DOI: 10.3745/KIPSTD.2010.17D.5.359

REST 서비스 패턴을 이용한 매시업 클라이언트 뷰 이동 코드 생성

이 은 정†

요 약

웹 2.0의 확산과 함께 기존의 페이지 간 이동 대신 하나의 페이지에서 여러 개의 웹서비스를 인터페이스하는 웹페이지가 많이 사용되고 있다. 이러한 클라이언트 페이지를 매시업 클라이언트라고 부르는데, 이들은 복잡하고 다양한 기능을 지원하는 제어부를 포함한다. 본 논문에서는 이동 제어의 모델 기반의 코드 생성 방법을 제시한다. 먼저 REST 서비스 패턴을 클라이언트 페이지의 뷰와 뷰 이동에 적용하는 방법을 제안하고, 각 뷰로부터 서비스 메소드 호출이나 뷰 이동이 가능한 타입 조건을 제시한다. 또한 제안된 방법을 적용하여 XForms 페이지의 코드를 자동생성하는 프로토타입 시스템을 개발하였다.

이동 설계 방법을 적용한 매시업 클라이언트 페이지 생성 시스템을 구현하였다. 개발된 자동 생성 시스템은 개발자의 관여 없이 이동 제어 기능을 포함한 클라이언트 페이지의 코드를 생성하며, 체계적인 모델과 이동 패턴에 기반하여 생성된 결과 코드가 이해하기 쉽고 간단하다. 또한 사용자가 필요한 컨트롤만을 포함하여 서비스의 개수가 많아지는 경우에도 적용할 수 있다.

키워드 : 뷰이동, REST 스타일 서비스, 모델 기반 개발, XForms

Generating Mashup Client View Navigation Codes using REST Style Service Patterns

Eunjung Lee†

ABSTRACT

As web 2.0 becomes one of the important architecture styles, more web applications adopt single page structure instead of multiple web pages and navigations between pages. A single page web application client, called a mashup client in this paper, interfaces more than one services and allows users to navigate in the page. A mashup client page includes complicated functions and has to handle various styles of services and user requirements, and therefore is usually developed manually. In this paper, we propose a model driven code generation approach for in-page navigations. We propose a page model and view navigation design approach, applying REST service architecture patterns. Then, we consider type conditions for each view to have service calls or navigation controls. Also, we developed an XForms page code generation system to demonstrate the efficiency of the proposed method.

The developed system generates mashup client pages including navigation controls between services and views. This system can generate ready to use codes from service specifications, so this can help to reduce the development overhead. Moreover, our approach is based on formal model and navigation patterns so the generated result code is simple and easy to understand, and includes only the necessary controls. Therefore, the proposed approach can be more effective for the case of a large number of services.

Keywords : View Navigations, REST Service Pattern, Model Based Development, XForms

1. 서 론

웹서비스를 통한 인터넷 서비스 제공이 늘어나면서 웹페이지 개발 방법이 큰 변화를 보이고 있다. 웹서비스의 매시업이나 서비스 인터페이스를 제공하는 방식으로 웹어플리케이션이 개발되고 있다. 클라이언트 페이지가 여러 개의 서비스에 대한 인터페이스를 제공하고 사용자는 서비스 호출 순서나 서비스 간의 데이터 연결을 직접 결정할 수 있다. 이렇게 사용자 주도 매시업 기능을 제공하는 클라이언트 페이지를 매시업 클라이언트라고 부른다.

기존의 서버 기반의 페이지 이동 방식의 웹어플리케이션

※ 본 연구는 경기도의 경기도지역협력연구센터사업[컨텐츠융합소프트웨어연구센터]의 일환으로 수행하였음
† 종신회원 : 경기대학교 전자계산학과 부교수
논문접수 : 2010년 4월 1일
수 정 일 : 1차 2010년 6월 21일, 2차 2010년 8월 2일
심사완료 : 2010년 8월 3일

에서 Ajax 기술을 중심으로 하나의 페이지에서 여러 서비스를 처리하는 클라이언트로 바뀌는 추세이다[2, 3]. 클라이언트의 성능 향상에 따라 서버에서 수행하던 기능들이 점차 클라이언트로 옮겨오고 있는데, 서비스 간의 연결이나 뷰와 서비스 간의 이동 제어, 클라이언트 지역 데이터 관리 기능 등이 여기에 포함된다. 특히 뷰와 서비스 간의 이동 제어는 서비스 메소드의 수가 증가할 때 코드의 복잡도가 크게 증가하여 개발에 어려움이 있다.

이와 같이 클라이언트의 제어 구조가 복잡해지면서 자바스크립트 등의 스크립트 언어를 이용한 웹어플리케이션 개발에서 해결되어야 할 문제들이 나타나고 있다. 스크립트 언어가 가지는 모듈화 되기 어려운 구조와 실행환경의 성능상 제약 등이 많이 지적되고 있으며[1, 2], 서비스의 형태나 대상 영역이 다양하여 체계적인 개발 방법의 적용이 어렵다. 그 결과 무거운 라이브러리를 사용하거나 플러그인 등의 기능을 활용하는 경우가 많다[2, 3]. 한편 기존의 서버 중심의 웹어플리케이션 개발 방법[4-6]에 대한 연구가 많이 있으나 클라이언트에 적용하는데 어려움이 있고 클라이언트 코드의 개발 방법에 관한 연구는 아직 초보적 단계이다. 클라이언트 페이지는 영속적인 데이터와 객체를 갖지 않으며, 서비스 매시업과 제어 기능이 사용자 인터페이스를 중심으로 설계되어야 한다는 점에서 새로운 개발 방법을 필요로 한다.

한편 REST(Representational State Transfer)는 R. Fielding이 제안한 아키텍처의 개념으로 느슨한 결합의 클라이언트 서버 구조를 지칭하는데[7], 최근에는 웹서비스 어플리케이션의 아키텍처 스타일을 가리키는 용어로 사용된다. REST 스타일 웹서비스는 가벼운 웹서비스 프로토콜로서 HTTP의 표준 메소드 종류만 사용하고 유형화된 데이터 타입을 가지며, 서비스나 리소스의 주소 지정 방식, 서비스의 구성 형태 등에 대해 표준화된 패턴을 제시하여 클라이언트 설계를 위한 유익한 모델을 제공한다[8, 9].

본 논문에서는 표준적인 서비스 메소드 유형으로서 REST 스타일의 데이터 모델과 사용 패턴을 활용하여 웹서비스의 다양성과 클라이언트 제어 기능의 복잡성을 해결하여 자동 코드 생성이 가능한 방법을 찾고자 한다. 이를 위하여 매시업 클라이언트에 맞는 페이지 모델과 이동 연산을 제안하고 이를 바탕으로 이동 연산의 설계 및 자동생성을 위하여 가능한 이동 집합을 뷰와 메소드의 데이터 타입으로부터 계산한다. 다음으로 가능한 이동 연산의 집합 중에서 필요한 것을 선택하기 위하여 REST 스타일의 아키텍처 패턴을 적용하는 설계 방법을 제시한다.

설계된 이동 연산 및 이동 관계 모델을 바탕으로 클라이언트 페이지의 코드 생성기를 개발하였다. 개발된 시스템은 WADL 명세 언어로 기술된[10] 서비스 명세로부터 XForms [11] 페이지를 생성한다. 개발된 시스템은 프리젠테이션과 통신부 뿐 아니라 이동과 사용자 인터페이스를 포함하는 매시업 클라이언트 페이지를 자동으로 생성하여 기존의 클라이언트 개발 환경 프레임워크와 차별성을 가진다. 생성된 결과 페이지는 이동을 위한 최소한의 메뉴를 가지므로 지원하는 서비스의 수가 증가할 때 사용자 컨트롤 및 코드 길이가 선형 비례하여 증가한다. 또한 사용자가 바로 사용 가능한 코드를 얻을 수 있어 동적인 서비스 환경에서 사용자의 필요에 따라 페이지를 동적으로 생성 가능하다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를 살펴보고, 3장에서는 클라이언트 페이지 모델과 REST 스타일의 서비스 패턴 및 서비스 명세를 소개한다. 4장에서는 이동 및 매시업 설계 방법을 소개하고, 5장에서는 구현된 코드 생성 시스템과 결과에 대해 살펴본다. 6장에서 결론을 맺는다.

