DOI: 10.3745/KIPSTD.2008.15-D.6.879

서비스 조합을 위한 XForms 기반의 모바일 사용자 인터페이스 개발

이 은 정†

요 약

최근 웹 서비스가 다양한 응용 분야에 도입되면서 모바일 환경에서도 웹서비스 조합 어플리케이션의 개발이 활발하다. 모바일 환경에서 여러 개 서비스를 지원하는 사용자 인터페이스는 각 서비스의 호출과 응답 확인을 위한 여러 개의 뷰가 필요하고 서비스 간의 조합 순서와 흐름을 분석할 필요가 있다. 본 논문에서는 서비스 집합의 명세로부터 프리젠테이션 요소인 뷰를 추출하고 서비스 입출력 데이터의 타입에 기반하여 뷰 및 서비스 흐름 분석 방법을 제안한다. 뷰 및 서비스 흐름 분석으로부터 서비스 호출을 위한 사용자 인터페이스 요소가 자동 생성될 수 있다. 제안된 방법은 서비스 조합을 위한 모바일 사용자 인터페이스를 생성할 수 있다. 마지막으로 본 논문에서는 자체 개발된 XForms 브라우저를 이용하여 제안된 방법에 의해 생성된 XForms 페이지가 REST 기반 서비스 접속을 위한 클라이언트로 동작할 수 있음을 보였다.

키워드 : XForms, 서비스 조합, 모바일 프리젠테이션 프레임워크

Developing XForms Based Mobile User Interface for Web Service Composition

Eunjung Lee†

ABSTRACT

As web services have become an important architecture solution, web service composition applications are developed actively. A mobile application supporting multiple services requires a complex user interface so that the interface needs to consist of more than one view and to provide a way to navigate between views. In this paper, we presented a formal way to analyze a set of views for a given service specification, and a relation model between views and methods. We then provided an algorithm to generate codes for service method calls and navigation between views. Therefore, with an optional user configuration input, we could automatically generated XForms codes from the web service specifications. Finally, we developed a proof of concept implementation of XForms browser to show that the generated codes works well as an interface for web service compositions.

Keywords : Xforms, Service Composition, Mobile Presentation Framework

1. 서 론

REST(Representational State Transfer) 기반의 웹서비스[9]는 서비스 중심 아키텍춰(Service Oriented Architecture)를 구현하기 위한 가벼운 패러다임으로 광범위하게 사용되고 있다. REST 서비스는 가벼운 메시지 통신 방법으로서 개발이 간단하고 유연하여 모바일 환경에 적합하여 상황이나 장소에 따라 필요한 웹 기반의 서비스를 제공하는 어플리케이션 개발이 가능하다.

그러나 모바일 환경에서는 개발 환경이나 방법론이 아직 미비하여 클라이언트 어플리케이션 개발이 어렵고 특히 사용자 인터페이스(User Interface, 이하 UI) 코드 개발은 많은 시간과 자원을 요구한다고 알려져 있다. 이를 극복하기 위하여 선언적 UI 명세 언어를 이용하는 모바일 클라이언트 개발 방법이 선호된다. 또한 서비스와 상호작용하는 모바일 어플리케이션 개발을 위해 XML 기반의 UI 기술 언어를 이용하는 방법이 널리 사용되고 있다. XForms는 차세대 웹 폼 언어로서[5, 7, 21] XML 데이터 모델을 이용하고 서비스 호출과 상호작용을 위한 통신 요소를 포함하고 있어 REST 기반의 서비스와 잘 결합될 수 있다[5].

웹서비스 클라이언트의 사용자 인터페이스 개발 방법은 많이 연구되었으나[2, 11, 12, 16], 아직까지는 모바일 환경에서 서비스 조합을 고려한 사용자 인터페이스에 대한 연구는

※ 본 연구는 경기도의 경기도지역협력연구센터사업[컨텐츠융합소프트웨어연구센터]과 경기대학교특성화사업[생물자원보존, 복원, 개발을 위한 BEIT융합기술특성화사업]의 일환으로 수행하였음.
† 종신회원 : 경기대학교 컴퓨터과학과 교수
논문접수 : 2008년 7월 7일
수 정 일 : 1차 2008년 11월 7일
심사완료 : 2008년 11월 10일

없었다. 특히 여러 개의 서비스 호출을 위한 통합 인터페이스는 모바일 환경의 단말 특성에 적합하지 않으므로 여러 개의 뷰(또는 카드)로 나누어 어플리케이션을 구성하는 것이 일반적이다. 예를 들면 목록, 개별 서비스 호출을 위한 입력, 그리고 결과를 보여주는 뷰로 나누어 볼 수 있다. 여러 개의 뷰로 구성된 어플리케이션에서는 뷰 간의 이동 기능이 제공되어야 한다.

본 논문에서는 서비스 명세로부터 서비스 조합을 지원하는 클라이언트 코드를 자동생성하는 방법을 제시하고자 한다. 서비스 명세를 위한 입력 언어로는 WADL(Web application descrition language) 표준[19]을 사용하고 코드 생성을 위한 목적 언어는 XForms를 이용한다. 서비스 조합을 지원하는 클라이언트 코드의 생성을 위하여 본 논문에서는 다음의 두 가지 문제를 해결하였다. 첫째, 서비스 명세로부터 지역 데이터 모델을 생성하는 방법을 제시하였다. 입력 매개변수 타입과 결과 데이터 타입을 이용하여 지역 데이터 모델의 생성 알고리즘을 제시하였고 이것을 통하여 코드생성에 필요한 데이터 부분의 경로를 구할 수 있다. 두 번째로 본 논문에서는 클라이언트 프로그램이 가질 뷰와 뷰 간의 이동 관계를 나타낼 정형화된 모델을 제시하였다. 타입 기반의 부분 매칭에 의한 웹서비스 조합 방법을 적용하여 뷰 간의 관계를 분석하고 이를 바탕으로 뷰와 메소드, 뷰와 뷰 간의 호출과 이동을 뷰이동그래프라는 유한상태 오토매타로 모델링하였다.

제안된 지역 데이터 모델과 뷰이동그래프를 이용하여 뷰와 뷰간의 이동 및 메소드 호출을 포함한 서비스 조합을 위한 XForms 페이지 코드를 생성할 수 있다. 본 논문에서는 한줄블로그 사례 시나리오를 통해 제안된 코드 생성 방법이 실제 모바일 환경에서 적용 가능함을 보였다.

본 논문의 구조는 다음과 같다. 2장에서는 모델을 활용한 코드 자동생성 기술과 XForms 관련 연구를 살펴보고 REST 기반 서비스를 중심으로 하는 서비스 기술을 살펴본다. 3장에서는 서비스 명세와 이를 통한 뷰 이동 그래프 모델을 제시한다. 4장에서는 지역 데이터 모델 및 뷰 코드 생성 방법을 살펴보고 5장에서 결론을 맺는다.

