Journal of Korea Multimedia Society Vol. 10, No. 5, May 2007(pp. 652-667)

XML 데이터의 입력을 위한 XForms 인터페이스의 자동 생성

송기섭[†], 이경호[††]

요 약

현재 다양한 영역에서 XML의 사용이 확대되며 XML 데이터의 입력을 위한 사용자 인터페이스에 대해 관심이 높아지고 있다. XML 데이터의 입력을 위한 사용자 인터페이스를 자동으로 생성하는 기존의 연구는 다양한 사용자의 환경에 대한 대응이나 복잡한 문서를 위한 인터페이스 제공이라는 면에서 미흡하다. 본 논문에서는 XML 데이터를 편리하게 입력할 수 있도록 XML 스키마로부터 XForms에 기반한 인터페이스를 생성하는 방법을 제안한다. 제안된 방법은 세 단계로 구성된다. 먼저 XML 스키마 정의를 받아들여 파싱하여 스키마 구조 트리를 생성한다. 그리고 제안된 규칙에 따라 XForms 코드를 생성한다. 제안된 규칙은 복합 타입의 스키마 구조를 위한 XForms 코드 생성 규칙과 단순 타입을 위한 XForms 컨트롤 매핑 규칙을 포함한다. 그리고 생성된 XForms 코드는 마지막 단계에서 템플릿으로 제작된 다른 XML 마크업 언어로된 문서에 내재된다. 본 논문에서는 제안된 방법을 평가하기 위해 다양한 도메인의 스키마를 대상으로 실험한 결과 다양한 복합 형식과 단순 형식으로부터 XForms 인터페이스를 생성함을 확인할 수 있었다.

Automatic Generation of XForms Interfaces for XML Data Input

Kisub Song[†] and Kyong-Ho Lee[††]

ABSTRACT

With the wide spread of XML data in various fields, the interest in developing convenient user interfaces for inputting XML data is increasing. Previous works on the automated generation of user interfaces have given insufficient supports for various user environments and inconvenient editing tools for complex documents. In this paper, we presents an approach to generate XForms based user interfaces from XML schemas for inputting XML data. The proposed method consists of three steps: parsing XML schemas, generating XForms codes, and embedding it into host language documents. Specifically, to generate XForms codes, the rules for generating XForms codes, which support complex types and map simple types to appropriate input controls, are proposed. Experimental results with various kind of schemas show that the proposed method can generate XForms interfaces successfully.

Key words: XML Schema(XML 스키마), XForms(XForms), User Interfaces(사용자 인터페이스)

1. 서 론

XML이 다양한 분야에서 데이터 표현을 위한 표준으로 널리 사용됨에 따라 XML 데이터를 편리하게 입력할 수 있는 사용자 친화적인 인터페이스가 필요하다. 일반적으로 사용자 인터페이스를 제작하는 작

※ 교신저자(Corresponding Author) : 이경호, 주소 : 서울 서대문구 신촌동 134 연세대학교(120-749), 전화 : 02)2123-3878, FAX : 02)365-2579, E-mail : khlee@cs.yonsei.ac.kr
접수일 : 2007년 1월 7일, 완료일 : 2007년 5월 21일
[†] 정회원, 연세대학교 컴퓨터과학과
(E-mail : kssong@icl.yonsei.ac.kr)
[††] 종신회원, 연세대학교 컴퓨터과학과
※ 본 지식재산권은 정통부 및 정보통신연구진흥원의 지원을 받아 수행된 연구결과임 (06-정책-118, 적응형 모바일 웹 서비스 기술)

업은 프로그램 제작 과정에서 큰 비중을 차지하기 때문에 이를 자동으로 생성한다면 작업량을 감소시킬 수 있다. 따라서 본 논문에서는 XML 데이터의 입력을 위한 사용자 인터페이스를 자동으로 생성하는 방법을 제안한다.

XML 데이터 및 문서를 편집하기 위한 사용자 인터페이스는 이질적인 다양한 시스템에서 사용 가능하여야 하고, 복잡한 구조의 XML 문서를 편집할 수 있어야 하고, XML 스키마[1]의 다양한 데이터 타입에 적합한 입력 컨트롤을 제공하여야 하고, 입력된 XML 문서를 검증할 수 있어야 한다. 기존의 XHTML/WML에 기반한 인터페이스[3,4]는 XML 기반이기 때문에 XSLT[5] 등을 사용하여 쉽게 생성할 수 있다는 장점이 있지만 지원하는 입력 컨트롤이 제한적이며 복잡한 구조의 XML 문서를 편집할 수 없으며 스키마 검증을 지원하기 어렵다. Java Swing 기반의 사용자 인터페이스[6]는 다양한 컨트롤을 제공할 수 있지만 아직까지는 다양한 플랫폼에서 사용하기에는 무리가 따른다.

XForms[2][7]는 XML 기반의 사용자 인터페이스 개발을 위한 마크업 언어로서 다양한 디바이스에 대한 인터페이스 지원을 목표로 하기 때문에 유비쿼터스 컴퓨팅 환경에 적합하다. 또한 기본적으로 XML 데이터를 전송하기 위한 목적으로 만들어졌기 때문에 XML 스키마 검증이 가능하며 복잡한 구조의 XML 문서 편집을 지원한다. 그 밖에 사용자의 입력을 돕는 다양한 컨트롤을 지원한다. 따라서 XML 문서 편집을 위한 최적의 사용자 인터페이스를 제공한다.

한편 XML 스키마로부터 XForms 인터페이스를 생성하는 기존 연구는 XML Schema 또는 DTD를 입력으로 받아들여 XHTML, Java Swing 등의 인터페이스를 생성하는 방법을 제안한다. 하지만 이들은 다양한 시스템에 대한 고려가 되지 않았으며, 복잡한 XML 데이터를 편집하기 힘든 구조로 되어있다.

따라서 본 논문에서는 XML 스키마로부터 XForms 기반의 GUI를 생성하는 효율적인 방법을 제안한다. 제안된 방법에서는 읽어 들인 XML 스키마를 파싱하여 스키마 구조 트리를 생성하고, 이를 분석하여 XForms 코드를 생성한다. 그리고 이를 다른 XML 언어로 된 문서에 적재(embed)하여 인터페이스를 완성한다.

제안된 방법의 성능을 평가하기 위해 다양한 스키마를 대상으로 실험하였다. 실험에 사용된 스키마는 XML 스키마의 복합 형식과 다양한 데이터 타입을 기술하고 있기 때문에 복잡한 인터페이스를 생성할 수 있어야 한다. 실험 결과, 대부분의 스키마에서 적절한 인터페이스를 생성해낼 수 있었다. 일부의 성공하지 못한 사례에 대해서는 실험 결과와 분석에서 기술한다.

본 논문의 구성은 다음과 같다. 2절에서는 XML 스키마를 효과적으로 표현할 수 있는 문서모델을 정의하고, XForms의 구조에 대하여 언급하고, 기존의 XML 데이터 편집에 관한 연구를 간략히 소개한다. 3절에서는 본 논문에서 제안한 인터페이스 생성 방법에 대하여 기술한다. 4절에서는 제안된 방법을 평가하기 위한 실험 결과와 분석을 기술한다. 끝으로 5절에서는 결론 및 향후연구 방향을 기술한다.

2. 문서 모델, XForms, 그리고 관련 연구

본 절에서는 XML 스키마를 표현하기 위한 문서 모델을 제안한다. 또한 XForms의 구조 및 특징을 간략히 소개한 후, 구조화된 문서의 편집 인터페이스를 자동 생성하는 기존 연구를 간략히 소개하고 문제점을 기술한다.

2.1 문서 모델

본 논문에서는 XML 스키마를 표현하기 위하여 트리구조에 기반한 문서 모델을 제안하며 제안된 문서 모델에 따라 표현된 트리구조를 스키마 구조 트리라고 정의한다. 스키마 구조 트리를 구성하는 각각의 노드는 엘리먼트(element), 애트리뷰트(attribute) 그리고 선택 및 순서 연산을 표현하는 구조 지시자(structure indicator)로 이루어진다. 구조 지시자 노드는 all, sequence, 그리고 choice의 3가지가 있으며 엘리먼트 노드 또는 다른 구조 지시자 노드를 자식 노드로 가질 수 있다. 엘리먼트 노드는 구조 지시자 노드 또는 애트리뷰트 노드를 자식으로 가진다. 애트리뷰트 노드는 어떠한 자식도 가지지 않는다.