2. 관련 연구

웹 2.0 이전의 웹어플리케이션 개발에서 웹페이지는 서버에서 생성되고 요청의 전송과 함께 생명 주기가 종료하여 웹페이지 내에서의 이동이나 제어는 큰 비중을 차지하지 않았다. 이 경우 XML 스키마 정의[11]와 웹서비스 명세 언어를 이용하여 통신부나 프리젠테이션 코드를 생성하는 기술이 웹페이지의 개발을 위해 실용적으로 사용되고 있다. 대표적인 예가 아파치 axis의 WSDL2JAVA와 마이크로소프트의 InfoPath 등이다[12-15].

한편 서비스 중심 아키텍처의 개발 프레임워크로서 마이크로소프트의 닷넷 기술은 WCF(Windows Communication Framework)를 이용한 웹서비스 통신 및 처리 기능과 WF(Windows Workflow Framework)를 이용한 액티비티 기반의 서비스 개발 프레임워크를 제공한다[4]. 이 기술은 닷넷 환경에서 웹서비스 어플리케이션 개발을 위한 플랫폼으로서 SOAP 프로토콜에 기반한 코드 생성 기술을 활용하고 있다. 닷넷의 공통 타입을 기반으로 하는 서비스 연결 기능을 제공하는데, 서버 기반의 제어 기능으로서 사용자 주도의 매시업이나 서비스 간 이동을 지원하는 클라이언트의 개발에 그대로 적용하기 어렵다. 또한 이 기술은 닷넷 환경에 의존적이고 무거워서 사용에 제약이 있다. 한편 IBM의 WebSphere는 J2EE 클라이언트의 개발과 서버 기반의 서비스 중개 기능을 제공한다[5]. WebSphere의 클라이언트 개발은 서버에 필요한 인터페이스 제공 기능에 한정되어 데이터 처리와 이동에 관한 처리는 서버에서 이루어진다. 이상에서 살펴본 바와 같이 현재의 웹어플리케이션 개발 환경은 서버 기반의 서비스 및 페이지 이동을 다루며, 클라이언트에 대해서는 프리젠테이션과 통신부 코드만 생성하고 있다. 그러므로 클라이언트 페이지 내의 이동이나 매시업의 제어부 개발을 위해서는 새로운 방법이 제시되어야 한다.

웹 어플리케이션 모델 기반의 페이지 이동 설계는 많이 연구된 분야이다[16, 17]. Winckler 등은 뷰 통합과 배치를 데이터 타입을 통해 해결하는 설계 및 개발 방법을 제시하였다[17]. Guell 등은 요구사항 명세로부터 웹어플리케이션의 사용자 상호작용과 페이지 이동을 모델링하는 방법을 제안하였다[18]. 컨텐츠 모델을 이용하여 개발자가 이동 기능

을 설계하도록 지원하는 방법도 제안되었다[19]. Narad 등은 사용자의 관심을 바탕으로 네비게이션을 최적화하는 방법을 제안하였다[6]. 워크플로우 및 데이터베이스 분야에서도 네비게이션에 대한 연구가 있었다. 워크플로우 프로세스의 분석과 컨텐츠 및 작업 모델로부터 네비게이션을 구하는 방법이 제안되었다[20, 21].

한편으로 공개 API 형태의 웹서비스를 기반으로 메소드를 조합하여 새로운 서비스를 만드는 매시업이 사용자 주도 웹프로그래밍의 형태로 주목받고 있다[22, 23]. 서비스의 매시업은 데이터의 통합과 연결이 이루어지는 장소에 따라 서버 또는 클라이언트 측 매시업으로 구분할 수 있고 데이터의 통합 또는 서비스 간의 연결을 위주로 하는 경우가 있다. 최근 클라이언트에서 사용자 주도의 매시업을 위한 개발 및 설계 도구가 많이 발표되었다[23-25]. 서비스 매시업 자바스크립트 코드의 안전성에 대한 연구도 활발하다[26].

클라이언트의 개발 방법에 관한 연구로서는 Ajax를 이용한 클라이언트 개발 방법[3, 27]에 대한 연구가 있다. Mesbah 등은 Ajax 기반 페이지의 아키텍처 구조와 성능을 조사 분석하였으며, 서비스 인터페이스를 위한 제어 논리의 기본 구성을 제시하였다. Liu 등은 클라이언트에서의 서비스 흐름을 설계하는 모델로 FSM을 이용한 방법[28]을 제시하였다. 이들의 연구에서 FSM은 데이터와 서비스의 호출 및 회신 처리를 설계하며, 사용자 인터페이스에서 뷰와 서비스 간의 이동 연산은 다루지 않는다. Benson 등은 클라이언트의 지역 데이터의 캐싱 및 서버와의 동기화 기법을 제안하였다[29]. 이들의 연구는 연속되는 서비스 호출에 대하여 클라이언트에서 결과 데이터를 유지 관리하는 기법을 다루는데, 본 연구에서는 단순한 데이터 관리 기법을 가정하고 뷰와 서비스에 대한 이동 제어를 포함하는 사용자 인터페이스의 개발 방법을 살펴본다.

3. 클라이언트 시스템 모델

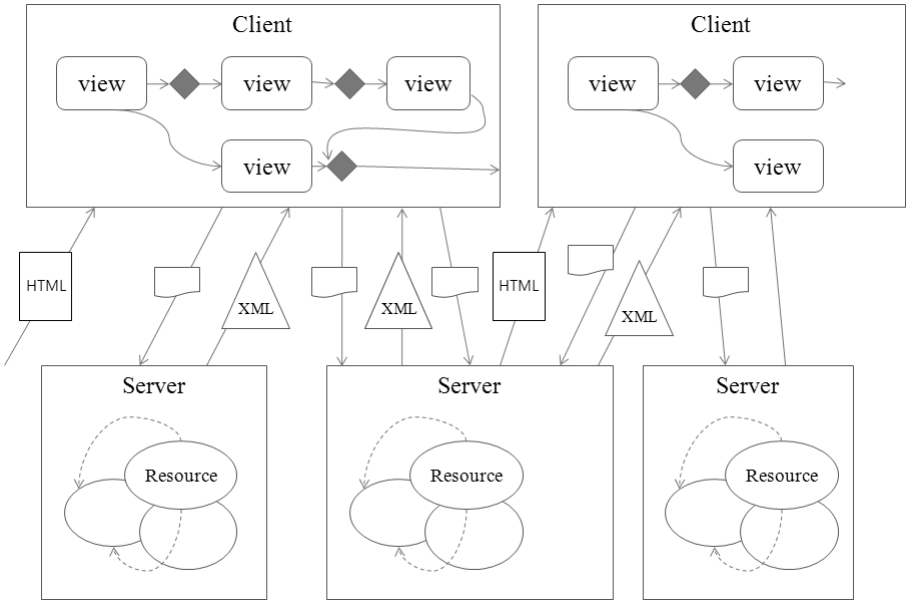
이 장에서는 독자적으로 여러 개의 서비스를 인터페이스하면서 메시지를 송수신하고 데이터를 제어하는 매시업 클라이언트 페이지의 유형과 페이지 모델을 소개한다. 또한 REST 스타일 서비스 메소드의 표준적인 형태와 매시업 페이지에서 지원할 서비스 메소드 집합의 정의 방법을 살펴본다. 마지막으로 본 논문에서 주 사례로 사용될 서비스 시나리오를 소개한다.

3.1 매시업 클라이언트 페이지

매시업 클라이언트 페이지는 서버와 독립적으로 동작하면서 서비스 호출과 회신 결과 데이터 처리, 여러 서비스 간의 흐름과 연결을 지원해야 한다. (그림 1)은 이러한 페이지의 구조를 보여준다. 하나의 HTML 페이지는 독자적으로 서비스 호출과 결과 데이터 처리를 담당하며, 이를 위하여 여러 개의 뷰와 서비스 메소드 간의 흐름과 이동을 제어한다.

매시업 클라이언트 페이지의 기능적 요구사항은 다음과 같다.

- 사용자는 각 서비스 메소드에 대해 매개변수를 입력할 수 있고 서비스 요청, 그리고 회신된 결과를 확인할 수 있어야 한다.
- 서비스 집합의 결과 데이터를 저장할 지역 데이터를 유지해야 한다.
- 사용자는 서비스와 뷰를 접근하는 이동 연산을 사용할 수 있어야 한다.
- 서비스 호출의 결과 데이터로부터 다른 서비스 호출 입력 매개변수가 연결될 수 있는 경우 사용자가 선택할 수 있어야 한다. (사용자 주도 매시업)



(그림 1) 매시업 클라이언트 페이지의 뷰 구성과 서비스 호출

웹서비스 데이터 타입으로부터 클라이언트의 입력 및 결과 뷰의 프리젠테이션 코드나 통신부 코드를 자동 생성하는 프레임워크가 알려져 있다[14, 15]. 또한 서버 기반의 서비스 흐름 제어와 페이지 이동의 개발 방법도 사용되고 있다[3, 4]. 그러나 클라이언트 페이지의 개발은 앞에서 살펴보았듯이 서버의 개발 방법을 그대로 적용하기에는 어려움이 있다. 본 논문에서는 위와 같은 기능적인 요구사항을 만족하는 사용자 인터페이스와 클라이언트 제어 코드를 설계 개발하는 방법을 살펴본다.