2. 관련 연구

웹서비스 조합[13, 21]을 위한 UI 개발은 주로 모델 기반의 자동생성 기술을 중심으로 연구되었다[11, 12].

모델링 방법을 통해 체계적인 웹어플리케이션 개발 방법론을 제시하고자 하는 노력이 많이 있었다. UML 모델링으로부터 자바 코드를 생성하는 연구로 UWE(UML based Web Engineering)가 있었고[12] 웹서비스의 기능적인 통합을 고려한 모델링 언어로 WebML(XML-based Web Modeling Language)을 이용하는 방법이 제안되었다[4]. 그러나 이들 연구에서는 웹어플리케이션의 아키텍춰 표준화와 모델링을 통한 체계적인 개발이 목적이었으므로 UI 코드 생성은 중심적으로 고려되지 않았다. 웹어플리케이션의 UI 개발 기술은 주로 선언적 언어로 기술된 플랫폼 독립적인 코드를 실제 여러 플랫폼에 실현하는 방식이 많이 연구되었다[14,15]. 여기에는 XUL(XML User Interface Language)이나 XForms를 이용하여 개발된 UI 코드를 처리할 수 있는 확장 브라우저 시스템에 대한 연구가 많이 있었다. 한편 개념적인 모델링과 적응 규칙을 이용한 상황 적응적인 웹 UI 생성 기술도 제시되었다[6]. 또한 프리젠테이션 요소의 통합 기술로 다양한 종류의 프리젠테이션 컴포넌트를 기술하고 조합 모델을 이벤트 기반의 리스너 모델로 기술하여 통합 UI를 생성하는 방법이 제안되었다[17].

웹서비스 코드 생성 기술로는 WSDL(Web service description language)로부터 자바나 HTML 등의 클라이언트 코드를 생성하는 기술이 제시되었으며[18] 특히 XForms 언어를 이용하여 웹서비스를 접속할 수 있는 UI 코드를 자동 생성하는 방법이 제시되었다[2, 16]. 또한 모바일 서비스 플랫폼에서 서비스 브로커가 서비스 별로 생성된 XForms 코드 페이지를 가지고 클라이언트에서 이를 다운로드하여 서비스를 접속하는 아키텍처가 BAMOS 시스템에서 제안되었다[8]. 그러나 이들 방법은 주로 하나의 메소드 호출(submit) 후 다음 페이지로 이동하는 방식으로 자바스크립트의 Ajax 기술[21]을 사용하는 웹서비스 클라이언트처럼 응답 결과를 처리하는 콜백 함수 기능은 자동으로 코드 생성할 수 있는 대상이 아니었다.

웹서비스 조합 UI라면 응답 수신 처리 뿐 아니라 뷰 이동이나 다음 서비스 호출 등 서비스 호출 및 뷰 시퀀스를 나타내는 코드를 가져야 한다. 특히 모바일 환경에서는 여러 개의 뷰를 가지는 클라이언트 어플리케이션에서 뷰 간의 이동이나 메소드 호출 순서를 표현할 수 있어야 한다.

본 논문에서는 데이터 중심 웹서비스 조합 어플리케이션에서 CRUD(Create, Read, Update, Delete) 기반의 사용 시나리오를 바탕으로 코드를 자동생성하는 방법을 제시하고자 한다.

3. 모 델

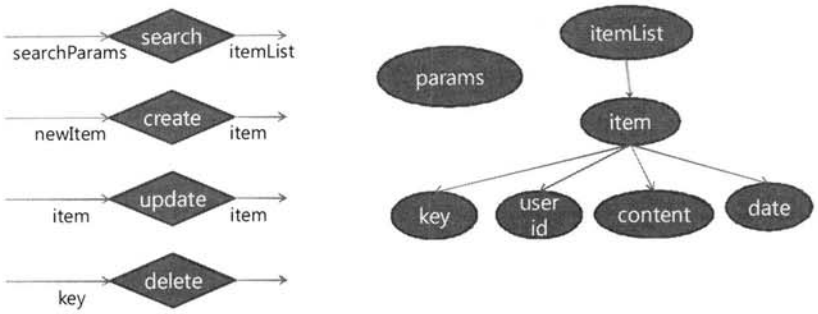
3.1 서비스 모델과 데이터 타입

본 논문에서는 웹서비스 명세를 이용하여 클라이언트 프로그램의 코드를 생성하는 방법을 살펴보고자 한다. 먼저 서비스 명세의 모델을 소개한다.

웹서비스의 메소드는 호출할 주소와 이름을 포함하는 URL(Universal Resource Location), 입력 매개변수 타입과 결과 타입으로 정의될 수 있다. 타입은 XML 스키마[21]에 정의된 요소의 이름으로 표시된다.

[정의 1] 메소드는 $m = (id, url, input, output)$으로 정의되며 차례로 메소드 아이디, 주소, 입력 및 결과 타입을 나타낸다.

메소드 m의 각 요소를 $m.id$, $m.url$, $m.input$, $m.output$으로 표기한다. 서비스 명세 M은 메소드의 집합으로 정의된다. $m.input$과 $m.output$은 스키마에 정의된 하나의 요소 타입으



(a) 서비스 메소드 명세 다이어그램 (b) 요소 타입 간의 포함관계

(그림 2) 한줄블로그 서비스 명세 $M_{onelineblog}$의 서비스 메소드 명세 다이어그램과 요소 타입 간의 포함관계

로 표시된다고 가정한다. 클라이언트 어플리케이션은 사용자들이 주어진 M의 메소드들을 사용할 수 있도록 인터페이스를 제공해야 한다. 예를 들어 본 논문의 기본 예제로 사용되는 한줄블로그 서비스는 다음과 같은 메소드 집합 $M_{onelineblog}$로 정의된다.

$M_{onelineblog}$ = {(search, http://locahost:9090/onelineblog/search.xml, params, itemList),
(create, http://localhost:9090/onelineblog/create.xml, item, item),
(update, http://localhost:9090/onelineblog/update.xml, newItem, item),
(delete, http://localhost:9090/onelineblog/delete.xml, key, e)}

이 서비스 명세는 (그림 1(a))와 같은 다이어그램으로 표시된다.