1) 일반적으로 XML 스키마는 DTD[1]와 XML Schema[2]를 포함한다. 본 논문에서는 XML Schema를 대상으로 한다.

2) 본 논문은 XForms 1.0을 기준으로 하였으며, XForms 1.0의 한계와 현재 개발 진행 중인 XForms 1.1의 가능성에 대해서 간단히 언급한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://icl.yonsei.ac.kr/kssong"
    xmlns="http://icl.yonsei.ac.kr/kssong">
  <xsd:element name="Config">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Database" minOccurs="1" maxOccurs="unbounded" />
        <xsd:element ref="Manager" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Database">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice>
          <xsd:element ref="Driver" />
          <xsd:element name="Name" type="xsd:string" />
        </xsd:choice>
        <xsd:element ref="Mapping" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:ID" use="required"/>
      <xsd:attribute name="engine" type="xsd:string" use="optional" default="generic" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Mapping">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="href" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Driver">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="url" type="xsd:string" use="required" />
      <xsd:attribute name="class" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
 <xsd:element name="Manager">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="parameters">
          <xsd:attribute name="name" type="xsd:string" default="local"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="parameters">
    <xsd:sequence>
      <xsd:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Parameter">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

그림 1. XML 스키마 예제.

자손으로 다른 엘리먼트 노드를 가지지 않는 엘리먼트 노드나 애트리뷰트 노드는 단순 데이터 타입을 자식으로 가질 수 있다. 자손으로 다른 엘리먼트 노드를 가지는 엘리먼트 노드는 복합 데이터 타입을 자손으로 가지지만 mixed 타입일 경우 문자열을 자식 노드로 가질 수 있다. 애트리뷰트 노드를 제외한 모든 노드는 빈도 지시자를 가질 수 있다. 또한 any 타입 노드는 다른 엘리먼트 노드의 자식이 될 수 있으며 다른 노드를 자식 노드로 갖지 않는다.

만일 스키마에 <extension>을 통해 확장된 복합 형식 또는 <redefine>을 통해 재정의 된 복합 형식이 존재할 경우 스키마 구조 트리에서는 이들 형식의 최종적으로 변형된 형태가 포함된다. 또한 다른 네임스페이스를 가진 외부의 스키마의 데이터 형식이 import 되어 사용되었다면 이 외부 스키마의 데이터 형식 역시 스키마 구조 트리 내부에 포함된다.

그림 2는 그림 1의 XML 스키마를 제안된 문서 모델로 표현한 결과이다.

2.2 XForms 개요

XForms는 구조화된 데이터를 처리하기 힘들다는 단점을 가진 HTML 폼을 대체하기 위해 만들어졌다. 이를 위해 데이터 입출력 및 처리 시 XML을 사용하고, 다양한 인코딩을 지원하며, 기존의 복잡한 스크립팅 언어로만 구현이 가능했던 기능을 쉽게 구현할 수 있게 하였다. 따라서 XML 문서를 편집하기 위한 사용자 인터페이스로서 적합한 환경을 제공한다.

XForms 인터페이스의 구조는 크게 모델과 컨트롤로 분리되어 있다. 모델은 데이터와 전송 정보를 프레젠테이션으로부터 독립시켜 XML 데이터를 처리할 수 있게 하며, 폼의 재사용성, 기기 독립성을

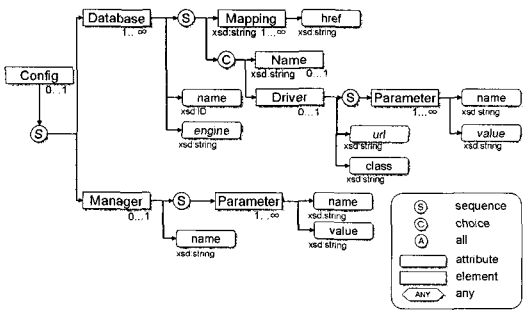


그림 2. 그림 1의 XML 스키마를 스키마 구조 트리로 표현한 결과.

구현한다. 모델은 XML 인스턴스(instance), 바인딩(binding), 그리고 전송부(submission)로 이루어진다. 그 구성은 그림 3과 같다.

XML 인스턴스는 초기에 주어지는 XML 데이터로써, 사용자의 수정을 위한 초기 값이 주어지거나 사용자의 입력을 위해 비어있는 골격 XML 문서의 형태로 제공된다. 사용자의 입력을 통해 얻어지는 최종 결과물은 이 XML 인스턴스가 수정된 형태이다. XML 인스턴스는 데이터의 구조를 사용자의 입력을 통해 얻기 위해 데이터를 임시로 저장하는 변수 용도로 사용되기도 한다. 본 논문에서는 데이터 반복 구조와 선택 구조를 구현하기 위해 이 특성을 사용한다.

전송부는 편집된 XML 데이터의 출력부를 결정한다. 완성된 문서를 직렬화하고, 프로토콜로 전송하기 위해 필요한 옵션 정보를 담고 있다. 본 논문에서는 XML 문서를 편집하기 위한 XForms 컨트롤과 모델을 생성하는 것을 목표로 하고 있으므로 전송부는 정의하지 않는다.

바인딩 정보는 인스턴스의 노드와 컨트롤을 연결하고 인스턴스의 제약조건을 정의한다. 또한 특정 인스턴스 노드에 XML 스키마의 데이터 타입을 매핑하여 입력 데이터를 한정짓는다.

XForms는 사용자의 데이터 입력 및 편집의 편의를 위한 다양한 컨트롤을 제공한다. 특히 XML 스키마에서 제공하는 기본 데이터 타입에 최적화된 데이

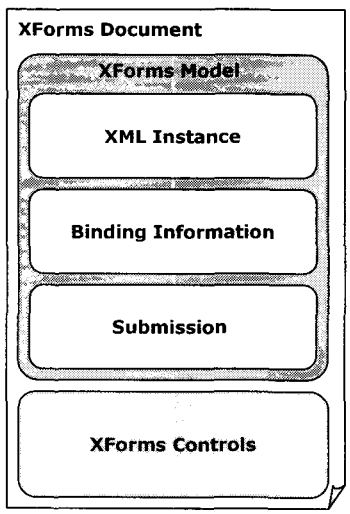


그림 3. XForms 의 구조

터 입력 컨트롤을 제공하며, 데이터의 타입에 따라 다중 선택, 범위 선택, 텍스트 입력 등의 다양한 사용자 편의를 위한 데이터 입력 폼을 지원한다. 또한 XML 인스턴스의 반복 구조, 선택 구조 등을 제어할 수 있는 컨트롤을 제공한다. XForms 컨트롤은 모델로부터 분리되며 장치에 따라 형태가 달라진다.

2.3 관련 연구

구조화된 문서를 입력하기 위해서 사용자 인터페이스를 생성하는 기존 연구는 표 1과 같다. XML 스키마로부터 GUI를 제공하기 위한 기존 연구는 대부분 제한된 환경에서 동작하는 플랫폼을 기반으로 하거나, 목표 플랫폼을 지정하지 않았기 때문에 범용 인터페이스로서의 가치가 높지 않다. 또한 많은 연구가 변환 과정에서 개발자의 개입을 요구하고 있기 때문에 GUI 개발시에 추가 비용이 들게 된다.

본 논문에서는 XML 스키마로부터 XML 문서 편집을 위한 사용자 인터페이스를 생성해 내는 구체적 알고리즘 및 프레임워크를 제안한다. 본 논문에서는 기존 연구의 한계를 극복하기 위해, 다양한 환경에서 지원 가능한 표준 인터페이스로 제안된 XForms를 목표 GUI로 한다. 그리고 사용자 인터페이스를 생성하는 과정을 완전 자동화하여 사용자의 개입을 최소화한다. 이 과정에서 다양한 사용자의 환경에 따른 사용자 인터페이스에 관한 연구[16-19]들을 참조한다.

