 그룹화 하여 한번에 지정하였다.

그림11에서는 전체 body에 해당하는 부분의 글씨체, 색상을 지정하고, 제목에 해당하는 부분의 문장정렬, 글자 두께, 글자 크기를 지정하였다. 또한, tr의 th에는 색상과 글자 두께, 글자 크기를, td에는 문장정렬, 글자 크기, 글자 두께를 지정하였다.

```
<html>
<head>
<style type="text/css">
body {font-family:궁서;color:navy;}
p {text-align:center;font-weight:bold;font-size:20pt;}
th {color:red;font-weight:solid;font-size:14pt;}
td {text-align:center;font-size:10pt;font-weight:bold;}
</style>
</head>
```

[그림11] XSL 문서내에서의 CSS

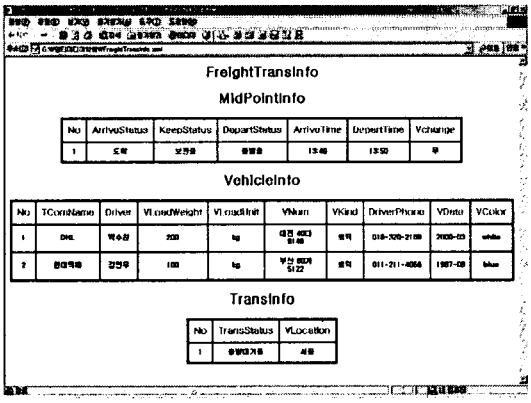
XML 문서만을 위해 사용되는 XSL 설계의 예는 그림12와 같다. 테이블을 생성하여 th의 요소들을 입력하고 xsl:for-each를 사용하여 지정한 형식에 대해 반복해서 화면에 출력하고, td의 xsl:value-of에서 각각의 값을 읽어온다.

또한, Driver에서 이름을 내림차순으로 정렬하였으며, No 요소를 추가하여 인덱스를 주었으며 이것은 삭제, 재배치도 가능하며, XML 문서의 내용을 임의의 순서로 전개할 수 있다.

```
<body>
<p>FreightTransInfo</p>
<p>MidPointInfo</p>
<table align="center" border="3" cellpadding="10" cellspacing="0" bordercolor="navy">
<tr>
<th>No</th><th>ArriveStatus</th><th>KeepStatus</th><th>DepartStatus</th>
<th>ArriveTime</th><th>DepartTime</th><th>Vchange</th>
</tr>
<xsl:for-each select="FreightTransInfo/MidPointInfo">
<tr>
<td><xsl:value-of select="position()"/></td>
<td><xsl:value-of select="ArriveStatus"/></td>
<td><xsl:value-of select="KeepStatus"/></td>
<td><xsl:value-of select="DepartStatus"/></td>
<td><xsl:value-of select="ArriveTime"/></td>
<td><xsl:value-of select="DepartTime"/></td>
<td><xsl:value-of select="Vchange"/></td>
</tr>
</xsl:for-each>
</table>
  .
  .
  .
<p>VehicleInfo</p>
<table align="center" border="3" cellpadding="10" cellspacing="0" bordercolor="navy">
<tr>
<th>No</th><th>TComName</th><th>Driver</th><th>VLoadWeight</th><th>VLoadUnit</th>
<th>VNum</th><th>VKind</th><th>DriverPhone</th><th>VDate</th><th>VColor</th>
</tr>
<xsl:for-each select="FreightTransInfo/ShipInfo/ComShipInfo/VehicleInfo">
<xsl:sort select="Driver" order="descending"/>
<tr>
<td><xsl:value-of select="position()"/></td>
<td><xsl:value-of select="TComName"/></td>
<td><xsl:value-of select="Driver"/></td>
<td><xsl:value-of select="VLoadWeight"/></td>
<td><xsl:value-of select="VLoadUnit"/></td>
<td><xsl:value-of select="VNum"/></td>
<td><xsl:value-of select="VKind"/></td>
<td><xsl:value-of select="DriverPhone"/></td>
<td><xsl:value-of select="VDate"/></td>
<td><xsl:value-of select="VColor"/></td>
</tr>
</xsl:for-each>
</table>
  .
  .
  .
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

[그림12] XSL내에서의 XSL 설계

지금까지의 XML 스키마와 XSL을 토대로 설계된 XML을 브라우저에 출력한 결과는 그림13과 같다.



[그림13] XML 실행 결과

5. 결론 및 향후 연구 방향

본 논문은 UML로 모델링 된 Track&Trace 시스템을 DTD와 XML 스키마, CSS와 XSL로의 변환에 관한 연구이다.

특히, Track&Trace에서 어떤 환경이든 상관없이 사용할 수 있는 데이터를 XML로 설계함으로써 확장성과 유연성을 가질 것이다. 또한, 문서의 구조를 정확하고 정밀하게 XML 스키마로 설계함으로써 안전하고 신뢰성을 바탕으로 한 고객과의 거래가 될 것이다.

앞으로 Web Service의 표준이라 할 수 있는 SOAP 및 WSDL을 이용한 변환 작업이 계속될 것이다.

[참고문헌]

[1] 하수철, "UML 기반 물류 Tracking 모델 개발에 관한 연구", 최종 보고서, 한국전자통신연구원, 2002

[2] 허준회, 이민우, 김종민, 최한석, "Jump To XML Web Programming", 정보게이트, 2001

[3] 권혜윤,"XML&Java For Web Master", 성안당, 2002

[4] 이종호, "XML과 전자상거래", 정보문화사, 2001

[5] 임순범, "XSL-FO를 이용한 XML문서의 스타일 정의, 정보처리학회지, 제8권 제3호, pp.38-46, 2001.5

[6] 고성임, "쉽고 재미있는 XML 홈페이지 만들기", 신화전산기획, 2000

[7] 최영근, 정계동, "XML How To Programming", 피어슨 어듀케이션 코리아, 2002

[8] 신행자, 박경환, "WBT에서의 강의 컨텐츠 구조를 위한 XML Schema 설계", 한국정보과학회 춘계학술대회논문집, 제28권 제1호, pp. 691-693, 2001