

e-비즈니스 환경 하에서의 도메인 모델 방식 기반의 전자문서 변환시스템 설계 및 구현

심언섭^o 김진우* 백두권*
고려대학교 정보통신대학원 정보통신공학과^o, 고려대학교 컴퓨터학과*
eonseob@korea.com^o, {pkm311, baik}@software.korea.ac.kr*

The Design and Implementation of Domain Model Based Electronic Document Translation System in e-business Environment

EonSeob Shim^o, JinWoo Kim*, DooKwon Baik*
Computer Engineering of Computer Information and Technology, Korea University^o,
Computer Science and Engineering, Korea University

요 약

e-Business 환경이 다양화되고 활성화되면서, 전자적으로 교환되는 문서의 종류와 형태가 다양해지고 있다. 다양한 문서 형태를 지원해야 하는 요구사항이 대두되고 이를 충족하기 위해 하나 이상의 문서 변환 시스템을 도입하여 사용하고 있다. 기존의 문서 변환 시스템은 하나 또는 두 가지의 문서 형태만을 지원함으로써, 동시에 다양한 문서 형태를 지원할 수 없는 단점이 존재하였다. 또한 단일 형태의 문서 변환만을 지원하고 있어 문서 형태별 변환 기능을 지원할 수 없고 파싱된 문서 요소와 비즈니스 로직이 혼합되어 순차적으로 처리됨으로써, 처리가 완료되기 전까지 하나의 트랜잭션으로 처리되고 있다. 변환 절차가 하나의 트랜잭션 내에서 처리됨으로써, 정보의 재사용성이나 효율적으로 시스템 자원을 사용할 수 없었다. 이러한 문제를 해결하기 위해 본 논문에서는 문서 형태별 비즈니스 로직을 포함하는 전자문서 변환 시스템을 도메인 모델방식 개념을 기반으로 설계, 구현함으로써 개선 시스템을 제안하고자 한다. 본 논문은 제안 시스템을 통해 기능별로 모듈화가 가능하게 되고 트랜잭션도 단계별로 분리하여, 처리 복잡성을 해결할 뿐 아니라, SOA(Service Oriented Architecture)기반의 진보된 개념을 쉽게 적용 가능한 확장성을 갖추는데 그 의의가 있다.

1. 서 론

정보화의 발달로 자료의 전자적 교환이 대두되면서, 단순 텍스트 형태에서 EDI(Electronic Data Interchange)를 거쳐 최근의 ebXML(electronic Business eXtensible Markup Language)까지 다양한 전자 문서 표준이 제정되어 사용되고 있다. e-비즈니스 환경이 다양화 되면서 상호 교환되는 문서와 비즈니스 실체가 또한 다양해지고 있다. 이미 e-비즈니스 환경에서 문서 간 형태 변환시스템은 존재하지만, 기존 시스템은 단일 형태 간의 문서 변환만을 지원하고 있어 상호 교환되는 문서 형태가 증가됨에 따라 하나 이상의 변환 시스템이 요구되고 있다[1-6].

기존 변환 시스템은 문서 파싱과 비즈니스 로직 그리고 문서 변환 기능이 혼합되어 하나의 트랜잭션 내에서 순차적으로 처리됨으로써, 데이터의 재사용이 어렵고 분산 처리에 어려움이 있다. 이로써 한정된 시스템 자원을 효율적으로 활용할 수 없게 되어 시간적, 자원적 손실이 발생하였다. 또한 기존 변환 시스템은 하나 또는 두 가지 형태의 문서 변환만을 지원하는 구조로 설계되었기 때문에 여러 형태의 문서 변환 기능을 제공하기 위해서는 추가로 변환 시스템을 도입해야 하는 문제점이 있다.

이러한 문제를 해결하기 위해, 본 논문에서는 도메인 모

델 기반의 구조로 다양한 형태 변환을 지원할 수 있는 문서 변환 시스템을 설계하였다.

제안 시스템에서 사용하는 데이터는 파싱된 결과를 공통의 데이터 모델 형태로 분리하도록 설계하였다. 단위 프로세스는 컴포넌트(Component) 기반으로 설계하여 생성된 데이터 모델의 재사용성과 분산처리 가능하도록 하였다. 이로써 자원의 효율성을 증가시키고 시스템에 대한 자원 낭비를 줄일 수 있다.