3. XForms 기반 사용자 인터페이스 생성

본 절에서는 스키마 구조 트리로부터 XForms 인터페이스를 생성하는 방법을 자세히 기술한다. 인터페이스 생성 과정의 전체적인 구성을 보이고, 가장 중요한 부분인 스키마 구조에 따른 XForms 인터페이스 생성 규칙과 스키마의 단순 데이터 타입에 맞는 XForms 입력을 선택하는 규칙을 정의한다.

본 논문에서는 XML 편집을 위한 인터페이스를 생성하기 위해 XML 스키마를 이용한다. XML 스키마에 유효한 문서를 처리하기 위한 XForms 인터페이스를 생성하는 과정은 그림 4와 같다.

표 1. 구조화된 문서를 입력하기 위한 인터페이스를 생성하는 기존 연구

저자	연도	특징	대상	목표GUI	방식
Jelinek 와 Slavik [8]	2004	애플리케이션 소스코드에 GUI에 관한 부가적 정보를 추가하여 GUI를 프로그램 코드로부터 분리하는 한편, GUI 제작을 용이하게 함.	Annotated Source Code	GUI	사용자 개입
Lay 와 Luttringhaus-Kappel [9]	2004	XML Schema로부터 Java Swing 기반의 GUI를 자동 생성. Java Swing의 컴퍼넌트를 XML 형식의 데이터로 직렬화한 JavaBeans를 사용함.	XML Schema	Java Swing	자동
Radha 등 [10]	2005	XML Schema로부터 XHTML, WML, Java와 같은 서로 다른 렌더링 언어로 된 GUI를 자동으로 생성.	XML Schema	GUI	자동
Kuo 등 [11]	2005	XML 어휘에 기반한 XML 문서를 편집하기 위한 HTML 기반사용자 인터페이스를 생성하는 동적 컴퍼넌트인 Forms-XML 제안.	XML Schema	HTML	사용자 개입
Lee 와 Kim [12]	2005	DTD로부터 XForms와 XSL 기반의 인터페이스 생성. 복잡한 구조를 정의할 수 있는 XML Schema에는 적용이 어려우며, XSL을 사용하므로 인터페이스를 실행할 수 있는 환경이 제한됨.	DTD	XForms+ XSL	자동
Kassoff 등 [13]	2003	GUIDD를 이용하여 WSDL로부터 HTML 기반 인터페이스 생성. 웹서비스 제공자가 GUIDD를 미리 제공해야 함.	WSDL	HTML	사용자 개입
Steele 등 [14]	2005	WSDL문서를 읽어 들여 XForms 및 VoiceXML에 기반한 인터페이스 생성. 구체적 알고리즘 제시되지 않음.	WSDL	XForms	자동
Garvey 와 French [15]	2003	스키마로부터 XForms 기반 사용자 인터페이스 생성. 복잡 형식에 대한 처리를 다루지 않음.	XML Schema	XForms	자동

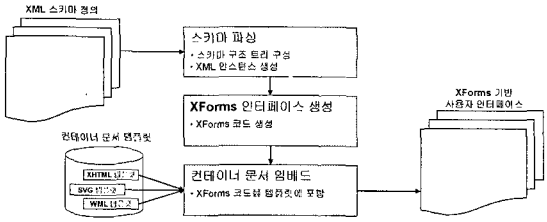


그림 4. XForms 인터페이스의 생성 과정.

최초로 XML 스키마 정의를 받아들여 스키마 파싱을 하는데, 이를 통해 스키마의 복잡한 문법을 그림 2와 같은 형태의 스키마 구조트리로 단순화한다. 또한 편집하게 될 XML 인스턴스 문서의 초기 형태를 이 과정에서 생성한다. 이 과정에서 생성된 XML 인스턴스의 초기 형태는 데이터는 삽입되지 않은 형태에 해당한다.

이렇게 생성된 스키마 구조 트리와 XML 인스턴스를 기반으로 XForms 인터페이스를 생성하는 과정은 복합 타입(complex type)의 스키마 구조와 단순 타입(simple type)을 위한 XForms 코드 생성을 포함한다. 구체적인 규칙은 3.1절과 3.2절에서 다룬다.

생성된 XForms는 다른 XML 마크업 언어(host language[3])에 내재되어 사용될 수 있도록 설계되어 있다. 따라서 생성된 XForms 인터페이스를 화면에 출력하기 위해 다른 문서에 포함시켜야 하는데, 이를 위해 미리 제작된 다른 XML 어휘 기반의 템플릿(template)을 이용한다. 이 템플릿은 XHTML, SVG[20] 등의 다양한 언어 기반으로 제작되어 있으며, 전 단계에서 생성된 XForms 코드를 삽입하여 완성된 형태의 사용자 인터페이스가 완성된다. 그림 5는 템플릿의 예이다. 이와 같은 형태로 제작된 템플릿에 XForms 모델과 컨트롤을 내재시켜 XForms가 실행될 수 있는 문서를 만든다.

3.1 스키마 구조 트리로부터 XForms 인터페이스 생성

XML 스키마는 다양한 스키마 구성요소를 사용하여 복잡한 XML 데이터를 정의한다. 이러한 구성요소를 통해 엘리먼트의 순서, 반복, 선택 구조 등을 정의한다. 다양한 형태의 구조를 생성하는 인터페이스를 생성하기 위해 각각의 스키마 구성요소를 위한 XForms 인터페이스 생성 규칙이 필요하다.

그림 6은 스키마로부터 인터페이스를 생성하기 위한 재귀 알고리즘을 간략하게 기술한 것이다. 제안된 알고리즘은 스키마 구조 트리의 루트 엘리먼트 노드와 함께 호출되어 XForms의 모델과 컨트롤을 생성한다. 스키마 구조 노드에 대한 처리 규칙은 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8"?>
<xhtml:html
    xmlns:ev="http://www.w3.org/2001/xml-events"
    xmlns:tns0="http://icl.yonsei.ac.kr/kssong"
    xmlns:xforms="http://www.w3.org/2002/xforms"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xhtml:head>
    <xhtml:title>[Title]</xhtml:title>

    [XForms Model]

    <xhtml:link href="xforms.css"
        rel="stylesheet" />
  </xhtml:head>
  <xhtml:body>

    [XForms Control]

  </xhtml:body>
</xhtml:html>
```

(a) XHTML 템플릿의 예.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg:svg xmlns:svg="http://www.w3.org/2000/svg"
    xmlns:tns0="http://icl.yonsei.ac.kr/kssong"
    xmlns:xforms="http://www.w3.org/2002/xforms"
    xmlns:ev="http://www.w3.org/2001/xml-events"
    width="700px" height="600px"
    viewBox="0 0 700 600">
  <svg:defs>
    <svg:polygon id="bullet" fill="#007138"
        points="-30,-30, -10,-10, -20,10" />

      [XForms Model]

  </svg:defs>
  <svg:title>[Title]</svg:title>
  <svg:g>
    <svg:foreignObject x="80" y="150"
        width="250" height="40">

      [XForms Control]

    </svg:foreignObject>
  </svg:g>
</svg:svg>
```

(b) SVG 템플릿의 예.

그림 5. 템플릿의 예제.

3) http://www.w3.org/TR/2006/REC-xforms-20060314/index-all.html#def-host-language

```
Algorithm: TraceSchema <스키마 구조 추적 알고리즘>

input: 스키마 구조 트리의 노드 N

if (N이 Element일 경우)                                          가
        if (N의 minOccurs < N의 maxOccurs)                       바
                Repetation을 위한 Control 생성;

        if (N이 SimpleType을 가지고 있을 경우)
                Element를 위한 Control 생성;

        if(N이 mixed 타입일 경우)                                 사
                mixed 타입을 위한 Control 생성;

        for-each (N의 모든 자식노드에 대해)
                TraceSchema(N의 모든 자식 노드); // 재귀 호출

else if (N이 Attribute일 경우)                                   나
        Attribute의 Control 생성;

else if (N이 ComplexType 이거나 Group일 경우)
        TraceSchema(N의 자식 노드); // 재귀 호출

else if (N이 sequence 노드일 경우)                               다
        for-each (N의 모든 자식 노드에 대해)
                TraceSchema(N의 모든 자식 노드); // 재귀 호출

else if (N이 choice 노드일 경우)                                 라
        choice를 위한 XForms 컨트롤 생성;
        for-each (N의 모든 자식 노드에 대해)
                TraceSchema(N의 모든 자식 노드); // 재귀 호출

else if (N이 all 노드일 경우)                                    마
        all을 위한 XForms 컨트롤 생성;
        for-each (N의 모든 자식 노드에 대해)
                TraceSchema(N의 모든 자식 노드); // 재귀 호출

else if (N이 any 노드일 경우)                                    아
        if (N이 namespace와 SchemaDefinition을 갖고 있을 경우)
                TraceSchema(첨부된 스키마); // 재귀 호출
```

그림 6. 스키마 구조 트리 추적 알고리즘.