서비스 명세는 스키마에서 정의된 요소 이름으로 표시되는 몇 가지 타입을 포함한다. 서비스 명세에서 사용되는 모든 타입의 집합을 다음과 같이 정의한다.

$$L = \{ \lambda \mid \lambda = m.input \text{ or } \lambda = m.output,\ m \in M \}$$

L의 타입들은 스키마의 구조 정의에 의해 서로 포함관계를 가질 수 있다. 이 관계는 타입 기반의 서비스 조합을 위하여 메소드 간의 관계를 분석할 수 있게 해 준다[10]. 또한 이들 타입으로부터 클라이언트 시스템이 사용할 지역 데이터 모델을 얻을 수 있다. 본 논문에서는 각 요소의 레이블이 스키마 정의에서 한 번씩만 사용된다고 가정하고 간단한 타입 간의 포함 관계에 기반한 부분 매칭 방법을 사용한다. 본 논문에서의 포함 관계를 다른 웹서비스 조합 방법으로 대치하여 사용하는 것도 가능하다. 본 논문에서 사용되는 포함 관계는 다음과 같이 정의된다.

[정의 2] $\lambda_1, \lambda_2 \in L$에 대해 스키마 정의에서 λ_1이 λ_2의 부모 요소라면 λ_1이 λ_2를 직접 포함한다고 하고 $\lambda_1 \rightarrow_d \lambda_2$로 표시한다. λ_1과 λ_n 사이에 $\lambda_1 \rightarrow_d \lambda_2 \rightarrow_d \ldots \rightarrow_d \lambda_n$, $n \geq 1$을 만족하는 $\lambda_2, \ldots, \lambda_{n-1}$이 존재한다면 λ_1이 λ_n을 포함한다고 하고 $\lambda_1 \rightarrow \lambda_n$로 표시한다.

한편 $\lambda_1 \rightarrow_d \lambda_2$이고 λ_2가 반복될 수 있다면 λ_1이 λ_2를 직접 반복 포함한다고 하고 $\lambda_1 \twoheadrightarrow_d \lambda_2$로 표시한다. 또한 $\lambda_i \twoheadrightarrow_d \lambda_{i+1}$를 만족하는 $1 \leq i < n$가 존재하면 반복 포함한다고 하고 $\lambda_1 \twoheadrightarrow \lambda_n$로 표시한다.

(그림 1)의 예제에서 $L_{onelineblog}$ = {itemList, item, key, params}이고 todoList $\twoheadrightarrow$ todoItem, todoItem$\rightarrow$name와 todoItem$\rightarrow$key 관계가 만족한다. $L_{onelineblog}$에 속하는 타입 간의 포함관계가 (그림 1(b))의 트리에 표현되어 있다.

3.2 서비스 명세의 뷰 집합 계산

클라이언트 어플리케이션이 여러 서비스 메소드와 상호작용하기 위해서는 모바일 단말의 크기 제약 때문에 여러 개의 뷰로 나누는 것이 자연스럽다. 이 절에서는 서비스 명세 M에 대해 필요한 뷰를 추출하는 방법과 뷰를 명세하는 방법을 소개한다.

초기의 기본 뷰 집합으로 모든 메소드에 대해 입력 및 결과 뷰를 하나씩 가정한다. 입력 뷰는 사용자가 서버에 전송할 매개변수 데이터를 준비하고 확인하는 뷰이고 결과 뷰는 서버로부터의 회신을 사용자에게 보여주기 위한 뷰이다. 뷰는 다음과 같이 정의된다.

$$v = (m,\ io,\ target,\ properties),$$

여기서 $m \in M$, $io \in \{in, out\}$이고 $target \in L$은 입력 또는 결과 데이터의 요소 타입이다. $properties$는 사용자가 제공하는 설정 값으로 뷰 목록과 이동의 최적화를 위하여 아래에서 따로 정의한다.

주어진 메소드 $m \in M$에 대해 입력 및 결과 뷰는 $view_{in}(m)$, $view_{out}(m)$로 표시하고 뷰의 집합 V는 다음과 같이 정의된다.

$$V = V_{in} \cup V_{out},\ V_{in} = \{view_{in}(m) \mid m \in M\},$$
$$V_{out} = \{view_{out}(m) \mid m \in M\}.$$

또한 뷰에 대응하는 입력 또는 결과 타입은 $v.target$으로 표시된다. 여기서는 이러한 뷰의 집합과 개별 뷰의 속성을 표시하기 위하여 뷰 명세 언어라는 XML 언어를 이용한다. 다음 알고리즘은 뷰 명세 파일을 생성하는 방법을 보여준다.

뷰의 target은 L에 포함된 요소 타입이므로 정의 2의 타입 포함관계에 의해 뷰 간의 관계를 계산할 수 있게 해 준다. 본 논문에서는 뷰 간의 관계를 데이터 포함 관계로 분

〈표 1〉 뷰 명세 파일 VS 생성 알고리즘

(1) 주어진 메소드 m = (id, url, input, output) ∈ M에 대해,
(1-1) v = $view_{in}$(m)이면, <view id=m.id, inout="in", target = m.input/>를 VS에 추가한다.
(1-2) v = $view_{out}$(m)이면, <view id=m.id, inout="out" target = m.output/>를 VS에 추가한다.

〈표 2〉 $M_{onelineblog}$에 대응하는 뷰 명세 $VS_{onelineblog}$

```
<view id="search" io="in", target="search_param" properties="initialized" start="true"/>
<view id="search" io="out", target="itemList" properties=""/>
<view id="create" io="in", target="newItem" properties="initialized"/>
<view id="create" io="out", target="item" properties=""/>
<view id="delete" io="in", target="key", properties="skipped"/>
<view id="delete" io"out", target="none", properties="skipped"/>
```

석하고자 한다. 즉 결과 뷰 v_1과 입력 뷰 v_2에 대해 v_1.target → v_2.target의 관계를 만족한다면, v_1에서 v_2로의 뷰 이동이 가능하다. 이것은 웹서비스 조합에서 타입 기반의 부분 매칭 방법과 유사하다. 뷰가 이동할 때 v_2.target에 해당하는 데이터가 v_1에서 v_2로 전달된다. v_1과 v_2의 target 타입들이 반복 포함관계를 가지면 전달 데이터는 반복부 리스트에서 현재 선택된 요소가 될 것이다.

한편 본 논문에서는 사용자가 설정 입력을 이용하여 뷰의 성질을 지정하여 이동 관계를 최적화할 수 있는데, 초기화(initialized) 및 생략가능(skipped)이라는 속성을 정의한다. 초기화 뷰는 뷰의 데이터 초기값이 필요 없는 경우로 사용자가 처음부터 새로 데이터를 입력하므로 이전 뷰로부터 데이터의 전달이 필요없는 경우이다.

한편 V에 속하는 뷰 중에는 사용자 상호작용이 전혀 필요 없거나 보여줄 데이터가 없는 경우가 있다. 입력 뷰가 생략 가능한 경우는 이전 뷰에서 입력 매개변수 데이터가 결정되어 바로 메소드를 호출한다. 또한 결과 데이터가 없는 경우는 결과 뷰를 생략할 수 있다. 예를 들어 삭제 메소드는 입력 파라매터로 id 값을 가지고 결과 데이터는 없는데, 이 경우 입력 및 결과 뷰는 생략 가능하다.