페이지가 여러 개의 뷰를 가지는 경우 뷰의 배치를 고려해야 한다. 보통 배치는 플랫폼에 의존적이어서 데스크탑 컴퓨터에서는 스크린 영역을 나누어 여러 개의 뷰를 표현하는 반면, 모바일 단말에서는 화면 크기가 작아서 한 번에 하나의 뷰만 보여진다. 본 논문에서는 문제를 간단하게 하기 위하여 뷰의 배치는 고려하지 않는다.

3.2 타입을 가진 REST 서비스 명세

보통 웹서비스 명세는 서비스 제공자에 의해 작성되는데 매시업 클라이언트 페이지가 지원할 메소드 집합을 모아서 따로 명세할 수 있다. 예를 들어 주유소에서 주유, 정비, 음료와 사용자설문 같은 임시 서비스와 주변 지리 정보 서비스가 지역적으로 제공되는 경우를 고려해 보자. 사용자가 원하는 서비스 집합을 선택하면 클라이언트 페이지가 동적으로 생성되어 사용자에게 제공되어야 한다. 이 절에서는 클라이언트의 관점에서 REST 스타일 서비스 집합을 정의하고 이를 표현하는 방법에 대해 살펴본다.

표준적인 REST 스타일 서비스의 특징은 잘 정의된 메소드 유형과 데이터 타입을 가진다는 점이다. REST 스타일 서비스에서 다루어지는 데이터 타입(또는 카테고리)을 리소스라고 부르고 리소스의 개별 데이터는 인스턴스라 부른다. 각 리소스에 대해 고유한 주소값(url, universal resource location)이 부여되고 인스턴스에는 아이디가 부여된다. 예를 들어 일정 리소스와 인스턴스는 각각 ../Schedules 와 ../Schedules/{id} 형태의 url로 표현된다. 여기서 url 구조는 서비스의 논리적인 아키텍처를 표현한다. 한편 REST 스타일 웹서비스의 각 메소드는 표준적인 HTTP 동사(GET, PUT, DELETE, POST)만을 이용하며 검색, 조회, 수정, 삭제, 추가의 연산 형태로 표현된다. 이를 여기서는 O_{SCRUD} = *{Search, Read, Update, Delete, Create}*로 표현한다. 메소드는 O_{SCRUD} 의 타입에 리소스 r을 결합하여 *Read(r)*과 같은 형태로 표현한다. *각* 메소드는 입력 및 결과 데이터 타입과 의미가 잘 정의되어 있다. 인터넷 상의 공개 서비스 정의 파일은 결과 데이터 형식이 XML 포맷을 사용하는 경우 공개 네임스페이스와 스키마 정의 타입을 사용하여 결과 데이터 타입을 정의하고 있다. 입력 데이터 타입의 경우도 기본 데이터 타입(문자열, 숫자, 날짜, 아이디)을 주로 사용하며 많은 경우 스키마 정의 타입으로 정의되거나 대응시킬 수 있다[7, 23, 24]. 본 논문에서는 모든 입력 및 결과 데이터가 스키마에 의해 미리 정의된 타입을 가진다고 가정한다. 이것을 *타입을 가진 REST 서비스*라고 부른다.

<표 1>은 리소스 *r*에 대해 O_{SCRUD} 연산 메소드의 HTTP 동사와 url 형식을 보여준다. 각 메소드의 표준적인 REST 메소드와 입력 및 결과 데이터 타입이 나타난다. 리소스 인스턴스의 아이디는 *id(r)*로 표현된다. 검색 메소드는 다양한 입력 매개변수 형식을 가질 수 있는데 다른 리소스 타입의 아이디나 키워드 스트링이 많이 쓰인다. 또한 검색 요청의 결과 타입은 *list(r)*로 표현되는데 이것은 일반적으로 아이디와 몇 가지 데이터를 포함하는 요약정보 쌍이 반복되는 타입이다. 수정과 생성 연산은 수정 또는 추가된 레코드의 정보를 돌려받는다. 삭제는 회신되는 결과 데이터 없이 성공/실패만 돌려받는다. 이러한 일반적인 메소드 유형은 어플리케이션에 따라 달라질 수 있다.

[정의 1]은 서비스 메소드를 정형적으로 정의한다.

[정의 1] 타입을 가지는 서비스의 집합을 M, 데이터 타입 집합을 T라 하자. 리소스 r에 대한 서비스 메소드 $m \in M$는 m = *(t, u, X, y)* 과 같이 정의된다. 여기서,

t : 메소드 타입, $t \in O_{SCRUD}$
u : 메소드 m의 url
X : 메소드 m의 입력 매개변수 타입의 집합, $X \subset T \cup \{\perp\}$
y : 결과 데이터 타입, $y \in T \cup \{\perp\}$

메소드 m = *(t, u, X, y)*에 대해 X는 in(m), y는 out(m)이라고 표시한다. 위에서 ⊥은 널 데이터 타입, 즉 데이터가 없음을 뜻한다. 리소스 *r*에 대한 메소드의 집합을 M^r이라 하고 T^r이 M^r에서 사용되는 타입의 집합이라 할 때 T^r는 다음과 같이 나누어 볼 수 있다.

$$T^r = \{r, id(r), list(r)\} \cup T^r_{Search}$$

〈표 1〉 리소스 r에 대한 REST 방식의 표준 메소드

메소드 타입	HTTP 동사	url 형식	입력 매개변수 타입	결과 타입
Search	GET	http://.../resources/	search parameters	list(r)
Read	GET	http://.../resources/{id}	id(r)	r
Update	PUT	http://.../resources/{id}	id(r), r	r
Create	POST	http://.../resources/	r	r
Delete	DELETE	http://.../resources/{id}	id(r)	⊥

여기서 T^r_{Search}는 검색 메소드 Search(r)의 입력 데이터 타입의 집합이다. 검색 요청의 입력 파라매터 타입은 어플리케이션에 따라 다양하다.

본 논문에서는 타입간의 포함관계를 이용하여 매시업을 계산하는데, 포함관계는 $\gg$ 기호를 사용하여 나타낸다. 타입 포함관계는 스키마 정의 상에서의 조상/후손 관계로 계산한다. 그러므로 타입 t_1이 t_2의 조상이면 $t_1 \gg t_2$로 표시한다. 또한 타입의 집합 $X \subset T$에 확장하기 위해 $t_1 \gg t_2$이고 $t_2 \in X$이면 $t \gg X$라고 표시한다. 반면에 $X \gg t$가 만족되려면 모든 $t_1 \in X$에 대해 $t_1 \gg t$가 만족하여야 한다. 이것은 X가 가능한 타입들의 조합으로 OR 조건을 나타내는 검색 입력 타입일 때 사용된다. 여기서 사용된 타입 포함관계를 터미널 데이터 타입의 포함관계 등 다른 타입 관계식으로 대치하여 사용할 수 있다.

3.3 사례 시나리오

REST 스타일 서비스는 리소스에 대한 표준 메소드를 제공하고 리소스는 필드에 의해 상호 참조된다. 리소스 간의 관계는 ER 다이어그램과 같이 상호 참조하는 필드로 표시된다. 본 논문에서는 팀원의 일정과 회의실, 교통정보 리소스를 포함한다.

REST 서비스 명세 언어의 표준으로 IBM 등에서 제안한 WADL(Web Application Descripton Language)이 사용되고 있다[10]. WADL은 리소스의 url과 메소드의 종류, 각 메소드의 입력과 결과 데이터 타입, 데이터 형식 등을 지정할 수 있다. WADL 표준은 서버에서 제공되는 서비스를 명세하기 위한 언어로 등장하였으나 여러 개의 리소스와 url을 표시할 수 있으므로 클라이언트의 서비스 메소드 집합 명세로도 사용될 수 있다. 본 논문에서는 결과 데이터 타입이 XML 형태이고 모든 매개변수와 결과 데이터의 타입은 스키마 네임스페이스에 의해 정의된다고 가정한다. 인터넷 상에 공개된 WADL 파일에서 결과 데이터는 <grammar> 요소에서 지정한 스키마 정의 타입을 사용하고, 입력 데이터도 타입을 대응시킬 수 있다면, 타입을 가진 웹서비스 명세 언어라고 볼 수 있다.