가. 엘리먼트

엘리먼트 노드는 자식 노드로 구조 지시자 노드 또는 애트리뷰트 노드를 가진다. 따라서 이에 대한 처리를 위해 자식 노드 각각을 대상으로 스키마 구조 트리 추적 규칙을 재귀적으로 적용한다. 자손 노드 중 다른 엘리먼트 노드가 포함되지 않은 엘리먼트 노드는 단순 데이터를 값으로 가질 수 있는데, 사용자가 이 데이터를 편집할 수 있게 하는 컨트롤이 생성되어야 한다. 이때는 4.2절의 단순 데이터 타입 적용 규칙을 적용한다. 만일 엘리먼트 노드가 mixed 타입으로 정의되어 있을 경우, 그에 해당하는 컨트롤 생성 규칙을 적용해야 한다. 이는 [사]에서 다룬다. 엘리먼트 노드에 빈도 지시자 정의에 의한 반복 구조가 정의되어있을 경우 이를 위한 컨트롤 생성은 [바]에서 다룬다.

나. 애트리뷰트

애트리뷰트 노드는 자식 노드를 가지지 않으며, 단순 데이터 타입의 값을 가진다. 따라서 이에 대한 컨트롤 생성을 필요로 한다. 이를 위해 단순 데이터 타입을 가지는 엘리먼트 노드와 마찬가지로 4.2절의 단순 데이터 타입 적용 규칙을 사용한다.

다. sequence

sequence 노드는 다른 구조 지시자 노드 또는 엘리먼트 노드를 순차적으로 나열한다. 따라서 하위에 속하는 다른 노드에 대한 규칙을 재귀적으로 적용한다.

라. choice

choice 구조는 하위에 정의된 두 가지 이상의 복합 데이터 타입들 중 한 가지만을 선택적으로 따른다. 선택 구조에서는 사용자에게 입력하기를 원하는 엘리먼트를 선택할 수 있는 컨트롤이 제공되어야 하며, 사용자의 선택에 따라 그에 따른 적합한 형태의 컨트롤이 선택 제공되어야 한다. 따라서 전송하기를 원하는 엘리먼트의 데이터만을 입력할 컨트롤만을 선택할 수 있는 select1 컨트롤을 생성한다. 생성된 XForms 코드의 인스턴스 부분에는 choice하위에 정의된 선택가능 엘리먼트를 모두 인스턴스화한다. 이 경우, 초기 조건에서 인스턴스가 스키마에 유효하지 않지만, 사용자의 선택에 따라 이 선택 가능 엘리먼트들 중 단 하나만이 활성화되므로 전송시에는 스키마에 적법한 인스턴스가 생성된다. 이를 위해 XForms 모델 내에 위치한 bind 엘리먼트의 relevant 속성을 이용하여 select1 컨트롤의 값과 특정 인스턴스 엘리먼트를 바인딩한다.

예를 들어, 그림 7은 그림 2의 choice 노드로부터 생성된 XForms 인터페이스이다. 스키마 구조 트리에 따라 ①의 인스턴스는 Database 엘리먼트 아래에 Driver 또는 Name 엘리먼트 중 하나만이 선택적으로 나타날 수 있다. ②는 사용자의 선택 사항을 임시로 저장하는 변수로 사용하기 위한 인스턴스이다. 사용자의 선택에 따라 ⓒ는 "Driver" 또는 "Name" 값을 가진다. ③은 ⓒ의 값이 "Driver"일 때만 ⓐ가 활성화될 것을 지정하고 있으며, 마찬가지로 ④는 ⓒ의 값이 "Name"일 때만 ⓑ가 활성화될 것을 지정하고 있다. 이를 통해 사용자의 선택을 통해 적절한 엘리먼트를 고를 수 있다.

마. all

all 지시자의 경우 사용자가 엘리먼트의 순서를 변

```
<xforms:instance id="instance_data">
  <Config xmlns="http://icl.yonsei.ac.kr/kssong"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Database engine="generic" name="">
      <Driver class="" url="">
ⓐ       <Parameter name="" value="" />
      </Driver>
ⓑ     <Name/>
      <Mapping href="" />
    </Database>
    <Manager name="local">
      <Parameter name="" value="" />
    </Manager>
  </Config>
</xforms:instance>                                              ①

<xforms:instance id="alternativeManager">
  <xfam:alter>
    <xfam:option id="alter_0">
      <xfam:item option="Driver" />                             ②
    </xfam:option>          ⓒ
  </xfam:alter>
</xforms:instance>

<xforms:bind id="bind_0" required="true"
  nodeset="instance('instance_data')/tns0:Database/tns0:Driver"          ③
  relevant="instance('alternativeManager')/xfam:option[@id='alter_0']/xfam:item/@option='Driver'"/>

<xforms:bind id="bind_1" required="true"
  nodeset="instance('instance_data')/tns0:Database/tns0:Name"            ④
  relevant="instance('alternativeManager')/xfam:option[@id='alter_0']/xfam:item/@option='Name'"/>
```

그림 7. choice로부터 생성된 XForms 인터페이스의 예.

경시킬 수 있는 컨트롤을 제공해야 한다. 하지만 XForms에서는 인스턴스의 엘리먼트 순서를 제어하기 위한 방법을 제공하지 않기 때문에 all 구조를 위한 컨트롤을 생성할 수 없다. 이는 XForms 1.0에서 제공하는 action들의 기능이 제한적이기 때문이며, 이러한 제약은 현재 개발이 진행 중인 XForms 1.1에서 다소 해소될 것으로 보인다.

바. maxOccurs & minOccurs

XML 스키마에서 동일 엘리먼트의 반복은 minOccurs와 maxOccurs 속성을 통해 정의된다. 특정 엘리먼트가 1회 이상 반복될 경우 각각의 엘리먼트의 하위에 위치할 복합 또는 단순 데이터 타입을 위한 컨트롤을 그 개수만큼 생성해야한다. 이때 만일 minOccurs와 maxOccurs의 값이 다를 경우 사용자가 엘리먼트의 개수를 지정할 수 있어야 한다. 따라서 이 경우 사용자가 엘리먼트를 추가/제거할 수 있는 컨트롤을 제공한다.

그림 8은 반복 구조에 대한 XForms 코드의 예이다. ⓐ에서 정의된 모델은 ⓑ의 컨트롤에 바인딩된다. "Database" 노드는 반복적으로 나타나게 되므로 repeat 컨트롤의 내부에 위치되며, 사용자의 입력에 의해 동적으로 개수를 변화시킬 수 있어야 하므로 Delete 버튼(ⓒ)과 Insert 버튼(ⓓ)을 생성한다. minOccurs와 maxOccurs의 값으로 제한된 빈도수 제한은 ①과 같이 XPath 수식을 bind 엘리먼트의 constraint 속성으로 지정하여 설정한다.

사. mixed

mixed 타입에서는 자식노드로 문자열과 엘리먼트가 혼합되어 나타날 수 있다. 이러한 데이터 타입은 엘리먼트들 사이에 위치한 문자열에 대한 편집도 가능해야 한다. 각각의 문자열을 편집하기 위한 컨트롤은 문자열과 매핑하기 위해 XPath 함수 text()를

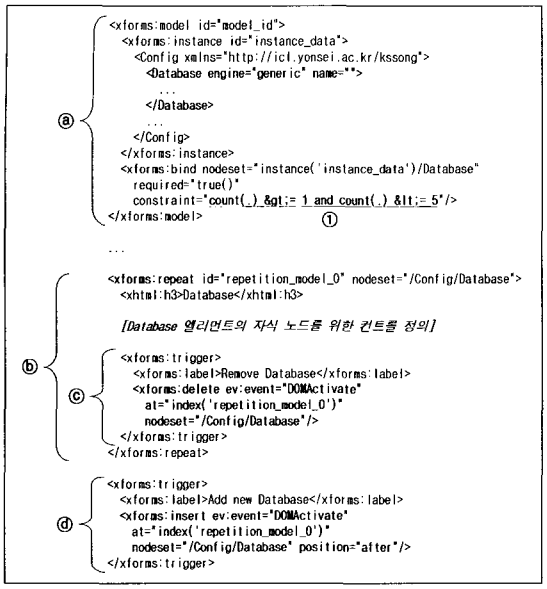


그림 8. 빈도 지시자에 대한 XForms 코드 생성의 예.