이러한 초기화 및 생략가능 속성은 사용자가 VS 파일을 편집하여 설정할 수 있다. <표 2>는 한줄블로그 예제의 뷰 명세 파일인 $VS_{onelineblog}$를 보여준다. 이 예제에서 검색과 생성 뷰는 초기화 속성을 가지고 삭제 메소드의 입력 및 결과 뷰는 생략되었다. 한편 초기뷰를 사용자가 설정하기 위하여 start 속성을 지정할 수 있다.

3.3 뷰 간의 관계

다음 단계로 뷰 명세에서 정의된 뷰 간의 이동을 정의하기 위하여 유한 상태 기계를 정의한다.

[정의 3] V는 뷰의 집합이고 M은 메소드의 집합, 그리고 L은 타입의 집합이라 하자. 또한 V에 속하는 뷰 v_0를 어플리케이션의 초기뷰라고 하자. 그러면 뷰이동그래프 A는 다음과 같은 상태 전이 기계로 정의된다.

$$A = (V \cup M, \delta, v_0, \varnothing),$$

여기서 $\delta = \delta_{call} \cup \delta_{move}$이고 이들은 다음과 같이 정의된다.

δ_{call} = {(v, m) | v = $view_{in}$(m)이거나, v.target → v_1.target이면서 v_1 = $view_{in}$(m), v_1.skipped = true},
δ_{move} = {(v_1, v_2) | $v_2 \in V_{in}$에 대해 v_2.initial = true이거나 v_1.target → v_2.target}.

뷰이동그래프에서 메소드 호출과 뷰 이동이 데이터의 포함관계를 이용하여 정의되었다. 전이 함수 $\delta = \delta_{call} \cup \delta_{move}$가 데이터의 포함관계를 만족함을 다음 보조정리에서 보여준다.

[보조정리 4] 정의 3의 뷰이동그래프 A에서 $(v, m) \in d_{call}$이면 v.target → m.input을 만족한다.

(증명) v = $view_{in}$(m)이라면 v.target = m.input이므로 v.target → m.input을 만족한다. 한편 v_1 = $view_{in}$(m)이라면 v_1.target = m.input이고 그러므로 v.target → m.input을 만족한다.

본 논문에서는 뷰 간의 이동을 데이터 전달 가능성으로 분석하였다. 즉 뷰 v_1에서 뷰 v_2로의 이동은 v_2가 필요한 데이터를 v_1이 전달할 수 있는 경우 가능하다. 이를 위하여 뷰의 필요 데이터를 v.pre로 다음과 같이 정의한다. $v \in Vin$에 대해

v.pre = ∅, if v.initial=true or
v.target, otherwise.

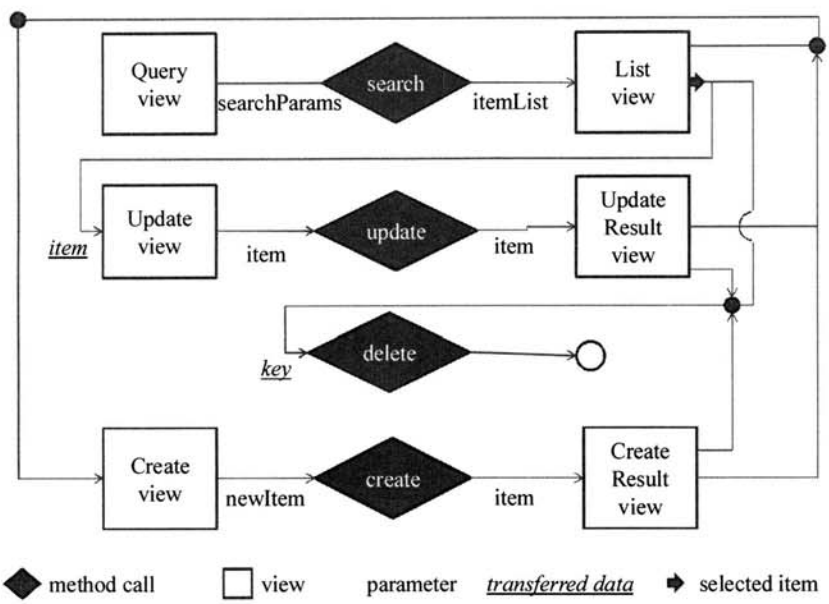
다음 보조정리는 δ_{move}에서 정의된 뷰 이동 전이가 뷰의 필요 데이터의 포함관계를 만족함을 보여준다.

[보조정리 5] 정의 3에서 정의된 뷰이동그래프 A에서 $v_1 \in V$, $v_2 \in V_{in}$에 대해 $(v_1, v_2) \in d_{move}$이면 v_1.target →

〈표 3〉 전이 함수 계산 알고리즘

입력 : 뷰 집합 V와 서비스 명세의 메소드 집합 M, 초기 뷰 v_0
결과 : $d = \delta_{call} \cup \delta_{move}$

(1) $V_{skipped} = \{ v \in V \mid v.skipped = true\}$, $V_{init} = \{v \in V \mid v.initialized = true\}$
(2) 메소드 $m \in M$에 대해 $v = view_{in}(m)$라면 $\delta_{call} \leftarrow \delta_{call} \cup \{(v, m)\}$.
(3) 메소드 $m \in M$에 대해 $v \in view_{out}(m)$라면
(3-1) 다른 메소드 $m' \in M$의 입력 뷰 $v_2 \in view_{in}(m')$에 대해
(3-1-1) $v_1.target \rightarrow v_2.target$의 관계를 만족하면 $\delta_{move} \leftarrow \delta_{move} \cup \{(v_1, v_2)\}$.
(4) $v \in V_{skipped}$인 모든 뷰 v에 대해
(4-1) $(v_1, v) \in \delta_{move}$이고 $(v, m) \in \delta_{call}$인 모든 뷰 v_1에 대해 $\delta_{call} \leftarrow \delta_{call} \cup \{(v_1, m)\}$.
(4-2) $V = V - \{v\}$.
(5) $v \in V_{init} \cap (V_{in} - V_{skipped})$인 모든 뷰 v에 대해
(5-1) $v' \in V_{out}$이면 $\delta_{move} \leftarrow \delta_{move} \cup \{(v, v')\}$.



(그림 4) 한줄블로그 예제의 뷰이동그래프 $A_{onelineblog}$

$v_2.pre$를 만족한다.

(증명) $v.initial$이 거짓인 경우는 $v_1.target \rightarrow v_2.target$을 만족하므로 $v_1.target \rightarrow v_2.pre$가 만족되고 $v.initial$이 참인 경우는 $v_2.pre = \varnothing$이므로 $v_1.target \rightarrow v_2.pre$를 만족한다.