한편 사례 시나리오에서 사용되는 서비스 메소드 집합을 <표 2>에서 요약하고 있다. 리소스 집합 R = {Schedule, Person, Room, Bus}이고, 각 리소스에 대해 schd, schd, rm, bus 네임스페이스 머리말을 사용한다. Schedule에 대한 검색, 읽기와 수정 함수, Person, Room과 Bus에 대한 검색, 읽기 메소드를 가진다. 결과 및 입력 매개변수 타입은 이동 관계의 분석에서 중요한 역할을 한다.

Person의 아이디 sch:pid 타입이 Schedule 및 Room의 검색에 입력 매개변수로 사용됨을 알 수 있다. 또한 Schedule의 필드인 schd:pid는 Person 데이터의 읽기 요청을 위한 매개변수가 될 수 있다. 이전 요청의 결과를 다음 요청의 입력으로 연결하는 것이 사용자에 의한 매시업이고 이러한 관계를 찾아내어 사용자 인터페이스에 반영하는 것이 본 논문에서 다루는 뷰 이동 설계의 목적이다.

4. 이동 연산의 설계 방법

이 장에서는 본 논문의 주요 목표인 이동 제어의 설계를 위해 뷰와 메소드 간의 이동 관계를 분석하는 방법을 소개한다. 또한 구해진 이동 관계에서 클라이언트 페이지에 필요한 이동 연산을 선택하기 위하여 REST 스타일 서비스 패턴을 적용한 알고리즘을 제시한다.

클라이언트 페이지 P에 대해 이 페이지에서 제공하는 서비스 메소드 집합을 M이라 하자. 그러면 뷰와 이동 연산을 이용하여 페이지를 정의할 수 있다. 즉 웹페이지 $P = (M, V, C)$로 표시되며, 여기서 V는 페이지를 구성하는 뷰의 집합, C는 뷰 및 메소드 간의 이동 연산의 집합이다.

페이지를 구성하는 단위인 뷰는 메소드에 대한 입력 및

〈표 2〉 시나리오 서비스 메소드 집합

번호	url	HTTP 메소드타입	메소드타입	입력 매개변수 타입	결과 타입
1	schedules/	GET	Search	null xsd:string schd:date schd:pid	schd:slist
2	schedules/{sid}	GET	Read	schd:sid	schd:schedule
3	schedules/{sid}	PUT	Update	schd:schedule	
4	schedules/	POST	Create	schd:schedule	
5	people/	GET	Search	null xsd:string	schd:plist
6	people/{pid}	GET	Read	schd:pid	schd:person
7	rooms/	GET	Search	null schd:pid xsd:string	rm:rlist
8	rooms/{rid}	GET	Read	rm:rid	rm:room
9	buses/	GET	Search	xsd:string	bus:blist
10	buses/{bid}	GET	Read	bus:bid	bus:bus

결과 뷰가 있으며, 각각을 메소드 $m \in M$에 대해 $view_{in}(m)$과 $view_{out}(m)$라고 표시한다. 입력 뷰는 입력 매개변수를 편집하고 요청을 전송하는 기능을 제공하고 결과 뷰는 결과 데이터를 확인하고 다른 뷰나 요청으로 연결하는 이동 기능을 제공한다. 모든 메소드에 입력 및 결과 뷰가 필요한 것은 아니며, 필요한 뷰를 결정하기 위해 다음과 같은 패턴을 적용한다.

i) 입력 매개변수가 없거나 편집가능하지 않은 데이터 타입이면 입력뷰는 필요하지 않다. 읽기 전용 값, 특히 서버에서 사용되는 아이디나 키 값 등이 여기에 속한다.
ii) 결과 데이터가 없으면 결과 뷰는 필요하지 않다.
iii) 동일한 결과 데이터 타입은 같은 결과뷰를 공유한다.

그러므로 메소드 타입의 집합 O_{SCRUD}에 대해 <표 3>과 같이 $VIEWS = \{IS, IC, IU, OS, OR\}$의 다섯 가지 뷰 타입이 필요하다.

뷰는 뷰 타입에 리소스를 결합하여 표현한다. 예를 들어 리소스 r의 검색 요청을 위한 입력 뷰는 $IS(r)$로 표시한다. 리소스 r에 대한 모든 뷰의 집합은 $VIEWS(r)$로 나타내고, 모든 리소스 집합을 R이라 할 때 모든 뷰의 집합은 $VIEWS(R)$로 나타낸다. 페이지를 구성하는 뷰의 집합은 $V \subset VIEWS(R)$로 표시할 수 있다. 한편 V 중에서 입력/출력 뷰 집합을 V_{in}/V_{out}이라고 표시하며 $V = V_{in} \cup V_{out}$을 만족한다.

페이지를 구성하는 이동 연산을 모델링하기 위하여 메소드 호출과 뷰 간의 관계를 살펴본다. 입력 뷰는 메소드 호출을 위한 매개변수를 준비하고 호출을 요청하는 컨트롤을 가지게 된다(C_{call}). 메소드 호출의 결과가 회신되면 이것을 사용자가 확인할 수 있도록 결과 뷰를 활성화시켜야 한다($C_{callback}$). 다음으로 다른 메소드의 기능을 이용하기 위해 해당 뷰로 포커스를 이동시키는 연산이 가능해야 한다(C_{move}). 이상의 관찰 결과를 바탕으로 페이지의 이동 연산 집합을 다음과 같이 정의한다.

〈표 3〉 REST 메소드 타입에 대해 필요한 뷰 타입

메소드 타입	입력뷰 타입	입력 매개변수	결과뷰 타입	결과 데이터
Search	IS	T^r_{Search}	OS	list(r)
Read	-	{id(r)}	OR	r
Create	IC	{r}		
Update	IU	{r}		
Delete	-	{id(r)}	-	

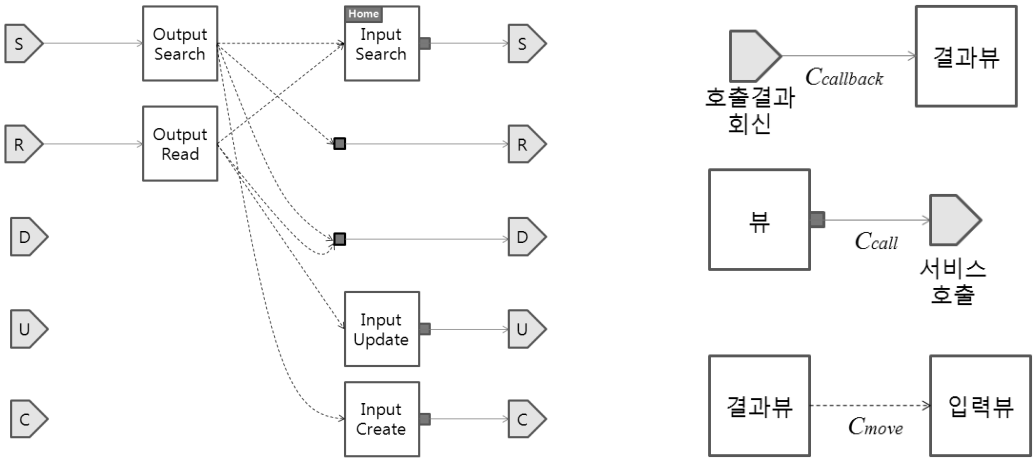
[정의 2] M이 메소드 집합이고 T는 M을 정의하는데 사용되는 데이터 타입의 집합이라고 하자. 웹페이지 $P = (M, V, C)$가 뷰의 집합 V에 대해 정의될 때 이동 연산의 집합 C는 다음과 같이 정의된다.

$C = C_{call} \cup C_{move} \cup C_{callback}$, 여기서
(1) 뷰에서 메소드를 호출하는 연산 집합 $C_{call} \subset V \times M \times T$,
(2) 뷰에서 뷰로 포커스 이동하는 연산의 집합 $C_{move} \subset V \times V \times T$, 그리고
(3) 메소드 호출의 회신에 대한 결과 뷰 이동 연산의 집합 $C_{callback} \subset M \times V \times T$이다.