이용한다.

그림 9는 mixed가 사용된 엘리먼트에 대한 스키마의 예이다. Database 엘리먼트 하위에 Parameter와 Mapping 엘리먼트가 위치하며, 각 엘리먼트의 전후에 문자열이 위치한다. 이를 위한 XML 인스턴스 및 XForms 코드는 그림 10과 같다.

하지만 mixed 타입 내부의 엘리먼트가 빈도 지시자에 의해 개수의 변동이 가능한 반복 구조로 정의되었을 경우, 이러한 엘리먼트를 XForms의 컨트롤로 편집할 수 없다. 이는 XForms에서 엘리먼트 노드의 빈도수를 제어하는데 사용하는 insert 구문이 인스턴스 내부에 존재하는 노드를 해당 위치에서 복제하도록 설계 되어있기 때문에 복제된 엘리먼트 노드 사이에 문자열을 삽입하는 것이 불가능하기 때문이다.

아. any

와일드카드를 지칭하는 any지시자가 사용된 경우, 스키마에 의해 지정되지 않은 엘리먼트들을 사용

```
<xsd:element name="Driver">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="Parameter" type="xsd:string"/>
      <xsd:element name="Mapping" type="xsd:positiveInteger"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

그림 9. mixed 타입으로 정의된 스키마.

```
<model>
  <instance>
    <tns0:Config>
      <tns0:Database>
        <tns0:Driver>
          String 1<tns0:Parameter/>String 2<tns0:Mapping/>String 3
        </tns0:Driver>
      </tns0:Database>
    </tns0:Config>
  </instance>
  ...
</model>

...

<input ref="tns0:Driver">
  <label>Driver String 1</label>
</input>
<input ref="tns0:Driver/tns0:Parameter">
  <label>Parameter</label>
</input>
<input ref="tns0:Driver/tns0:Parameter/following-sibling::text()[position()=1]">
  <label>Driver String 2</label>
</input>
<input ref="tns0:Driver/tns0:Mapping">
  <label>Mapping</label>
</input>
<input ref="tns0:Driver/tns0:Mapping/following-sibling::text()[position()=1]">
  <label>Driver String 3</label>
</input>
```

그림 10. mixed 타입을 위한 XForms 코드 생성 예.

하여 문서를 확장할 수 있다. 이 경우 형식이 규정되지 않았기 때문에 인터페이스를 정의할 수 없다. 하지만 특정 네임스페이스로 한정 되어있고, 관련 스키마가 존재하는 경우 이에 관련된 인터페이스를 생성할 수 있다.

3.2 단순 데이터 타입과 XForms 컨트롤 매칭

XForms는 사용자의 편의를 위해 다양한 형태의 컨트롤을 제공하고 있는데 이러한 컨트롤은 전달되는 데이터의 형식에 따라 선택된다. 본 논문에서는 표 2와 같이 XML 스키마에서 정의된 데이터 타입에 따라 적절한 XForms 컨트롤을 선택하는 규칙을 제안한다.

표 2. 스키마 단순 데이터 타입과 XForms 컨트롤간의 매핑

	XML 스키마 단순 데이터 타입		XForms 컨트롤
가	내장 데이터 타입		xforms:input
나	restriction에 의한 정의	한정된 길이의 문자열	xforms:input 또는 xforms:textarea
다		enumeration을 통한 선택항목화	xforms:select1
라		유한 구간의 정수형 또는 시간/날짜형	xforms:range
		기타	xforms:input
마	list에 의한 정의	enumeration을 통한 선택항목화	xforms:select
바	union에 의한 정의		다양한 컨트롤 복합

가. 내장 데이터 타입

W3C의 XForms 1.0 권고안은 XML 스키마가 제공하는 44개의 내장 데이터 타입에 대하여 input 컨트롤을 통하여 사용자 친화적인 입력 방식을 제공하도록 한다. 따라서 내장 데이터 타입으로 정의된 데이터의 입력을 위해 XForms의 input 컨트롤을 사용한다.

그림 11은 내장 데이터 타입의 XForms 컨트롤 매핑을 보여주는 예제이다. 이 예제에서 ①에서와 같이 XML 스키마 내장 데이터 타입인 date로 바인딩된 엘리먼트에 대하여 ②와 같이 input 컨트롤을 생성하면 그림의 오른쪽과 같이 날짜 선택에 최적화된 컨트롤이 출력된다.

나. 한정된 길이의 문자열

XML 스키마 내장 데이터 타입인 string 또는 normalizedString 타입을 베이스 타입으로 정의된 데이터 타입은 XForms의 input 컨트롤과 매핑한다. 이때 미리 설정된 임계치 이상의 값을 가진 minLength, maxLength, length의 패싯(facet)을 통해 정의되었을 경우, 긴 문자열의 입력이 예측되므로 XForms의 textarea 컨트롤과 매핑한다.

그림 12는 textarea 컨트롤을 적용한 예이다. 이 그림에서 ⓒ은 string 타입을 기본으로 하여 minLength 패싯을 통해 최소 길이가 100으로 한정된 LongString 타입을 정의하고 있다. 그리고 ⓐ에서 instance의 my:Query 노드는 ⓑ와 같이 bind에 의해 LongString 타입으로 바인딩되었다. 이 경우 ⓓ와 같이 textarea 컨트롤을 생성하여 데이터 입력을 받게 한다.

다. enumeration을 통한 선택항목화

enumeration 패싯을 통해 유도된 사용자 정의 데이터 타입은 스키마에 정의되어있는 항목들 가운데 하나만이 선택될 수 있다. 따라서 사용자가 주어진 항목들 중 하나를 선택하여 전송하는 select1 컨트롤로 변환한다.

그림 13은 select1 컨트롤을 적용한 예이다. 그림에서 ⓐ로 정의된 스키마는 ⓑ와 같이 select1 컨트롤로 변환 가능하며 브라우저 상에는 ⓒ와 같이 출력된다.

라. 유한 구간의 정수형 또는 시간/날짜형

XML 스키마 내장 데이터 타입 중 시간/날짜형(dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth) 또는 정수형(integer, nonPositiveInteger, negativeInteger, long, int, short, byte, nonNegativeInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, positiveInteger)의 내장 데이터 타입을 베이스로 하여

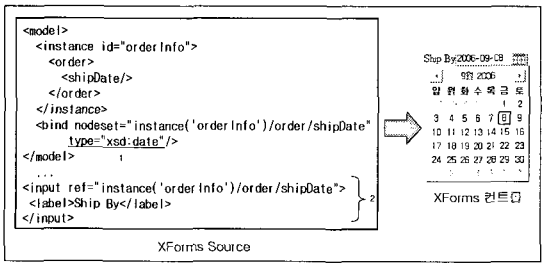


그림 11. XML 스키마 내장 데이터 타입의 XForms 데이터 타입 바인딩의 예.