이러한 성질을 이용하여 뷰이동그래프의 전이 에지를 계산하는 알고리즘이 <표 3>과 같다.

한줄블로그 뷰 명세 $VS_{onlineblog}$에서 계산된 뷰 이동 그래프는 (그림 2)와 같다. (그림 2)에서 뷰는 네모로, 메소드 호출은 마름모꼴로 표시된다. 이 그래프 모델은 뷰 간의 이동을 위한 사용자 액션을 제공한다. 뷰가 메소드로의 에지를 가지면 그 뷰는 해당 메소드에 대한 submit 액션을 포함할 것이다. 또한 뷰 v_1에서 뷰 v_2로의 에지가 있으면 뷰 이동을 위한 사용자 액션(버튼 또는 메뉴 아이템)이 포함됨을 나타낸다.

4. 서비스 조합을 위한 XForms 코드 생성

주어진 서비스 명세로부터 XForms 코드를 생성하기 위하여 본 논문에서는 서비스 명세를 위하여 실제 인터넷에서 사용되는 REST 서비스 명세 언어인 WADL을 사용한다. 또한 목적 언어로서 XForms는 사용자 인터페이스를 위한 웹 폼 컨트롤을 제공하고 서버와의 XML 데이터 통신을 위한 기능을 제공하다. XForms 페이지는 모델과 인터페이스 부분으로 나누어지고 후자는 구조 및 컨트롤 요소들로 구성된다.

서비스 명세는 사용되는 데이터 타입의 스키마 정의를 포함한다. 데이터 타입으로부터 사용자 인터페이스 컨트롤 코드를 생성하는 것이 가능하다[16, 18]. 보통 터미널 데이터는 읽기 및 쓰기 연산의 후보가 되고 반복부 데이터는 삽입 및 삭제 연산의 대상이 된다. 그러나 기존의 웹서비스를 위한 XForms 코드 생성에 관한 연구에서는 메소드 호출과 뷰 이동은 다루어지지 않았다. 여기서는 3장에서 소개된 뷰이동그

래프로부터 이러한 코드를 생성하는 방법을 제안하고자 한다.

4.1 서비스 명세 언어 : WADL

REST 기반의 웹서비스는 기존의 방법과 달리 WSDL(Web service specification language)이나 SOAP 프로토콜을 사용하지 않고 HTTP 통신을 이용하여 직접 XML 데이터를 전달한다. 그 결과 작고 가벼운 메시지와 처리 시스템이 가능했다. REST 기반 서비스는 주로 XML 스키마와 HTML 문서 형태로 기술되었는데 최근에는 서비스 발견이나 조합 등의 새로 등장한 기능을 제공하기 위하여 서비스 명세가 필요하게 되었다. 이러한 요구를 위하여 제안된 표준이 WADL이다[19].

WADL로 기술된 하나의 파일은 어플리케이션을 정의하는데 이것은 여러 개의 스키마 정의와 리소스를 포함하고 각 리소스는 여러 개의 메소드를 포함한다. 각 메소드는 서비스 호출 주소와 이름, 입력 매개변수 타입과 결과 데이터 타입을 기술한다. 각 입력 매개변수 타입과 결과 타입은 하나의 요소 타입으로 정의된다.

한줄블로그 예제 $M_{onelineblog}$에서 검색 메소드의 서비스 명세는 <표 5>와 같은 WADL 코드로 표현된다.

4.2 지역 데이터 모델과 submission 요소 생성

클라이언트 시스템은 서비스 인터페이스를 위하여 지역 데이터 모델을 가지고 있다. 지역 데이터 모델은 클라이언트가 필요로 하는 데이터의 복사본으로 볼 수 있다. 이것은 입력 매개변수를 위한 메모리 역할을 하고 서비스 호출의 결과 데이터가 저장되는 공간이기도 하다. 결과 데이터는 부분 트리의 형태로 지역 데이터 모델에 저장된다. 서비스 호출의 응답이 돌아오면 해당 부분 트리가 갱신된다. <표 6>의 알

〈표 4〉 WADL 파일 구조

```
application
grammars
resources @base
        resource @uri
                (*) method href
method @name @id
        request
                query_variable @name @type @required
        response
                representation @mediaType @element
                fault @id @status @mediaType @element
```

〈표 5〉 한줄블로그 예제에 대한 WADL 코드 일부

```
<method name = "GET" id="search">
  <request>
    <param name="search_param" type="search_param" required="true" /></ />
  <response>
    <representation mediaType="application/xml" element="itemList" />
    <fault id="searchError" status="400" mediaType="application/xml"
          element="error" />
</ /></ />
```

〈표 6〉 지역 데이터 모델 생성 알고리즘

입력 : *서비스 명세 M*
결과 : *지역 데이터 모델 트리 집합* T_M
(1) *입력 매개변수를 위한 트리* t_{params}*를 만들고* T_M*에 추가한다.*
(1-1) *이 트리에 id="params"로 속성을 지정한다.*
(1-2) *모든* $m \in M$*에 대해 m.input이 비어있지 않으면*
(1-2-1) *m.input 타입의 요소가* t_{params}*에 포함되어 있지 않으면*
(1-2-2) *m.input 타입의 최소 인스턴스 요소 트리를 만들어* t_{params}*의 child로 추가한다.*
(2) *모든* $m \in M$*에 대해 결과 요소 타입 m.output이 비어있지 않으면*
(2-1) *대응하는 최소 인스턴스 요소 트리를* t_m*이라 하면,* t_m*을* T_M*에 추가한다.*
(2-2) t_m*의 루트 <instance> 요소의 id 속성을 m.id로 설정한다.*
(2-3) *m.out에 대응하는 최소 인스턴스 요소 트리를 만들어* t_m*의 child로 넣는다.*
(2-4) T_M*에* t_m*을 추가한다.*

고리즘은 서비스 명세로부터 지역 데이터 모델을 생성하는 방법을 보여준다.

[보조정리 6] T_M의 인스턴스 트리에서 $m \in M$의 입력 데이터 *m.input* 타입의 유일한 요소가 존재한다.

(증명) 위 알고리즘에 의해 생성된 t_{params} 트리는 (1-2-2) 단계에 의해 *m.input* 타입의 요소를 포함하고 있다. 또한 같은 타입 요소는 한번만 포함되므로 유일한 요소를 구할 수 있다.

최소 인스턴스 트리는 주어진 요소 정의로부터 얻어질 수 있다. 본 논문에서는 모든 child 요소를 가지고 터미널 요소의 값은 빈 스트링을 주는 방법을 사용하였다. $M_{onelineblog}$의 서비스 명세에 대해 만들어진 데이터 모델은 <표 7>과 같다.