(그림 2)는 이러한 이동 연산의 종류를 다이어그램으로 표시한 것이다. 각 이동 연산은 데이터를 전달하는데, 전달될 데이터 타입을 이용하여 뷰와 메소드 간의 이동 연산 조건을 구할 수 있다. 먼저 뷰가 가지는 데이터의 타입을 다음과 같이 정의한다.

[정의 3] 메소드 m $\in M$에 대해 뷰 v의 데이터 타입 $type(v)$는 다음과 같이 정의된다.

$type(v) = in(m), \ v \in V_{in}$일 때,
$= \{out(m)\}, \ v \in V_{out}$일 때.



(그림 2) 이동연산의 종류와 기본 패턴

뷰의 데이터 타입은 그 뷰가 가져야 할 데이터를 나타내는데, 입력 뷰는 해당 메소드의 입력 매개변수 타입 집합이고 결과 뷰는 보여주어야 할 회신된 데이터의 타입으로 한 개의 요소만 가진다. 한편 뷰가 활성화되기 위해서 필요한 데이터가 있다. 예를 들어 수정 뷰를 활성화하려면 수정할 인스턴스의 데이터가 준비되어 있어야 한다. 이를 뷰의 *전달데이터*라 하고 $pre(v)$로 표시한다. 전달데이터는 연결되는 호출 함수의 매개변수인데, 수정 입력 뷰의 경우 특별하게 리소스 인스턴스 데이터가 있어야 수정이 가능하다. 뷰와 전달데이터 타입을 이용하여 가능한 이동 연산을 다음과 같이 정의한다.

[정의 4] 뷰 v와 메소드 m, 그리고 타입 t에 대해 가능한 이동 연산 $c \in C$는 다음의 세 가지 유형 중 하나이다.

(1) 뷰 이동 연산은 $c = (v, v', t) \in C_{move}$ 로 표시되며, 이 때 $t = pre(v')$이다.
(2) 메소드 호출 $c = (v, m, t) \in C_{call}$로 표시되며 $t \in in(m)$이다.
(3) 컬백 이동 $c = (m, v, t) \in C_{callback}$로 표시되며 $t \in out(m)$이고 $v = view_{out}(m)$이다.

서비스 요청을 위해 사용자 작업이 필요한 경우 해당 메소드의 입력 뷰로 이동해야 하는데, 이 때 서비스 메소드 호출의 결과를 다른 호출의 입력에 연결시키는 매시업이 가능하다. 그러므로 $C_{move} \subset V_{out} \times V_{in} \times T$를 만족한다. 전달되는 데이터가 없는 뷰 이동은 여기서는 고려하지 않는다. 한편 (2)의 메소드 호출은 입력뷰에서 그 뷰에 해당하는 메소드를 서브미트하는 연산으로, $c_{call} = (v, m, t)$는 $v = view_{in}(m)$를 만족한다.

메소드 m은 그 메소드의 입력 뷰인 $view_{in}(m)$에서 요청할 수 있지만 입력 매개변수 데이터를 포함하는 다른 뷰에서도 가능하다. (그림 2)에서 Output Search 화면으로부터 delete 연산을 바로 호출하는 경우가 그러한 예이다. 메소드 결과 데이터가 다른 메소드 요청에 필요한 입력 매개변수를 가지고 있다면 입력창을 거치지 않고 해당 메소드 요청으로 바로 연결될 수 있다. 이것이 서비스 연결에 의한 매시업을 제공하는 사용자 컨트롤을 제공하는 형태이다. 즉 서비스 메소드 $m \in M$과 m의 입력 매개변수 타입 t $\in$ in(m)에 대해 $type(v) \gg t$가 만족하면 메소드 호출 이동 연산 $c_{call} = (v, m, t) \in C_{call}$가 가능하다.

정리 1은 데이터 타입을 이용하여 가능한 이동 연산의 종류와 조건을 제시한다.

[정리 1] 메소드 집합 M과 뷰 집합 V, 이동 연산의 집합 C에 대해 페이지 P가 P = (M, V, C)이라고 하자. 또한 $m \in M$, $V = V_{out} \cup V_{in}$, $C = C_{call} \cup C_{move} \cup C_{callback}$라고 하자. 그러면 가능한 연산의 집합 C가 다음과 같이 주어졌을 때 $C \subset C$를 만족한다. 즉 $C = C_{call} \cup C_{move} \cup C_{callback}$는 다음과 같이 얻어진다.

(1) $C_{move} = \{(v_1, v_2, t) \mid v_1 \in V_{out}, v_2 \in V_{in}, type(v_1) \gg t = pre(v_2)\}$,
(2) $C_{call} = \{(v, m, t) \mid t \in in(m)$이고 i) $v = view_{in}(m)$거나 ii) $v \in V_{out}$이고 $type(v) \gg t\}$,
(3) $C_{callback} = \{(m, v, t) \mid v = view_{out}(m), t = out(m)\}$.

정리 1은 동일한 리소스에 대한 메소드와 뷰의 이동 연산에 대해 적용할 수 있는 조건이다. 그러나 페이지가 하나 이상의 리소스를 포함하는 경우 가능한 이동 연산을 모두 포함하면 이동관계가 지나치게 복잡해진다. 예를 들어 입력 매개변수가 없거나 $pre(v) = \varnothing$인 뷰는 모든 뷰에서 이동이나 호출 연산이 가능해진다. 그러므로 다른 리소스의 뷰나 메소드로의 이동 연산에 대해서는 최소한으로 필요한 경우를 설계해야 한다. 이를 위해 리소스 간 이동 연산을 정의한다. R이 리소스의 집합이고 두 개의 리소스 $r_1, r_2 \in R$에 대해 m이 r_2의 메소드라고 하자. 그러면 리소스 간 메소드 호출 c_{inter}는 다음과 같이 표현된다.

$$c_{inter} = (r_1, m, t),\ t \in T^C.$$

가능한 리소스 간 이동 관계는 정리 1과 비슷하게 리소스 데이터 타입으로부터 구할 수 있다. 타입 t가 메소드 호출의 입력 데이터라면 $r_1 \gg t$를 만족하여야 한다. 이것은 리소스 r_1에서 다른 리소스의 메소드의 입력 매개변수를 포함하는 경우 바로 호출할 수 있음을 의미한다. 예를 들면 일정 상세정보에서 참석자의 아이디를 통해 그 사람의 상세정보를 읽어 올 수 있다.

(그림 2)의 사례 시나리오에서는 다음과 같은 타입 관계를 얻을 수 있다.

type(IS(Schedules)) = {⊥, id(Person), date},
type(IS(Person)) = {⊥, department, id(Person)},
Schedule ≫ id(Person), Schedule ≫ id(Room), Schedule ≫ date.

그러므로 다음과 같은 가능한 리소스 간 메소드 호출 연산을 얻을 수 있다.

C_{inter}(R) = {(Schedule, Read(Person), id(Person)), (Schedule, Read(Room), id(Room)), (Person, Search (Schedule), id(Person))}

리소스 간 이동 관계를 다음과 같이 설계하였다. 주어진 뷰가 다른 리소스의 읽기나 검색 호출에 필요한 데이터를 포함한다면 해당 이동 연산은 추가되는 것이 유용하다. 그 이외의 연산은 각 리소스 별로 홈뷰를 두고 홈뷰를 거쳐 사용자의 접근을 보장한다. 각 리소스의 검색 입력뷰가 홈뷰