```
...
<xforms:model id="HotelReservation">
  <xforms:instance id="messages">
    <my:Search>
      <my:Query/> ⓐ
    </my:Search>
  </xforms:instance>
  ...
  <xforms:bind nodeset="my:Query" type="my:LongString"/> ⓑ
  ...
  <xsd:schema targetNamespace="http://icl.yonsei.ac.kr/Hotel/">
    <xsd:simpleType name="LongString">
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="100"/>
      </xsd:restriction>
    </xsd:simpleType>                          ⓒ
  </xsd:schema>
  ...
</xforms:model>
  ...
<xforms:textarea ref="./my:Query">
  <xforms:label>Query: </xforms:label>
</xforms:textarea>                             ⓓ
  ...
```

그림 12. textarea 컨트롤 적용 예.

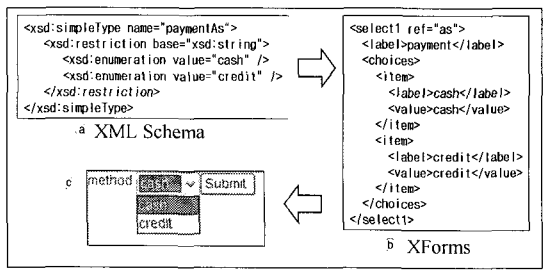


그림 13. select1 컨트롤 적용 예.

maxInclusive, maxExclusive, minExclusive, 그리고 minInclusive의 패싯을 통해 정의된 타입은 range 컨트롤과 매핑한다.

그림 14에서는 XML 스키마에서 maxInclusive와 minInclusive로 정의된 데이터 타입이 range 컨트롤로 변환된 예를 보여준다. 0이상 20이하의 정수를 표현하기 위해 range컨트롤의 start, end, step 속성을 사용하였다.

마. enumeration을 통한 선택항목화한 list

list 타입은 하나 이상의 데이터들을 공백 단위로 구분하여 나열하는 형태로 복수개의 데이터를 포함한다. 그러므로 사용자로부터 복수개의 항목을 선택받아 공백단위로 구분하여 전송하는 XForms의 select 컨트롤과 매핑된다. 하지만 select 컨트롤은 선택가능한 데이터가 유한개 제시되어야 한다. 따라서 list 타입으로 정의된 데이터 타입 중 유도되는 베이스 타입이 restriction 타입을 사용하여 enumeration 패싯으로 유도되었을 경우 select 컨트롤로 변환하고 그렇지 않은 경우 input 컨트롤을 사용한다.

그림 15는 XML 스키마에서 list와 enumeration을 통해 정의된 타입을 위한 XForms 컨트롤을 나타낸

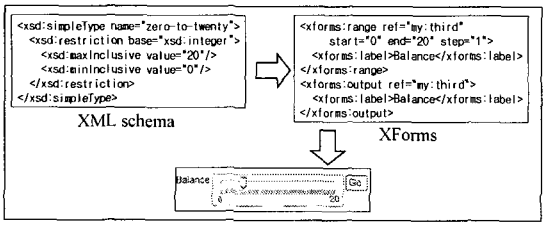


그림 14. range 컨트롤 적용 예.

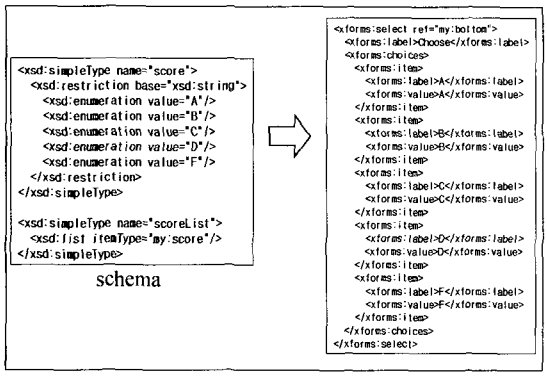


그림 15. select 컨트롤 적용 예.

것이다. 복수 선택 가능한 항목들이 대하여 select 컨트롤을 통해 제공된다.

바. union에 의한 정의

union은 2가지 이상의 단순 데이터 타입의 결합이다. XForms에서는 결합된 데이터 타입 각각에 맞는 컨트롤을 모두 제공하고, 사용자가 입력하려는 값에 따라 적당한 인터페이스를 선택할 수 있게 한다. 예를 들어, xsd:decimal과 enumeration으로 제한된 데이터 타입이 union으로 결합되어있을 경우 각각의 타입에 대한 입력에 적합한 xforms:input과 xforms:select1를 제공하되, 사용자는 이 중 하나를 선택할 수 있다.

그림 16는 union을 위한 컨트롤의 예제이다. union으로 통합된 형태의 컨트롤은 사용자가 선택한 타입에 따라 적합한 컨트롤을 제공한다. 주어진 선택항목에서 사용자가 선택한 타입에 따라 XForms의 switch/case문을 통해 사용자가 입력을 원하는 타입에 맞는 컨트롤을 제공한다. 그림 16에서와 같이 nonNegativeInteger와 enumeration으로 제한된 타입이 합쳐진 두가지 타입이 결합된 타입의 경우 사용자에게 nonNegativeInteger와 enumeration을 선택할 수 있는 컨트롤이 제공된다. 사용자가 nonNegativeInteger를 선택하였을 경우(①) input 컨트롤이 제공되고, enumeration을 선택하였을 경우(②) select1 컨트롤이 제공된다.

4. 실험 결과

본 절에서는 제안된 방법의 성능을 평가하기 위해서 34개의 스키마 파일로부터 인터페이스를 생성하

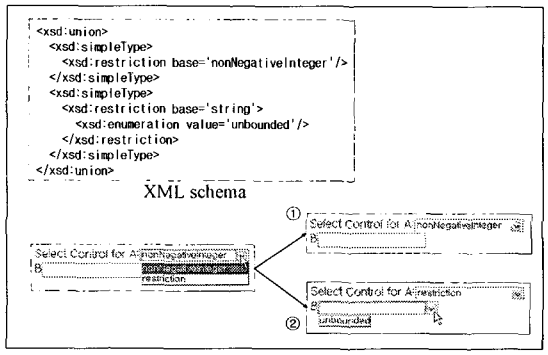


그림 16. union을 위한 컨트롤.

는 실험을 통해 성능을 분석하였다. 특히 생성된 XForms 인터페이스가 XML 스키마의 구조를 효과적으로 표현하고, 데이터 입력을 위한 적절한 컨트롤을 생성하는지를 검사한다. 또한 기존연구와의 비교를 통해 제안된 방법의 우수성을 분석, 기술한다.

4.1 시스템 구현

그림 17는 그림 1에서 정의된 스키마로부터 제안된 방법을 통해 생성된 XForms 인터페이스를 보여준다. 이 인터페이스는 생성된 XForms에 Cascading Style Sheets로 스타일을 적용하였으며, Mozilla Firefox의 XForms 모듈을 통해 렌더링하였다.

생성된 인터페이스는 XML 문서의 구조를 파악하기 쉽게 각 엘리먼트들을 블록 단위로 구분한다. 스키마에서 빈도 지시자에 의해 반복 구조로 정의된 엘리먼트인 Database, Mapping, 그리고 Parameter의 경우, 추가와 제거를 위한 버튼이 제공되고 있음을 보여준다.

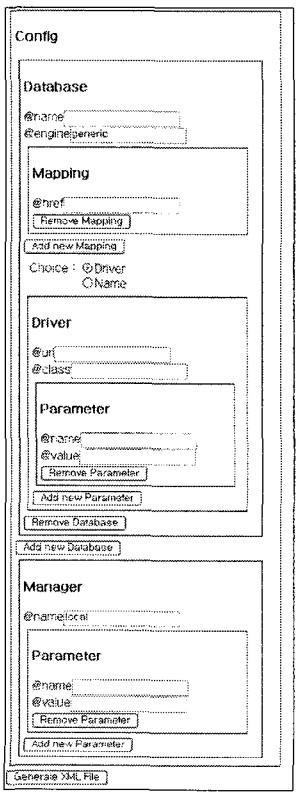


그림 17. 그림 1의 XML 스키마로부터 생성된 XForms 인터페이스의 예.

그림 18은 그림 17의 Manager 엘리먼트에 대한 편집 블록이다. XForms 인터페이스가 초기화되어 화면에 렌더링되었을 때 출력되는 컨트롤은 그림 18의 (a)에 해당한다. 그림 18의 (b)는 (a)에서 [Add new Parameter] 버튼을 2회 누른 후의 결과이다. 다음과 같이 [Add new Parameter] 버튼과 [Remove Parameter] 버튼의 조작을 통해 Parameter 엘리먼트의 개수를 늘이거나 줄일 수 있으며, 그에 따라 컨트롤의 개수 또한 변화한다.

스키마에서 choice로 선택 가능 하도록 정의된 Driver와 Name 엘리먼트를 위한 컨트롤은 그림 19와 같이 주어진다. Driver 또는 Name 엘리먼트를 위한 데이터를 입력할 컨트롤이 출력될 위치 상단에 두 엘리먼트를 선택할 수 있는 select1 컨트롤이 생성된다. 이 컨트롤에서의 사용자의 선택에 따라 엘리먼트가 활성화되며 그에 따른 컨트롤만이 화면에 출력된다.

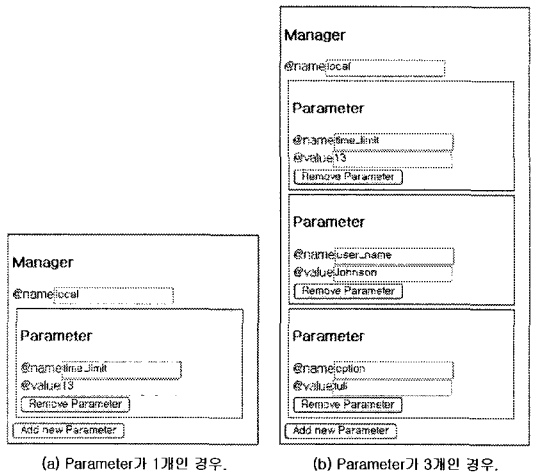


그림 18. 반복 구조의 인터페이스.

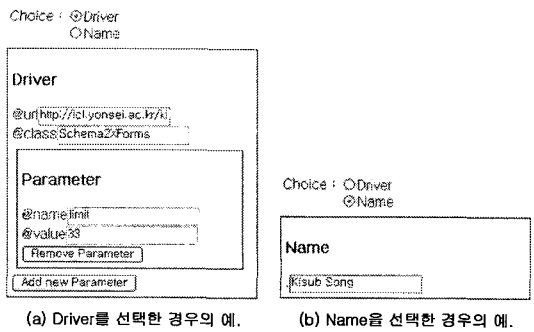


그림 19. 선택 구조의 인터페이스.

그림 17에서 주어진 XForms 인터페이스를 통해 생성된 XML 문서는 그림 20과 같다. 이 문서는 Driver와 Name의 선택 문항에서 Driver가 선택되었고, [Add new Parameter] 버튼을 통해Driver 하단의 Parameter 엘리먼트가 2개로 설정되었다. 사용자의 입력에 의해 주어진 스키마에 유효한 XML 문서가 생성된다.

4.2 성능 분석

제안된 방법의 성능을 분석하기 위해서 34개의 XML 스키마를 대상으로 실험하였다. 표 3은 실험에 사용된 34개의 스키마 파일의 목록이다.

34개의 스키마 파일에서 정의된 스키마 모델 노드의 개수 각각의 XForms 컨트롤 변환 결과는 표 4와 같다.

표 4에서 보듯 대부분의 노드에서 적절한 XForms 컨트롤을 생성하였다. 하지만 all 구조 정의에서는 XForms 인터페이스를 생성해내는 것에 실패했다. 또한 mixed 타입의 경우 반복 구조와 복합될 경우 실패하였다. 이 문제들은 현재까지 권고안으로 발표된 XForms 1.0에서 구현이 불가능하기 때문에 발생하는 것이며, 다음 버전인 XForms 1.1에서 해결될 수 있을 것으로 예상된다.

한편 데이터 타입에 따른 컨트롤 매핑의 실험 결과는 표 5와 같다. 수많은 내장 데이터 타입을 xforms:input 컨트롤로 매핑하였으며, 다양한 단순 데이터 타입에 대해 3.2절의 규칙을 적용하여 가장 적합한 인터페이스를 성공적으로 매핑해낼 수 있었다.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Config xmlns="http://icl.yonsei.ac.kr/kssong"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://icl.yonsei.ac.kr/kssong example.xsd"
    xmlns:tns0="http://icl.yonsei.ac.kr/kssong"
    xmlns:ev="http://www.w3.org/2001/xml-events"
    xmlns:xfam="http://icl.yonsei.ac.kr/XForms/AM"
    xmlns:xforms="http://www.w3.org/2002/xforms"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <Database engine="generic" name="engine">
    - <Driver class="XForms" url="http://icl.yonsei.ac.kr/Forms">
        <Parameter name="length" value="12" />
        <Parameter name="height" value="30" />
      </Driver>
      <Name>Schema2XForms</Name>
      <Mapping href="http://icl.yonsei.ac.kr/Forms/" />
    </Database>
  - <Manager name="local">
      <Parameter name="weight" value="3.1" />
    </Manager>
</Config>
```