XForms 언어는 'submission' 요소를 이용하여 REST 서비스의 메소드 프로토타입을 선언한다. 본 논문에서는 서비스 명세의 각 메소드마다 submission 요소가 정의된다.

Submission 요소는 메소드의 주소와 이름, 입력 요소와 결과 데이터를 저장할 위치의 경로를 가지고 *s = (id, url, ref, instance)*와 같이 표시한다. 메소드가 *m = (id, url, input, output)*이라면 submission 요소의 *id*와 *uri* 속성은 *m.id*와 *m.url* 값에서 바로 얻어지는데 입력 및 결과 요소의 위치는 지역 데이터 모델로부터 구해져야 한다. <표 6>의 알고리즘에서 지역 데이터 모델을 생성한 방식에 의해 각 경로는 다음과 같이 결정될 수 있다.

- *m.input*의 요소 레이블이 lin이라면 *s.ref* = "instance('params')/lin"이 된다.
- 메소드 *m*의 아이디가 *mid*라 할 때 이 메소드의 결과 데이터가 저장될 트리는 <표 6>의 알고리즘에 의해 *mid*를 아이디로 가지므로 *s.instance* = "instance('mid')"로 표현할 수 있다.

예를 들어 $M_{onelineblog}$에 속하는 생성 메소드에 대응하는 submission 요소는 다음과 같다.

s_{create} = *(create, ... instance('params')/newItem, instance('create'))*

이상과 같이 submission 요소 *s*가 구해지면 그에 대응하는 XForms 코드가 생성될 수 있다. 입력 매개변수를 가지는 요소의 경로는 ref 속성으로 표시되고 결과 데이터가 저장될 요소의 경로는 'replace'와 'instance' 속성으로 <표 8>과 같이 표시된다.

4.3 뷰 코드의 생성

3장에서는 뷰 명세로부터 타입 포함관계를 바탕으로 뷰 간의 이동과 메소드 호출 관계를 분석하여 뷰 이동 그래프를 생성하였다. 이 절에서는 정의 3의 뷰 이동 그래프로부터 뷰와 메소드 호출, 그리고 뷰 이동을 위한 XForms 코드 생성 방법을 살펴본다.

메소드 *m = (id, uri, input, output)*에 대해 뷰 *v = (m, io, target)*와 submission 요소 *s = (id, uri, ref, instance)*가 정의되었다고 하자. 그러면 뷰 *v*가 다루어야 할 실제 데이터의 위치가 *s*에서 결정될 수 있다. 즉 *v*가 입력 뷰인 경우는 뷰가 다룰 대상 데이터가 submission의 입력 데이터인 *s.ref*와 같고 결과 뷰인 경우는 *s.instance* 경로와 같다. 이것은 뷰가 실제 참조하는 데이터의 위치로서 참조위치라고 하고 *v.ref*라고 표시한다. *s*가 메소드 *m*에 대응하는 submission 요소라면 다음 식이 만족한다.

$$v.ref = s.ref \text{ if } v = view_{in}(m),\ s.instance \text{ if } v = view_{out}(m)$$

그 뷰가 제공할 사용자 컨트롤이나 데이터의 종류는 *v.target* 타입에 의해 결정되는데 스키마 정의 타입을 이용한 인터페

〈표 7〉 서비스 입출력 타입 정보로부터 자동 생성된 모델 인스턴스 코드 부분

```
<model>
<instance id="params" namespace="">
    <search_param><author/><keyword/><date/></search_param>
    <item key=""><author/><description/><date/></item>
    <id/>
</instance>
<instance id="search" namespace="">
    <itemList/>
</instance>
<instance id="create" namespace="">
    <newItem/><author/><description/><date/></newItem>
</instance>
...
```

〈표 8〉 서비스 입출력 타입 정보로부터 자동 생성된 Submission 코드 부분

```
<submission id = "create" uri = "http://localhost:9090/onelineblog" ...
              ref = "instance('params')/newItem"
          replace="instance" instance="create"/>
```

이스 코드가 생성될 수 있다[2, 15]. 뷰의 생성은 <group> 요소를 이용하는데 여기서 id는 *m.id*를 사용하고 ref 속성은 *v.ref* 경로를 가진다.

각 뷰는 메소드 호출과 다른 뷰로의 이동을 위한 코드를 가지는데, 이것은 뷰이동그래프의 전이 에지인 d_{call}과 d_{move}에 대응한다. 단 메소드 호출 시 또는 입력 뷰로의 이동 시에 필요한 데이터가 이전 뷰의 참조위치 데이터로부터 해당 뷰의 참조위치에 복사되어야 한다. 이를 위하여 복사되어야 할 데이터를 다음 보조정리에서 보여준다.

[보조정리 7] $v_1 = (mid_1, io_1, lable_1)$, $v_2 = (mid_2, io_2, label_2)$이고 $(v_1, v_2) < d_{move}$라면 p = $v_1.ref$라 할 때 p가 가리키는 요소의 자손 중에 $label_2$ 타입의 요소가 반드시 존재한다.

보조정리 7의 부분 타입의 자손 경로를 구하는 함수를 *getpath(p, label2)*로 정의한다. 이것은 본 논문에서 가정하고 있는 각 레이블이 스키마 정의에 한 번만 등장한다는 성질에 의해 유일하게 구해진다.

[보조정리 8] <표 10>의 알고리즘에서 submit 요소의 호출을 위하여 사용될 입력 매개변수는 이 뷰가 시작될 때 *s.ref* 경로의 요소에 복사되어 있다.

본 논문에서는 copy라는 매크로 액션을 가정하는데 이것은 원본 부분트리를 목적 경로 위치의 요소에 대치하여 복사하는 것이다. 복사할 목적 요소가 존재하지 않으면 삽입하여 생성한다. 이것은 <copy ref = path1, target = path2/> 형태를 가지고 ref 속성은 원본 트리의 경로, target은 복사할 목적 요소의 경로를 가진다. XForms 표준의 몇 가지 액션 요소의 결합으로 복사 연산을 기술할 수 있으나 코드 생성과 논의의 단순화를 위하여 본 논문에서는 copy 요소를 삽입과 값의 복사를 통한 트리의 복사 연산으로 새로 정의하였다. 이 액션은 결과 인스턴스 트리에서 매개변수 트리로, 또는 반대 방향으로의 데이터 복사를 위하여 사용된다.

여기서는 가장 기본적인 응답의 처리, 즉 서버가 회신한 데이터를 지역 데이터 모델의 해당하는 부분에 저장하고 결과 뷰에서 이를 사용자에게 보여주는 기능을 자동생성에서 고려한다. 서버로부터 회신을 받은 후 클라이언트 어플리케이션은 응답의 처리를 위하여 콜백 기능을 가질 수 있다. 콜백 처리부에서 응답의 처리는 어플리케이션마다 달라지는데, XForms에서는 이러한 응답 처리 기능을 제공할 수 있도록 action 요소를 이벤트 처리부에 따라 추가할 수 있다. 그러나 추가적인 데이터 처리 기능은 어플리케이션에 따라 달라지는 부분으로 자동생성의 범위를 벗어난다.