(리스트 2) 이동 연산 계산 알고리즘

[알고리즘 1] 이동 연산 계산
입력 : 메소드 집합 M, 데이터 타입 집합 T
결과 : 페이지를 구성하는 뷰의 집합 V, 이동 연산의 집합 C
1) 각 메소드 m ∈ M에 대해,
  1-1) in(m) = {t | t ∈ T, m의 입력 데이터 타입에 대응하는 스키마 타입 요소},
  1-2) out(m) = t, t ∈ T, m의 결과 데이터 타입에 대응하는 스키마 타입 요소
  1-3) m의 메소드 타입이 Search, Create, Update이면
    1-3-1) m의 입력뷰를 V에 추가 // IS, IC, IU 타입의 뷰를 추가
    1-3-2) 아니면 m의 입력뷰는 없음.
  1-4) m의 메소드 타입이 Search 또는 Read이면
    1-4-1) m의 결과뷰를 V에 추가, // OS, OR 타입의 뷰를 추가
    1-4-2) 아니면 m의 결과뷰는 없음.
2) 입력뷰를 가지는 모든 메소드 m에 대해, v가 m의 입력뷰라 하면,
  2-1) (v, m, in(m))를 C_{call}에 추가. // v에서 in(m)를 매개변수로 m 호출
3) 모든 m_1, m_2 ∈ M에 대해, $m_1 \neq m_2$이면서 동일한 리소스에 대한 메소드인 경우,
  3-1) v_1= $view_{out}(m_1)$의 결과뷰이고, ∀ t ∈ in(m_2) s.t. out(m_1) ≫ t,
    3-1-1) (v_1, m_2, t)를 C_{call}에 추가 // v_1에서 t를 매개변수로 m 호출
4) 모든 결과뷰 v_1과 입력뷰 v_2에 대해, 해당 메소드 m_1, m_2가 동일한 리소스에 대한 메소드일 때,
  4-1) m_1의 메소드 타입이 Read이고 m_2의 메소드 타입이 Update이면,
    (v_1, v_2, out(m_1))을 C_{move}에 추가, // OR -> IU로 이동 연산
  4-2) m_1의 메소드 타입이 Search이고, m_2의 메소드 타입이 Create 이면,
    (v_1, v_2, ⊥)를 C_{move}에 추가 // OS -> IC로 이동연산
  4-3) m_1의 메소드 타입이 Search가 아니고, m_2의 메소드 타입은 Search인 경우
    (v_1, v_2, ⊥)을 C_{move}에 추가. // 모든 결과뷰에서 해당 리소스의 홈뷰로 이동
5) m_1, m_2 ∈ M에 대해 m_1과 m_2가 다른 리소스에 대한 메소드인 경우,
  5-1) m_1의 메소드 타입이 Read이고 v = $view_{out}(m_1)$이면
    ∀ t ∈ in(m_2) s.t. out(m_1) ≫ t, (v, m_2, t)을 C_{call}에 추가.
      // m_1의 결과 데이터로 m_2를 호출할 수 있는 경우,
  5-2) m_2가 Search이면 ($view_{out}(m_1)$, $view_{in}(m_2)$, ⊥)를 C_{move}에 추가. // 다른 리소스의 홈뷰로이동

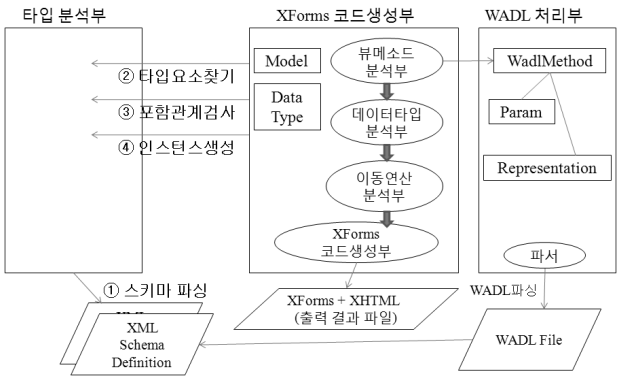
가 된다. 그 이외의 이동 연산은 어플리케이션의 특성이나 사용자의 작업 패턴 등에서 필요한 경우 추가될 수 있을 것이다.

(리스트 2)의 알고리즘 1은 이상의 이동 연산 계산 과정을 보여준다. 알고리즘 1은 주어진 메소드 집합 M에 대해 타입 포함관계와 REST 서비스 패턴을 적용하여 이동 연산 집합을 계산한다. 서비스에 대한 페이지 이동 계산 방법은 기존의 서버 기반 프레임워크에서도 사용되고 있었으나, 이 알고리즘은 클라이언트 페이지 내에서 호출과 이동, 컬백이라는 세가지 이동 연산 종류로 나누어 뷰 및 서비스 이동 관계를 계산한다. 또한 REST 스타일 서비스 패턴을 적용하므로서 데이터 타입의 조건을 만족하는 경우 중에서 필요한 것만 선택한다. 그 결과 가능한 한 적은 수의 컨트롤을 선택적으로 포함시켜 클라이언트에서 이동 연산의 복잡도를 낮추고 사용자 인터페이스의 효용성을 높일 수 있다.

5. 구 현

5.1 시스템 구성

정리 1의 가능한 이동 연산 집합과 알고리즘 1의 이동 연산 분석 방법을 구현하여 매시업 클라이언트 페이지의 코드 생성기를 XForms 언어에 대해 개발하였다. XForms 언어로 구성된 페이지는 별도의 스크립트 코드 없이 서비스 통신과 이동 연산 및 데이터 처리가 가능하여 REST 클라이언트의 개발을 위해 적합하다고 알려져 있다. 개발된 코드 생성기는 (그림 3)과 같은 구조를 가지며 WADL 처리부, 코드생성부, 타입분석부로 이루어진다. 먼저 선택된 WADL 파일에 대해 WADL 처리부는 입력 파일을 읽어들인다(가). 코드 생성부에서는 생성될 코드의 메소드와 뷰의 자료구조를 생성한다 (뷰메소드 분석부). 생성된 메소드와 뷰에 대해 입력 및 결과 데이터 타입을 이용한 이동 연산 분석을 통해 이동



(그림 3) 코드 생성기의 시스템 구성도

〈표 4〉 페이지 모델의 요소에 대응하는 XForms 언어의 요소 및 구조와 역할

페이지 모델의 요소	XForms 요소	부모 요소
페이지	html	-
메소드	xf:submission	html/header/xf:model
뷰	xf:group	html/body
이동연산	xf:submit, xf:trigger	xf:group

xmlns:xf="http://www.w3.org/2002/xforms"

연산을 구한다(코드생성부의 이동연산분석부). 생성된 메소드와 뷰, 이동 연산에 대해 대응하는 XForms 요소를 생성하는 방식으로 매시업 클라이언트 페이지가 생성된다. 타입 분석부는 코드생성부의 분석 기능을 지원하기 위하여 WADL에서 참조되는 스키마를 읽어들이고(①) WADL에서 사용되는 데이터 타입에 대해 XML 스키마 정의 파일의 요소를 찾고(②) 타입 요소 간의 포함관계 검사 기능을(③) 제공한다. 마지막으로 XForms 코드 생성부는 뷰 및 이동연산 분석 결과를 바탕으로 타입 요소에 대한 디폴트 인스턴스 생성 기능(④)을 이용하여 모델, 뷰, 이동연산 코드를 생성한다.

XForms 페이지를 생성하기 위하여 메소드 및 뷰의 이동연산에 대해 <표 4>와 같이 XForms 요소가 대응된다. 뷰와 submission 코드의 생성에 대해서는 이전의 연구에서 자세하게 다루었다[14]. XForms 에서는 데이터와 프리젠테이션 및 컨트롤을 분리하며 데이터는 model 요소의 instance에 모여 있다. 인스턴스의 구성을 위해 입력 매개변수 트리 하나와 각 메소드에 대한 결과 트리 한 개씩을 가지도록 model을 구성하였다.

제안된 방법에 의해 생성된 XForms 페이지의 사례가 (그림 4)와 같다. 생성된 파일은 하나의 페이지 안에 주어진 모든 메소드에 대한 뷰와 서비스 접근을 위한 코드, 그리고 이동 연산을 포함한다. 코드 생성기는 자바 표준 개발환경 J2SE 상에서 개발되었으며, 생성된 XForms 페이지 코드는 기개발된 J2ME 환경의 모바일 XForms 브라우저에서 실행된다[14]. <model> 요소에서 데이터를 가지고 submission은 요청 전송과 회신 처리부(replace 속성부)를 포함하고 있다. group은 뷰에 대응하며, submit와 trigger가 각각 호출과 뷰 이동 연산에 대응한다.