그림 20. XForms 인터페이스로부터 생성된 XML 문서의 예.

표 3. 실험에 사용된 스키마

번호	스키마	크기(byte)
1	0423CommonRecord1pt0gVR.xsd	26715
2	20-16.xsd	894
3	BioCASE-MetaProfile-124.xsd	45247
4	CensusData.xsd	2421
5	CompetencyWeight.xsd	1888
6	DemographixQuery.xsd	12483
7	IndividualCertificate.xsd	1049
8	MemoChoiceMin0Max1.xsd	951
9	MemoChoiceMin1Max1.xsd	922
10	MemoElementGroup.xsd	1249
11	ProviderRegistry.xsd	1641
12	SendService.xsd	7062
13	TreeGen.xsd	2685
14	VOTable.xsd	14701
15	address.xsd	459
16	addressRedefine.xsd	700
17	binding.xsd	10655
18	conmap_2_1.xsd	13408
19	doc.xsd	1333
20	fraudlabswebservice.xsd	4477
21	ip2locationwebservice.xsd	1956
22	jdo-conf.xsd	3379
23	keywordTransfer_1.2.xsd	4033
24	mapping.xsd	13553
25	maven-v3_0_0.xsd	62876
26	mmcif_ccp4.xsd	48386
27	pdbx-v1.005.xsd	2313267
28	persistence.xsd	11801
29	persistence_1_0.xsd	9116
30	providerSetup_1_5.xsd	8653
31	settings-1.0.0.xsd	26073
32	uwm.xsd	2121
33	wrapperconfig_1.3.xsd	3138
34	ws4lsql.xsd	7224

* 실험 데이터: http://icl.yonsei.ac.kr/XForms/testdata

표 4. 스키마 구조 노드 변환 성공 여부

구조	개수	XForms 변환
엘리먼트	4208개	성공
애트리뷰트	1493개	성공
sequence	822개	성공
choice	32개	성공
all	1개	실패
maxOccurs & minOccurs	458개	성공
mixed	4개	일부 성공
any	1개	성공

표 5. 단순 데이터 구조 - 컨트롤 매핑 성공도

	XML 스키마 단순 데이터 타입		빈도	성공	성공률
가	내장 데이터 타입		…	…	100%
나	restriction에 의한 정의	한정된 길이의 문자열	5	5	100%
다		enumeration을 통한 선택항목화	123	123	100%
라		유한 구간의 정수형 또는 시간/날짜형	3	3	100%
마	list에 의한 정의	enumeration을 통한 선택항목화	2	2	100%
바	union에 의한 정의		3	3	100%

4.3 기존 연구와의 비교

스키마로부터 인터페이스를 생성하기 위해서는 지시자로 정의되어있는 반복, 선택 등의 스키마 구조에 대해 적절한 컨트롤을 제공해야 하며, 기본 데이터 타입에 대해 적절한 컨트롤이 매핑되어야 한다. 기존의 연구에서는 이 두 가지에 대한 세밀한 규칙 정의가 부족하다.

Lay와 Luttringhaus-Kappel[9], Radha 등[10], 그리고 Lee와 Kim[12]의 연구에서는 스키마의 반복 구조(Repetition), 선택 구조(Alternative)에 대한 처리 규칙을 다루고 있다. 하지만 XML 스키마의 모든 지시자에 대한 고려 없이 스키마를 반복 구조와 선택 구조로만으로 단순화 시켰기 때문에 복잡한 형태의 스키마에 대한 지원이 어려워진다는 단점이 있다. 제안된 방식에서는 XML 스키마의 복잡한 형태 역시 지원 가능하다.