4.4 서비스 조합 XForms 페이지 생성 시스템의 설계

이상에서 살펴본 코드 생성을 위한 모델과 코드 생성 알고리즘을 전체 시스템으로 구성하면 (그림 4)와 같다. 이 시스템의 입력은 서비스 명세인 WADL 파일과 XML 스키마 정

〈표 9〉 XForms 페이지의 〈group〉 요소 코드 생성 예

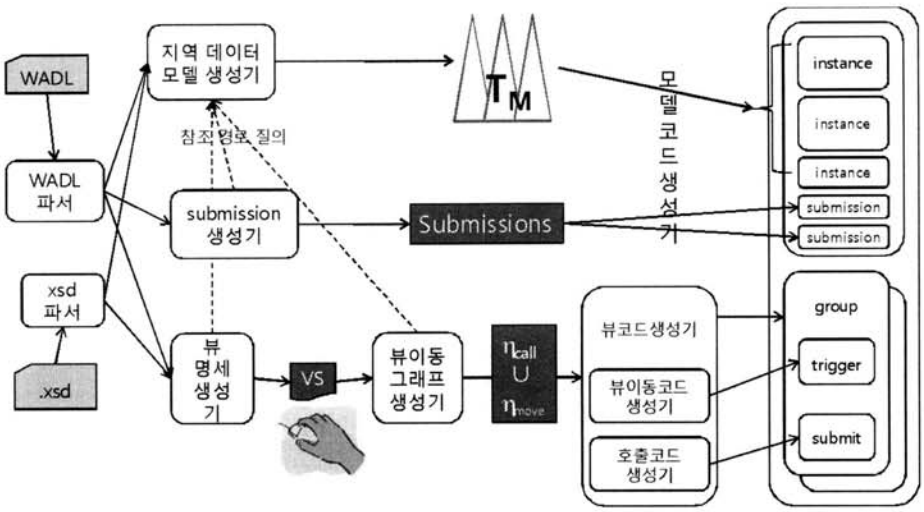
```
<group id="create" ref="instance('create')/newItem">
... </group>
```

〈표 10〉 뷰 이동 및 메소드 호출 코드 생성 알고리즘

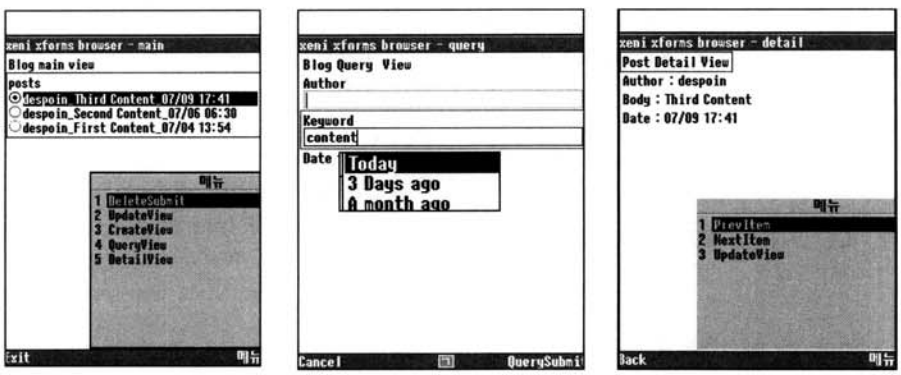
(1) v ∈ V이고 m ∈ M일 때 (v, m) ∈ δ_{call}이라면 group child 요소로 <submit id="mid"/>를 생성한다. 여기서 mid는 대응하는 submission 요소의 id이다.
(1-1) v ≠ $view_{in}$(m)이라면 π_{ref} = v.ref + getpath(v_1.ref, m.input)을 π_{target} = s.ref에 복사하는 action 요소를 <xeni:copy ref=π_{ref} target = π_{target}/>와 같이 생성한다.
(2) 뷰 v_1, v_2 ∈ V에 대해 (v_1, v_2) ∈ δ_{move}라면 group 요소의 child로 trigger 요소를 생성한다.
(2-1) trigger의 child로 <xeni:copy ref=π_{ref} target = π_{target}/>와 같이 생성한다.
(2-1-1) π_{ref} = v1.ref + getpath(v_1.ref, v_2.target), 여기서 getpath는 v_1.ref에서 v_2.target에 대응하는 부분 트리의 경로를 구한다.
(2-1-2) π_{target} = v_2.ref
(2-2) v_2로 포커스를 이동하도록 setfocus 요소를 trigger의 child로 추가한다.

〈표 11〉 자동 생성된 액션을 포함한 뷰 이동 코드

```
<trigger>
  <label>update</label>
  <action ev="dom-activate">
      <copy ref ="instance('search')/itemList/item[repeat-index('items')]"
         target="instance('params')/item"/>
      <setfocus control="update"/>
  </action>
</trigger>
```



(그림 5) 웹서비스 조합을 위한 멀티뷰 XForms 페이지 생성 시스템 구성도



(그림 6) 다양한 뷰를 가진 XForms 페이지의 실행 화면

의이다. 이 파일을 파싱하여 지역 데이터 모델과 submission 모델이 생성된다. 이들 모델로부터 XForms의 instance 및 submission 요소들이 생성된다. 한편 서비스 메소드 정의로부터 얻어진 뷰 명세는 입력 및 결과 타입으로부터 뷰 간의 관계를 추출할 수 있고 이를 바탕으로 뷰 이동 그래프를 생성한다. 뷰 이동 그래프로부터 뷰의 인터페이스, 뷰 이동, 메소드 호출 등의 코드가 생성된다.

본 연구에서 제안된 방법은 기존의 방법과 비교하여 몇 가지 장점을 가진다. 우선 여러 개의 서비스 메소드를 조합하여 하나의 클라이언트에서 지원할 수 있다. 그를 위하여 뷰를 나타내는 group 요소를 생성하여 그 안에 사용자 인터페이스 컨트롤과 trigger, submit 요소를 포함하였다. 이 방법에 의해 생성된 XForms 페이지는 서비스 명세에 포함된 메소드들을 호출하기 위한 뷰와 호출 메뉴를 모두 포함하게 된다. 두 번째로 모든 과정이 거의 자동으로 코드 생성을 할 수 있다는 점이다. 뷰 설정을 위한 사용자의 입력을 반영할 수 있으나 기본적으로 뷰 목록을 추출하고 뷰 간의 관계를 분석하여 코드를 생성하는 전 과정을 시스템에 의해 수행하는 것이 가능하다.