5.2 구현 결과

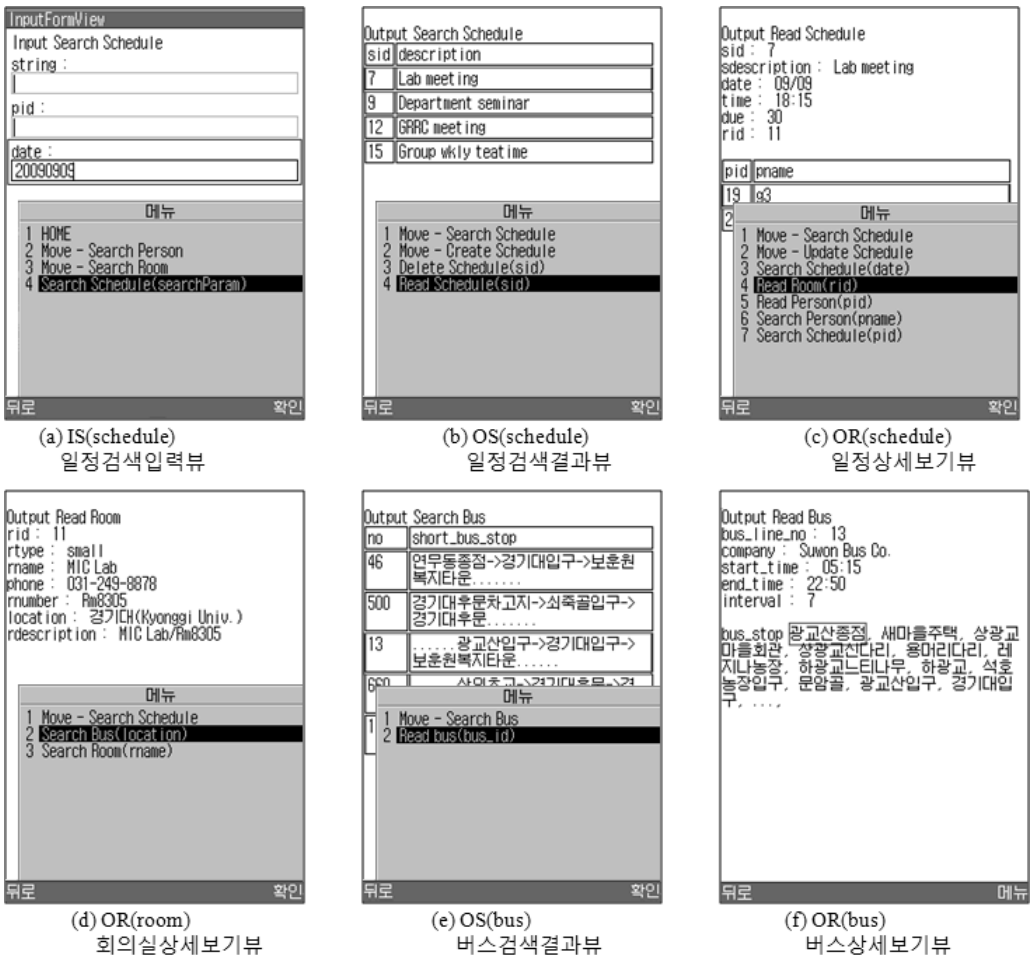
3장의 사례 시나리오에 대한 구현 결과를 (그림 5)에서 보여준다. (그림 5)의 각 화면은 (a) 일정 검색 뷰, (b) 검색 결과 리스트 뷰, (c) 선택된 일정 상세보기 뷰, (d) 해당 회의실 정보 상세보기 뷰, (e) 해당 장소로 가는 버스 검색 결과 뷰, (f) 선택된 버스에 대한 상세보기 뷰이다. 각 화면에서는 그 뷰에서 연결 가능한 메소드 호출과 뷰 이동 연산을 메뉴의 형태로 보여준다. 뷰이동은 Home 또는 Move 형태로 표시되고 메소드 호출은 해당 메소드 타입과 입력 매개변수 타입으로 표시한다. 예를 들어 화면 (b)에서는 검색 결과 리스트에서 하나의 일정을 선택하여 일정 삭제 또는 일정 상세보기 요청이 메뉴 3, 4번으로 제공된다. 이 때 검색 결과 리스트에서 현재 선택된 일정의 아이디를 메소드 요청의 입력 매개변수로 전송한다. 한편 화면 (c)에서는 일정 상세보기로부터 같은 날의 일정 검색, 회의실 정보 상세 보기, 참석자 정보 상세보기, 이름으로 사람 검색, 참석자 아이디로 일정 검색 등의 메뉴가 3-7번으로 제공된다. 이렇게 각 뷰에서 가능한 메소드의 호출과 뷰 이동은 알고리즘 1의 방법에 의해 계산된 것으로 개발된 결과 시스템은 개발자의 추가적인 수정 없이 바로 사용 가능한 페이지 코드를 생성해 줄 수 있다. 특히 화면 (c)에서 (d), 또는 (d)에서 (e)로의 이동은 회의 정보에서 교통정보 관련 메소드를 바로 호출한 경우로 편리한 매시업 인터페이스를 제공할 수 있다.

(그림 5)에서 보여지는 생성 결과 코드는 WADL과 스키마에서 사용되는 태그 명과 메소드 이름, title 요소의 값 등

```
<?xml version="1.0" encoding="euc-kr" ?>
- <html xmlns="http://www.w3.org/1999/xhtml" xmlns:xf="http://w
  - <head>
    - <model>
      - <instance id="inputParam">
        + <param>
        </instance>
      - <instance id="outputResult">
        + <result>
        </instance>
      + <submission id="m0" method="GET" ref="instance('inputParam
          action="http://203.249.21.227:3001//rooms/{roomId}.x
          [@id='m0']/room">
      + <submission id="m1" method="GET" ref="instance('inputParam
          action="http://203.249.21.227:3001//rooms.xml" replace
          [@id='m1']/rlist">
      </model>
    </head>
  - <body>
    + <group id="m0-output" nodeset="instance('outputResult')/resu
    + <group id="m1-input" nodeset="instance('inputParam')/param,
    </body>
  </html>
```

```
- <group id="m0-output" nodeset="instance('outputRes
    <h1>ROOM</h1>
  + <output ref="room/rid">
  + <output ref="room/rtype">
  + <output ref="room/rname">
  + <output ref="room/phone">
  + <output ref="room/location">
  + <output ref="room/rdescription">
  - <submit submission="m1">
      <label>Search Rooms</label>
      <setvalue ref="instance('inputParam')/param/m
    </submit>
  - <trigger>
      <label>Search Schedules</label>
      <setfocus control="m1-input" />
    </trigger>
  </group>
```

(그림 4) 생성된 XForms 코드 부분



(a) IS(schedule) 일정검색입력뷰 (b) OS(schedule) 일정검색결과뷰 (c) OR(schedule) 일정상세보기뷰

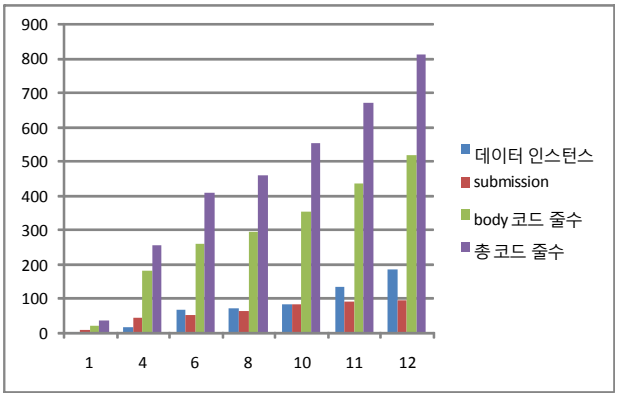
(d) OR(room) 회의실상세보기뷰 (e) OS(bus) 버스검색결과뷰 (f) OR(bus) 버스상세보기뷰

(그림 5) 사례 시나리오에 대해 생성된 코드의 브라우저 실행 화면

을 이용하여 자동 생성되었다. 예를 들어 뷰의 타이틀은 본 논문에서 제안된 메소드 분류 방법(O_{SCRUD})과 리소스의 이름을 이용하여 생성하였고 상세보기 뷰의 레이블은 태그 명을 사용하였다. 적절한 이름을 사용하여 생성된 코드가 효과적인 정보를 제공할 수 있다.

생성된 코드의 이동 연산은 사용자가 제공되는 메소드들을 모두 접근할 수 있고 편리하게 사용할 수 있도록 사용자 인터페이스를 제공한다. 특히 모바일 환경에서는 제공된 서비스 메소드 집합에 대해 바로 사용가능한 페이지 코드가 생성되어 실용적으로 사용할 수 있을 것으로 기대된다.

본 논문에서 생성된 코드는 서비스 별로 구조화되어 체계적이고 서비스의 개수 증가에 대해 컨트롤 개수가 선형적으로 증가하여 사용자 인터페이스의 복잡도가 그리 높아지지 않는 특징을 가진다. 이를 확인하기 위하여 <표 3>의 시나리오 메소드 10개와 2개의 공개된 REST 서비스[1]에 대해 생성된 페이지의 길이와 이동 컨트롤 개수를 분석한 결과가 (그림 6)과 같다. 이 그림에서 보여지는 바와 같이 서비스 메소드가 증가할 때 코드의 길이는 비례하여 증가함을 알



(그림 6) 서비스 메소드 개수에 따른 부분별 코드 줄 수

수 있다. 이것은 컨트롤 개수가 비례적으로 증가하기 때문에 가능한 것으로, 타입 조건을 만족하는 서비스 메소드의 이동 관계 중 일부만을 포함하는 제안된 방법의 효과를 보여준다.