기존 연구에서는 스키마의 데이터 타입에 따른 적합한 컨트롤을 매핑하지 않았다. Garvey와 French [15]의 연구에서 가이드라인을 제시하는 수준의 낮은 단계의 매핑을 보여주고 있을 뿐, 컨트롤에 대한 규칙을 제시하는 연구는 없었다. 제안된 방법에서는 XML 스키마의 단순 데이터 타입에 대한 세부적인 컨트롤을 제공하여 사용자 편의성을 높였다.

5. 결론 및 향후 연구 방향

XForms는 기존의 HTML 기반의 웹 폼을 개선할 수 있는 웹 기반의 인터페이스로서 XML기반의 데이터 송수신을 제공하기 때문에 웹 서비스의 인터페이스에 적합할 뿐만 아니라, 모델과 데이터를 분리함으로서 멀티 모달 인터페이스를 제공할 수 있도록 설계되었다.

본 논문에서는 XML 스키마 문서로부터 XForms 기반의 인터페이스를 자동 생성하는 방법을 제안하였다. 이 방법을 통해 스키마 구조 분석을 통해 빈도 지시자로 정의된 반복 구조의 XML 구조를 위한 인터페이스와 choice 지시자를 통해 정의된 선택 구조

표 6. 제안된 방법과 기존 연구와의 비교

비교 항목	Jelinek와 Slavik [8]	Lay와 Luttringhaus -Kappel [9]	Radha 등 [10]	Kuo 등 [11]	Lee와 Kim [12]	Garvey와 French [15]	제안된 방법
기반 스키마	Annotated Source Code	XML Schema	XML Schema	XML Schema	DTD	XML Schema	XML Schema
GUI	GUI	Java Swing	GUI	HTML	XForms +XSL	XForms	XForms
스키마 구조 처리 규칙	×	○	○	×	○	×	○
컨트롤 매핑 규칙	×	×	×	×	×	△ (낮은 수준)	○
자동화 (사용자 개입 불필요)	사용자 개입	자동	자동	사용자 개입	자동	자동	자동

의 XML 구조를 완벽하게 처리한다. 또한 XML 스키마의 복잡한 스키마 구조를 대부분 처리할 수 있는 규칙을 정의하였다. 제안된 방법에서는 단순 데이터 타입에 대한 XForms 컨트롤 매핑 규칙을 세밀하게 정의하여 사용자에게 가장 편리한 입력 방식을 제공할 수 있게 하였다. 따라서 본 논문에서 제안한 방식으로 XML 문서에 대한 편집에 대한 편의성을 높일 수 있었다.

현재 점점 더 많은 정보가 XML 형태로 생산되고 있으며, 웹에서뿐만 아니라 디지털 기기들의 프로파일이나 데이터 관리에서 XML이 표준으로 사용되고 있다. 따라서 XML 데이터를 편집하기 위한 다양한 방법은 더욱더 필요할 것이다. 하지만 XML 데이터를 편집할 수 있는 사용자 인터페이스를 제공하기 위한 구체적인 방법에 대한 연구는 부족하다. 현재까지 제안된 연구는 다양한 환경을 지원하는 XForms에 대한 고려가 없으며, 있다하더라도 복잡한 스키마에 대한 처리가 부족하고 다양한 단순 데이터 타입에 대한 적절한 컨트롤을 제시하지 못했다. 본 논문에서는 이런 점들을 보완하여 사용자에게 더욱 편리한 인터페이스를 제공할 수 있게 하였다.

향후 연구에서는 현재 개발 완료를 목전에 두고 있는 XForms 1.1에 대응하는 인터페이스 생성 방법을 제시할 예정이며, 그에 관련한 기술에 대하여 연구할 예정이다. 이를 위해 온톨로지 관련 기술을 응용하는 방법 등을 연구할 예정이다. 또한 이 기술을 응용하여 웹 서비스를 호출하기 위한 사용자 인터페이스를 자동 생성하는 방법에 대해서 연구할 예정이다.

참 고 문 헌

[1] ArborText Inc., "W3C XML Specification DTD (XMLspec)," http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm.

[2] World Wide Web Consortium, "XML Schema," http://www.w3.org/XML/Schema.

[3] World Wide Web Consortium, "HyperText Markup Language (HTML)," http://www.w3.org/MarkUp/.

[4] WAP Forum, LTD., "WAP WML Specification Version 1.3," http://www.wapforum.org, 2000.

[5] World Wide Web Consortium, "XSL Transformations (XSLT) Version 1.0," http://www.w3.org/TR/xslt.

[6] World Wide Web Consortium, "The Swing Tutorial," http://java.sun.com/docs/books/tutorial/uiswing/.

[7] World Wide Web Consortium, "XForms - The Next Generation of Web Forms," http://www.w3.org/MarkUp/Forms/.

[8] J. Jelinek and P. Slavik, "GUI generation from annotated source code," *Proc. 3rd Annual Conf. on Task Models and Diagrams*, pp. 129-136, 2004.

[9] P. Lay and S. Lüttringhaus-Kappel, "Transforming XML Schemas into Java Swing GUIs," *1st Workshop on Web Applications and Middleware (WAM 2004)*, Vol. 50, pp. 271-276, 2004.

[10] V. Radha, S. Ramakrishna, and N. Pradeep kumar, "Generic XML Schema Definition (XSD) to GUI Translator," *Proc. Distributed Computing and Internet Technology: Second International Conf. ICDCIT 2005, LNCS 3816*, pp. 290-296, 2005.

[11] Y. S. Kuo, N. C. Shih, Lendle Tseng, and Hsun-Cheng Hu, "Generating Form-Based User Interfaces for XML Vocabularies," *Proc. 2005 ACM Symposium on Document Engineering*, pp. 58-60, 2005.

[12] E. Lee and T. Kim, "Automatic Generation of XForms Code Using DTD," *Proc. Fourth Annual ACIS Int'l Conf. Computer and Information Science*, pp. 210-214, 2005.

[13] M. Kassoff, D. Kato, and W. Mohsin, "Creating GUIs for Web Services," *IEEE Internet Computing*, Vol. 7, No. 5, pp. 66-73, 2003.

[14] R. Steele, K. Khankan, and T. Dillon, "Mobile Web Services Discovery and Invocation Through Auto-Generation of Abstract Multimodal Interface," *Proc. Int'l Conf. Information*

Technology: Coding and Computing, Vol. 02, pp. 35-41, 2005.

[15] P. Garvey and B. French, "Generating User Interfaces from Composite Schemas," http://www.idealliance.org/xmlusa/03/call/xmlpapers/03-03-04.994/.03-03-04.html#S1.

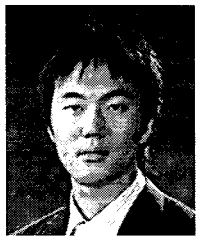
[16] J. Kawash, "Declarative User Interfaces for Handheld Devices," *Proc. Winter Int'l Symposium Information and Communication Technologies*, pp. 1-6, 2004.

[17] C. Nam, G. Jang, and J. J. Bae, "An XML-based active document for intelligent web applications," *Expert Systems with Applications*, Vol. 25, No. 2, pp. 165-176, 2003.

[18] M. Honkala, P. Cesar, and P. Vuorimaa, "A device independent XML user agent for multimedia terminals," *Proc. IEEE 6th Int'l Symposium Multimedia Software Engineering*, pp. 116-123, 2004.

[19] A. Tiwari, R. A. Hosn, and S. H. Maes, "Conversational multi-modal browser: an integrated multi-modal browser and dialog manager," *Proc. 2003 Symposium Applications and the Internet*, pp. 348-351, 2003.

[20] World Wide Web Consortium, "Scalable Vector Graphics (SVG)," http://www.w3.org/Graphics/SVG/.



송 기 섭

2005년 8월 연세대학교 컴퓨터산업공학부 졸업(학사)
2005년 9월~현재 연세대학교 컴퓨터과학과 석사과정
관심분야 : User Interfaces for Web Services, Multimodal User Interfaces

이 경 호

1995년 2월 연세대학교 전산과학과 졸업(학사)
1997년 2월 연세대학교 컴퓨터과학과 졸업(석사)
2001년 2월 연세대학교 컴퓨터과학과 졸업(박사)
2001년 3월 National Institute of Standard and Technology(NIST) 객원연구원
2002년~현재 연세대학교 컴퓨터과학과 부교수
관심분야 : Internet Computing, Service-Oriented Computing, Multimedia Document Engineering