논문의 구성은 다음과 같다. 먼저 제 2장에서는 기존 변환시스템의 개요와 문제점을 제시하고 이를 해결하기 위한 방안에 대해 설명하겠다. 그리고 제 3장에서는 제안 시스템의 구조와 특징 및 장점에 대해 기술하겠다. 제 4장에서는 시나리오를 따라 성능평가를 수행하여 평가 결과를 분석하여 제안모델의 우수성에 대해 설명하겠다. 마지막으로 제 5장은 결론과 향후 연구계획에 대해 기술한다.

2. 관련연구

2.1 기존 변환 시스템 구조

2.1.1 단일 어플리케이션 방식

단일 프로세스 내에서 변환 동작에 필요한 모든 이벤트를 처리하는 방식으로 전자문서에 대한 데이터 파싱부터 문서의 적합성 및 유효성을 검증하는 검증 단계, 최종 결과

형태로의 변환에 이르는 과정을 하나의 싱글 어플리케이션으로 구현한다.

이 방식은 문서 종류가 추가되거나 또는 문서 형태가 수정될 경우 반드시 코어 프로그램을 수정해야하며, 동일 기능을 수행하는 모듈이 여러 프로그램에 존재함으로 인해 비효율적인 프로세스 운영으로 자원의 낭비가 발생한다 [6-8].

2.1.2 스크립트 기반 어플리케이션 방식

소스 코드를 통해 변환 과정을 처리하는 단일 어플리케이션 방식과 달리 스크립트 기반 방식은 변환에 관련된 정보를 별도의 스크립트로 분리하여 처리하도록 동작한다. 즉, 코어 프로그램은 공통으로 사용될 수 있도록 구성하고, 문서 형태에 따라 데이터 파싱과 필드 매핑 과정은 스크립트를 통해 비즈니스 로직을 결합하여 트랜잭션을 처리하도록 하였다.

이로써 새로운 문서를 추가하거나 수정할 경우, 코어 프로그램을 수정하는 대신 스크립트 정보만을 수정함으로 인해 단일 어플리케이션 방식보다 효율적인 운영 방식을 취하고 있다[6,7].

2.2 기존 시스템의 한계와 문제점

기존 변환 시스템을 분석한 결과, 문서형태에 종속적인 비효율적인 내부구조로 인해 하나 또는 두 가지 문서 형태만을 지원함으로 다른 형태의 문서변환에 있어 이미 구현된 기능을 재활용 할 수 없고, 새롭게 구축해야 한다는 점에서 투자대비 효율성 측면의 문제점이 존재한다. 즉, 동일 기능을 수행하는 모듈이 각 문서 형태별로 존재함으로 인해 불필요한 자원 낭비가 발생하고, 하나의 트랜잭션내에서 스크립트기반 처리부분을 분리할 수 없는 구조적 문제로 인해 시스템의 가용성 측면의 위험과 유지보수의 복잡성 문제가 존재한다.

아래 그림 1은 기존 스크립트 방식은 단일 어플리케이션 보다는 효율적인 구조이나, 파싱된 결과와 비즈니스 로직을 결합하여 트랜잭션을 처리함으로써 분산처리가 불가능한 문제점과 새로운 문서 형태가 추가될 경우 별도의 변환 시스템을 도입해야 하는 문제점을 보여준다.

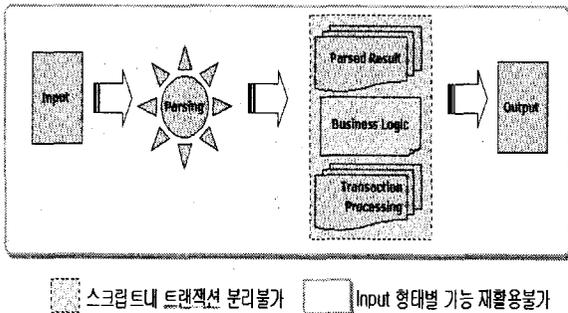


그림 1. 기존 시스템의 문제점

3. 제안 모델

2장에서 언급한 문제점을 해결하기위해 본 논문에서는 도메인 모델 기반의 변환 시스템을 제안하였다. 제안 시스템은 대상 업무를 파악하여 추상화 과정을 거쳐 새로운 형태의 모델로 변환하는 구조로 설계되었다. 즉, 도메인 모델은 비즈니스 영역에서 사용되는 객체를 판별하고, 객체가 제공해야 하는 기능을 추출하며, 각 객체간의 관계를 정립하는 과정을 거친다.