웹서비스를 지원하는 XForms 페이지 코드를 생성하는 기존 연구에서는 대부분 한번 submit 하면 다음 페이지로 이동하는 방식을 사용하였다. 본 연구에서 다룬 메소드 호출에 따른 뷰의 이동 방법은 여러 개의 뷰로 구성된 모바일 어플리케이션의 개발을 위하여 유용하게 사용될 수 있을 것으로 기대된다. 실제 본 연구에서 개발한 모바일 XForms 실행환경[1, 3]에서는 하나의 XForms 페이지를 읽어 여러 개의 뷰를 생성하고 뷰 간의 이동을 지원한다. 이 실행환경에서는 본 논문의 방법으로 생성된 페이지를 처리하기 위하여 웹서비스 비동기 호출, copy 연산 지원, 오류 처리뷰 지원 등을 포함한다.

본 논문에서 제안된 방법으로 생성된 구조의 XForms 파일을 이용하여 실행한 화면은 (그림 6)과 같다.

5. 결 론

여러 모바일 웹서비스를 지원하는 XForms 페이지 코드를 자동생성하는 것을 목표로 하여 본 논문에서는 뷰의 이동 모델과 코드 생성 알고리즘을 제안하였다.

제안된 방법은 서비스 명세로부터 메소드 인터페이스를 얻고 이를 분석하여 입력 및 결과 데이터 타입을 이용하여

뷰 및 뷰 간의 관계를 분석한다. 뷰 간의 관계를 분석하기 위하여 뷰에서 메소드 호출, 뷰에서 다른 뷰로의 이동을 나타내는 두 가지 종류의 전이 에지를 가지는 뷰 이동 그래프를 이용하였다. 또한 WADL 명세 언어를 이용하여 XForms 페이지 코드를 생성하는 방법을 제안하였다. WADL 서비스 명세로부터 지역 데이터 모델을 구하고 뷰 이동 그래프로부터 실제 메소드 호출과 뷰 이동 코드를 생성하는 방법을 살펴보았다. 이상과 같은 과정을 거쳐 여러 개의 뷰를 가지고 여러 서비스를 인터페이스할 수 있는 XForms 페이지 코드가 자동으로 생성될 수 있다. 또한 사용자가 설정 속성을 지정하여 뷰나 뷰 이동 코드를 최적화할 수 있도록 하였다.

제안된 방법이 실제 모바일 REST 서비스 플랫폼으로 활용될 수 있음을 보이기 위하여 본 연구팀에서 개발된 XForms 실행 환경을 이용하여 한줄 블로그 예제를 개발하였다. 이 사례를 통하여 제안된 방법의 생성 코드가 하나의 XForms 페이지에서 다양한 모바일 서비스를 인터페이스할 수 있음을 보였다.

그러나 본 논문의 방법은 몇 가지 점에서 한계를 가진다. 우선 타입 관계 분석이 단순 포함 관계 만을 이용하여 메소드 간의 관계를 잘 반영하지 못할 수 있다. 시맨틱 정보를 이용하는 최근의 서비스 조합 기술을 활용한다면 이러한 문제가 해결될 수 있을 것으로 기대된다. 한편 지역모델 생성에서 간단한 스키마를 가정하여 모든 요소를 다 가지는 인스턴스로 생성하였다. 또한 타입 관계로부터 인스턴스의 대상 경로를 찾는 방법에서도 재귀가 없는 단순한 스키마 구조를 가정하였고 모든 요소가 트리에 한 번씩만 나타나는 것으로 가정하였다. 이러한 제약은 추후의 연구 과제로 남겨져 있다.

참 고 문 헌

[1] 유가연, "오픈 API 플랫폼을 위한 XForms 브라우저의 개발", 경기대학교 석사학위논문, 2007.

[2] 이은정, 김태훈, "XForms 기반의 UI 코드 자동생성 시스템 개발", 정보처리학회논문지D, 제12-D권 제6호, December, 2005.

[3] 이은정, "XForms 페이지의 접근제어를 위한 공유 조건식의 효율적 계산 방법," 정보처리학회논문지D, 제15-D권 제4호, August, 2008.

[4] M.Brambillar, "Generation of WebML web application models from business process specifications," ICWE'06, pp.85-86, 2006.

[5] R. Cardon, et.al, "Using XForms to simplify Web programming," WWW conference, pp.215-224, 2005.

[6] S.Ceri, et al., "Model-driven development of context aware web applications," ACM TOIT Vol.7, Issue 1, Article No. 2, 2007.

[7] M.Dubinko, *XForms Essentials*. O'Reilly and Associates, 2004.

[8] J.Dunkel, R. Bruns, "Model-driven architecture for mobile applications," LNCS vol.4439, pp.464-477, 2007.

[9] R.T. Fielding, "*Architectural Styles and the Design of Network-Based Software Architectures*," doctoral dissertation, Dept. of Computer Science, Univ. of Calif., Irvine, 2000.

[10] G.Giogio, et.al, "Efficient Inclusion for a Class of XML Types with Interleaving and Counting," DBPL 2007, LNCS 4797, pp.231-245, 2007.

[11] D.A.Kateros, et.al, "A Methodology for Model-Driven Web Application Composition," SCC'08, Vol.2, pp.489-492, 7-11 July, 2008.

[12] A. Kraus, N. Koch, "Generation of web applications from UML models using an XML publishing framework," Proc. of IDPT'02, 2002.

[13] N. Milanovic, M. Malek, "Current Solutions for Web Service Composition," *IEEE Internet Computing*, pp.51-59, Dec. 2004.

[14] M. Pohja, M. Honkala, "Web user interaction - comparison of declarative approaches," WEBIST 2006, pp.295-302, 2006.

[15] A. Seffah, H. Javahery, "*Multiple user interfaces: cross-platform applications and context-aware interfaces*," J.Wiley, 2003.

[16] K. Song and K.Lee "An Automated Generation of XForms Interfaces for Web Services," IEEE International Conference on Web Services, pp.856-863, July 2007.

[17] J.Yu, et al., "A framework for rapid integration of presentation components," WWW'2007, pp.923-932, 2007.

[18] Apache group, "Axis web services," http://ws.apache.org/axis/.

[19] java.net, "Web application description language," https://wadl.dev.java.net/.XUL, http://www.mozilla.org/projects/xul/.

[20] World-Wide Web Consortium standards including XForms, XML Schema, XPath and Cascading Style Sheets. http://www.w3.org.

이 은 정

e-mail : ejlee@kyonggi.ac.kr

1988년 서울대학교 계산통계학과(이학사)

1990년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1994년~2000년 한국전자통신연구원 선임연구원

2001년~현 재 경기대학교 컴퓨터과학과 교수

관심분야 : XML 처리기술, 웹서비스, 모바일 웹기술 등