3.1 제안 시스템 아키텍처

제안 시스템은 사실형태문서(UDF : User Define Format), EDI, DTD(Document Type Definition) 기반 XML, Schema 기반 XML 파일과 데이터베이스를 지원할 수 있다.

제안 시스템은 크게 Converter, DOM(Document Object Model) 생성기, 비즈니스 처리기, 객체 생성기, Mapping 관리자, 제어기 등으로 구성된다. 제안 시스템은 입력된 문서 형태에 따라 문서 타입과 규격 정보에 따라 해당 파서(Parser) 모듈과 검증(Validating) 모듈을 호출한다.

각 모듈에 대한 설명은 다음 표 1과 같고 시스템 아키텍처는 다음 그림 2와 같다.

결과 파일 형태는 크게 텍스트 기반과 데이터베이스 기반으로 구분할 수 있다. 텍스트 기반은 결과물이 파일 형태로 생성되며, 데이터베이스는 결과가 데이터베이스 시스템에 저장된다.

표 1 제안 시스템 구성 요소

구분	설명
컨버터 (Converter)	문서형태별 데이터 파싱과 형태에 대한 적합성 및 유효성을 검증
DOM 생성기	컨버팅된 데이터를 XML 노드 트리 형태의 공통 데이터 셋 생성
비즈니스 처리기	공통 데이터 셋을 입력으로 하여 비즈니스 로직을 처리하며, XSL(XML Stylesheet Language)을 이용하여 동작함
객체 생성기	해당 XSL Loading 후 객체 생성
매핑 관리자	해당 프로세스의 호출 및 생성된 객체의 재사용 처리
제어기	Transe Manager 등

목표 문서 형태를 위해 데이터 객체를 생성하며, 생성된 객체는 결과 문서 포맷으로 변환되는 독립된 프로세스가 호출되어 결과물을 생성하게 된다.

모든 프로세스의 인터페이스는 객체로 단일화되었으며, 변환하고자 하는 프로세스의 경우 Xalan을 활용하였다. 데이터베이스 변환의 경우, 영속성 관리에 대한 안정성과 효율성 측면에서 우수한 하이버네이트(Hibernate)[9]를 적용하였다.

3.2 제안시스템 특징

제안 시스템의 특징은 기존 변환 시스템의 단점을 극복하

기 위해 트랜잭션을 단계별로 분리한 것이다. 파싱과 검증 과정을 포함하는 컨버팅(Converting) 단계, 컨버팅된 정보를 노드 트리 형태의 공통 데이터 셋(DataSet)으로 치환하는 데이터 모델 생성(Dom Creator) 단계, 데이터 모델 생성 단계를 거쳐 비즈니스 로직을 적용하고 최종 데이터 객체 모델로 변환(Object Creator)하는 단계를 거쳐게 된다. 그리고 최종 데이터 객체 모델을 기반으로 결과물을 생성하는데, 독립적 분산처리가 가능하도록 해당 프로세스를 호출한다.

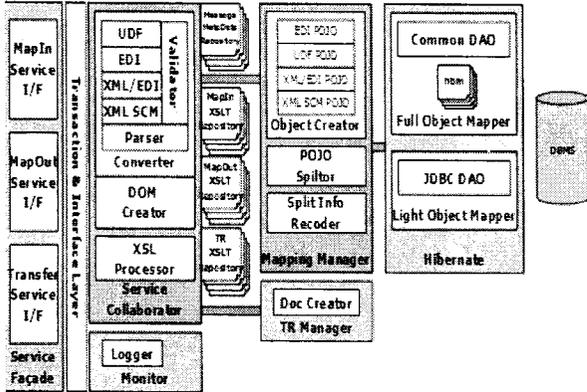


그림 2. 제안 모델의 시스템 아키텍처

그림 2는 기존 스크립트 방식의 문제를 해결하기 위해 존재하는 문서형태별 파서(Parser)와 유효성 검증(Validator)을 하나의 컨버터(Converter) 기능으로 추상화하여 새로운 형태의 모델로 변환하는 구조로 설계됨을 보여주고 있으며, 기능별로 해당 프로세스를 컴포넌트화가 가능하여 분산 구조를 지원할 수 있음을 보여주는 제안모델의 시스템 아키텍처이다. 제안 시스템은 노드 트리 형태의 공통 데이터 셋을 정의함으로 인해 서로 다른 문서 형태의 데이터를 단일화된 구조로 표현할 수 있다.

공통 데이터 셋의 구조는 다음 그림 4와 같다. 즉, 입력되는 문서 형태가 추가될 경우 해당 문서 형태에 대한 파서만을 추가하여 공통 데이터 셋으로 생성하기만 하면 다른 모듈에서는 그대로 재활용이 가능하다. 트랜잭션을 단계별로 구분하고 프로세스를 컴포넌트화 함으로 인해 기존 시스템의 단점을 해결할 수 있다.

제안 시스템 구현 시, 객체 지향 기술의 일반화 와 ORM(Object Relational Mapping)[10]과 같은 안정적인 프레임워크를 기반으로 하였다. 데이터베이스 시스템으로 변환 시 매핑 하는 쿼리문을 실행 코드 없이 실행할 수 있도록 오픈 소스를 이용하여 구현하였다. 이로써 대다수 데이터베이스 시스템에 대한 이식성 문제를 해결할 수 있었으며, 안정성 및 개발 생산성 그리고 성능적인 측면에서 우수하다고 볼 수 있다.

5. 구현 및 성능평가

본 장에서는 제안 시스템의 우수성을 증명하기 위해 몇

가지 변환 케이스에 대해 기존 변환 시스템과 성능평가를 하고 그 결과를 비교 분석하였다. 성능평가를 실행한 환경은 아래 표 2와 같다.

표 2 성능평가 환경

항목	내용
운영 체제	Windows XP Professional (Service Pack 2)
CPU	640 MHz, 1 CPU
메모리	2GHz, Memory
데이터베이스	Oracle 9i

```
UNB+UNOA:1+BCTOC010:KL+KLHETCS:KL+060601:0004+20060601000010+UHH+00
0001+COARRI:D:95B:UH:ITG12 BGM+98+20060601000010+9 FTX+OSI+D:TDT+20
++1+SYL:172:20+++VZBK5:103: HELNLOHELENE LOC+11+KRPUS:139:6+DTM:132
:200605312100:203 DTM:133:200606010900:203 NAD+CA+SYL:160:20+EQD+CN+
CA XU6794046+2200:102:5++3+5 DTM:203:200605312358:203 LOC+9+CWTR0:139
:6 LOC+147+0250308:5 LOC+150+4E210101 MEA+WT+G+KGM:15700 NAD+CF+SYL
:160:20 CNT+1:1 UNT+000017+000001 UHH+000002+COARRI:D:95B:UH:ITG12 B
GM+270+20060601000010+9 FTX+OSI+L TDT+20++1+KMD:172:20+++HOUV:103:
MMSA03MAMITSA LOC+9+KRPUS:139:6 DTM:132:200605310800:203 DTM:133:20
0606010200:203 NAD+CA+HAS:160:20+EQD+CN+KMTU9263156+4510:102:5+2+5
RFF+BN:0814849 DTM:203:200606010003:203 LOC+11+SGSIR:139:6 LOC+147+0
120886:5 LOC+150+4C300101 MEA+WT+G+KGM:14766 SEL+05242+CA NAD+CF+KH
D:160:20 CNT+1:1 UNT+000019+000002 UHH+000003+COARRI:D:95B:UH:ITG12
BGM+270+20060601000010+9 FTX+OSI+L TDT+20++1+CMA:172:20+++ABM3:10
3:VDUK02CHA CGM
UKRAINE LOC+9+KRPUS:139:6 DTM:132:200605310800:203 DTM:133:200606010
200:203 NAD+CA+CMA:160:20+EQD+CN+TRIU9843020+4500:102:5+2+5 DTM+203
:200606010003:203 LOC+11+TRIST:139:6 LOC+147+0340784:5 LOC+150+2848
0402 MEA+WT+G+KGM:08550 SEL+3198741+CA NAD+CF+CMA:160:20 CNT+1:1 UNT
+000018+000003 UN2+000003+20060601000010
```

그림 3. EDI 문서 예

```
<document>
...
<segment name="LOC" group="GROUP1">
  <single>
    <element>11</element>
  </single>
  <composite>
    <element>KRPUS</element>
    <element>139</element>
    <element>6</element>
  </composite>
</segment>
<segment name="DTM" group="GROUP1">
  <composite>
    <element>132</element>
    <element>200605312100</element>
    <element>203</element>
  </composite>
</segment>
...
</document>
```

그림 4. 공통 데이터셋 구조의 예

시나리오는 EDI 문서를 데이터베이스로 변환하는 것으로, 기존 변환 시스템은 스크립트 기반의 변환 시스템을 채택 하였다. 위의 그림 3은 성능평가에 쓰인 EDI 문서에 대한 예제를 보여주며, EDI의 내용 중 블록 지정된 내용으로 예 를 구체화하였다. 그림 5,6,7은 문서와 데이터베이스 간의 매핑을 위한 ORM 과정의 관계성을 보여준다.

표 3 성능평가 결과

(단위 : 초)

항목	제안 시스템	기존 시스템
처리건수	1,345	1,345
평균 처리 시간	8.17249071	218.468892
최대 처리 시간	939	8087
최소 처리 시간	0	4
최대 루핑 문서 처리 시간(924)	939	6469

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan"
  exclude-result-prefixes="xalan"
  version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="document">
    <xsl:for-each select="/segment[@name='EQD' and @group='GROUP3']">
      <unloading-port>
        <xsl:value-of select="preceding-sibling::
          segment[@name='LOC' and @group='GROUP1']/single[1]/
          element[1][text()='11']/composite[1]/element[1]" />
      </unloading-port>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

그림 5. XSL Script의 예

그림 5는 XPath를 이용하여 공통 데이터 셋에 접근후 ORM Mapper 내 Object의 멤버와의 매핑을 위한 XSL Script를 보여주며, 블록 지정된 부분은 그림 3의 EDI 예제의 블록 지정된 부분에 대한 표현이다.

그림 6의 블록 지정된 부분은 Hibernate내에서 Object의 멤버와 실제 DataBase의 Column과의 매핑에 필요한 Meta 정보를 나타내며, 아래 그림 7은 Java로 구현된 DB Column에 매핑 될 실제 Object Class의 예를 보여준다.

```
<hibernate-mapping>
  <class
    name="com.kinet.model.coarri.ImportDetail"
    table="TEST_IMPORT_DETAIL">
    <property
      name="unloadingPort"
      type="java.lang.String"
      column="UNLOADING_PORT"
      length="15">
    </property>
  </class>
</hibernate-mapping>
```

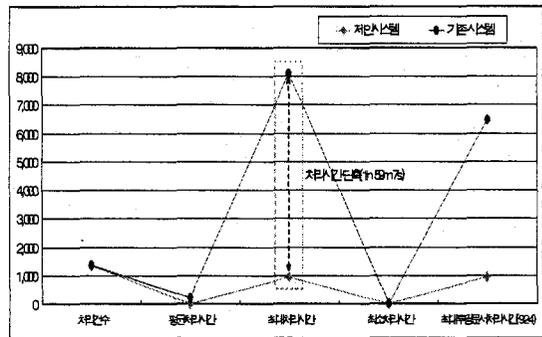
그림 6. Hibernate Mapping File

성능평가를 위한 전자문서의 표본 데이터는 약 1,345건의 문서를 대상으로 하여, 총 10번의 반복 테스트를 통한 결과에 대한 평균치를 측정하였다.

성능평가 결과는 아래 표 3과 같다. 결과에서 보는 것과 같이 본 논문에서 제안한 시스템을 통한 변환이 기존 Script 방식의 변환 시스템보다 성능 면에서 훨씬 우수함을 볼 수 있다. 결론적으로, 문서 크기가 클수록, 문서 내 반복되는 케이스가 많을수록 제안 시스템의 성능이 월등하다는 결과를 도출할 수 있다.

```
public class ImportDetail implements Serializable {
  private String unloadingPort;
  ...
  public String getUnloadingPort() {
    return this.unloadingPort;
  }
  public void setUnloadingPort(String unloadingPort) {
    this.unloadingPort = unloadingPort;
  }
  ...
}
```

그림 7. Hibernate of the Object Class



6. 결론 및 향후 연구

e-비즈니스 환경이 다양화 되면서 상호 교환되는 문서와 비즈니스 실체가 또한 다양화해지고 있다. 이미 e-비즈니스 환경에서 문서 간 형태 변환시스템은 존재하지만, 기존 시스템은 단일 형태 간의 문서 변환만을 지원하고 있어 상호 교환되는 문서 형태가 증가함에 따라 하나 이상의 변환 시스템이 요구되고 있다.

기존 변환 시스템은 하나의 트랜잭션 내에서 순차적으로 처리됨으로 인해, 데이터의 재사용이 어렵고 분산 처리가 불가능하다. 또한 기존 변환 시스템은 하나 또는 두 가지 형태의 문서 변환만을 지원하는 구조로 설계되었기 때문에 여러 형태의 문서 변환 기능을 제공하기 위해서는 추가로 변환 시스템을 도입해야 하는 문제점이 있다. 이러한 문제를 해결하기 위해, 본 논문에서는 도메인 모델 기반의 구조로 다양한 형태 변환을 지원할 수 있는 문서 변환 시스템을 설계하였다. 제안 시스템에서 사용하는 데이터는 파싱된 결과를 공통의 데이터 모델 형태로 분리하도록 설계하였다. 단위 프로세스는 컴포넌트 기반으로 설계하여 생성된 데이터 모델의 재사용성과 분산처리가 가능하도록 하였다. 이로써 자원의 효율성을 증가시키고 시스템에 대한 자원 낭비를 줄일 수 있다. 객체지향의 재사용성, 유지보수의 편리성을 확대 가능케 하고, 처리시간을 단축시켰으며 컴포넌트 기반, SOA(Service Oriented Architecture)[11]와 같은 진보된 개념을 쉽게 적용 가능한 확장성을 갖추게 되었다. 제안 시스템의 우수성을 증명하기 위해 본 논문에서는 기존 변환 시스템과의 테스트를 하여 성능 면에서 우수함을 증명하였다.

향후 연구로는 다양한 문서 형태에 적용하여 성능면에서

제안 시스템의 우수성을 증명하고, 병렬 처리가 가능한 구조로 확장하여 처리 시간을 단축시킬 수 있도록 하겠다.

7. 참고문헌

- [1] 한국전산원, "EDI 환경에서의 웹기반 e-비즈니스 프레임워크 표준 수용에 대한 연구", CAIV-RER-04012, 2004.9
- [2] 산업자원부, 한국전자거래진흥원, e-Biz 표준화 백서, KIEC-063, 18-27, pp74-197, 2004.1
- [3] UN/CEFACT, "Core Component Technical Specification Version 2.01, Part 8 of the ebXML Framework", 2003.11
- [4] UN/CEFACT, "XML Naming and Design Rules Draft 1.0", 2004.8
- [5] <http://www.xmlmedi-group.org/xmlmedigroup/guide.htm>, "Guidelines for using XML for Electronic Data Interchange"
- [6] 조민양, 백두권, "XMA:XSLT 기법을 이용한 매핑 에이전트, 정보과학회, 제10권, 제1호, pp771-774, 2003.5
- [7] KyeongRim, Ahn, JaeHong. Ahn, BaekKi. Moon, WhaYeon. Lee, JinWook, Chung, The Converting /Transfer Agent for E-Commerce, APIS-3, 2004.1
- [8] 서성보, 이용준, 황재각, 류근호, "e-Logistics 시스템의 메시지 상호운용성", 정보과학회 논문지, 제11권, 제5호, 2005.10
- [9] <http://www.hibernate.org>
- [10] Mark L. Fussell, "Foundations of Object Relational Mapping", <http://www.chimu.com/publications/index.html>, 1997
- [11] M.W.A Steen, et al, "Service-Oriented Software-Oriented Software System Engineering: Challenges and Practice", 2004