5.3 기존 방법과 비교

현재 웹 2.0 클라이언트 페이지는 대부분 자바스크립트의

1) 구글 날씨 검색, http://www.google.co.kr/ig/api, 야후 지명 검색, http://kr.open.gugi.yahoo.com/service/poi.php

Ajax 기술을 이용하여 개발되고 있다[2, 3]. 자바스크립트는 언어의 성능이나 모듈화의 한계로 인하여 복잡해지는 클라이언트 기능을 개발하는데 비용과 시간, 유지보수 측면에서 많은 문제점을 가지고 있다. 이러한 문제점을 해결하기 위한 시도로는 Ajax 패턴이나 라이브러리 등을 적용하여 좋은 코드 구조를 이용하는 접근 방법이 있다. 현재 매시업 클라이언트의 개발은 Ajax 기법을 사용하여 핸들러 함수가 회신된 결과를 처리하여 브라우저 페이지를 갱신하는 방식으로 구현된다. 자바스크립트의 웹서비스 처리 기법에 대한 패턴이 여러 가지 제시되었으며[27] 특히 프로그래밍 패턴은 회신된 XML 데이터에 대한 처리 기법이나 서버와 브라우저 사이의 데이터 흐름 등을 다루고 있다. 본 논문에서 다룬 서비스와 뷰의 이동 연산은 기존의 Ajax 패턴에서는 다루어지지 않았다.

한편 서버 기반의 웹어플리케이션 개발 프레임워크에서도 데이터 타입을 이용하여 서비스 연결을 설계하고 그를 바탕으로 페이지 이동 코드를 생성한다[4, 5]. 본 논문에서 제안된 방법은 다음과 같은 점에서 서버 기반의 접근 방법과 차이를 가진다. 첫째, 웹 2.0 환경의 하나의 페이지 기반의 클라이언트에서 이동 제어 모델을 제공하고 있다. 둘째로 꼭 필요한 이동 컨트롤만을 생성하여 사용자 인터페이스의 효용성을 높인다. 셋째로 개발자의 개입이나 모델링 작업을 필요로 하지 않고 전체 코드의 자동 생성이 가능하다. 마지막으로 코드생성 시스템이 가볍고 독자적으로 동작하여 클라이언트 시스템에서도 수행 가능하다.

이상에서 살펴본 기존 방법과의 비교를 <표 5>에서 요약한다. 닷넷 웹서비스 플랫폼과 IBM의 웹스피어, 아파치 axis 웹서비스 기술은 서버 측 개발 지원 기능을 주로 제공하며 클라이언트의 뷰와 데이터 처리 코드를 부분적으로 생성하지만 이동 코드는 생성하지 않는다. 한편 Ajax 패턴은 클라이언트의 뷰, 이동코드, 데이터 처리 패턴을 제시하나 완성된 코드를 제공하지는 않는다. 본 논문에서 제안된 매시업 클라이언트 코드생성기는 클라이언트 코드의 모든 부분을 자동생성하여 바로 사용 가능한 페이지를 제공할 수 있다. 또한 HTML과 Ajax를 포함한 자바스크립트 코드를 생성하므로 실제 개발에 그대로 활용될 수 있다.

그러므로 본 논문의 방법은 현재 클라이언트 코드 개발을 위한 비용과 복잡도로 인한 유지 보수의 어려움, 그리고 동적인 서비스 환경에 따르는 클라이언트 코드의 동적인 생성 필요성을 해결하는 유용한 기술을 제시하고 있다.

6. 결 론

본 논문에서는 클라이언트 기반의 서비스 매시업 페이지에서 이동 연산의 설계 방법을 제시하였다. 서비스 메소드의 호출이나 새로운 뷰로의 이동 등을 포함하는 이동 연산은 메소드의 입력 및 결과 데이터 타입으로부터 가능한 연결 관계를 찾는 방법을 제안하였다. 한편 가능한 이동 연산 중에서 최적의 이동 집합을 구하기 위해 REST 스타일 서비스 패턴을 적용하였다. 그 결과 주어진 메소드 집합에 대해 페이지에 포함될 뷰와 이동 연산 집합을 계산하는 알고리즘을 제시하였다.

한편 제안된 방법을 구현하여 WADL 서비스 명세 언어를 입력으로 받아 XForms를 포함한 웹페이지 코드를 생성하는 시스템을 개발하였다. 생성된 코드의 이동 연산은 사용자가 제공되는 메소드들을 모두 접근할 수 있고 편리하게 사용할 수 있도록 사용자 인터페이스를 제공한다. 특히 모바일 환경에서는 제공된 서비스 메소드 집합에 대해 바로 사용가능한 페이지 코드가 생성된다.

본 논문에서 제안된 이동 연산 설계 방법과 코드 생성 기법은 여러 개의 서비스를 지원하는 매시업 클라이언트에서 서비스 호출과 회신 처리, 이동 연산 등을 REST 서비스 패턴을 도입하므로서 단순화하고 이를 통해 일관된 모델을 통한 코드 생성이 가능하였다. 현재 자바스크립트를 이용하는 매시업 클라이언트 개발은 언어의 한계와 서비스 유형의 다양성, 제어 구조의 복잡성 등으로 인해 개발과 유지 보수에서 높은 비용이 소요되고, 서비스의 생성과 소멸이나 상황에 따른 서비스의 변동 등 동적인 성격이 높아지면서, 본 논문에서 제안된 간단한 모델에 의한 자동 생성 방법이 이러한 문제의 해결에 도움이 될 것으로 기대된다.

참 고 문 헌

[1] ECMAScript Language Specification, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf.

[2] A.Mesbah, A.Duersen, “An architectural style for Ajax,” WICSA2007, 2007.

[3] A.Mesbah, “Analysis and testing of Ajax-based single-page web applications,” PhD thesis, TUDelft, 2009.

[4] Pinch, “WCF 4.0: Building WCF services with WF in

〈표 5〉 웹 클라이언트 페이지의 코드 생성 시스템 기능 비교

코드생성 기능별 비교	서버 코드	클라이언트		
		코드생성언어	이동 코드	데이터처리 코드
닷넷 웹서비스 플랫폼 (WCF/WF)[4]	O	HTML, C#,ASP	X	△
WebSphere 웹서비스 플랫폼[5]	O	HTML, JSP	X	△
아파치 axis 웹서비스 플랫폼 [12]	O	Java	X	△
Ajax 패턴 [27]	X	Javascript	△	△
매시업 클라이언트 코드생성기	X	HTML,Javascript	O	O

Microsoft.net 4.0," PDC 2008, 2008.
[5] IBM, "WebSphere Application Server," http://www-01.ibm.com/software/webservers/appserv/wasproductline/
[6] Vosloo, I., Kourie, E. "Server-centric web frameworks: an overview," *ACM computing surveys*, 40(2), 4:1-4:33, 2008.
[7] R.T. Fielding. "*Architectural Styles and the Design of Network-Based Software Architectures,*" Doctoral dissertation, University of California Irvine, 2000.
[8] L.Richardson, S.Ruby, *RESTful Web Services*, O'Reilly Media, Inc., 2007.
[9] Erik Wilde, "What is REST?" Tutorial at ICWE 2009.
[10] Web application description language(WADL), http://www.w3.org/Submission/wadl.
[11] World-Wide Web Consortium standards including XForms, XML Schema, XPath and Cascading Style Sheets. http://www.w3.org.
[12] Apache group, "Axis web services," http://ws.apache.org/axis/.
[13] Microsoft office online, (2007). InfoPath: 2007, http://office.microsoft.com/infopath.
[14] 이은정, "서비스 조합을 위한 XForms 기반의 모바일 사용자 인터페이스 개발", 정보처리학회논문지D, 15-Drnjs 6호, pp.879-888, 2008.
[15] Kisub Song, Kyong-Ho Lee, "An Automated Generation of XForms Interfaces for Web Servic," icws, pp.856-863, IEEE International Conference on Web Services (ICWS 2007), 2007
[16] Bozzon, A., Comai,,S., Fraternali,P., Toffetti Carughi, G., "Conceptual Modeling and Code Generation for Rich Internet Applications," In *ICWE 2006*, Menlo Park, California, USA, 2006.
[17] Winckler, M., Palanque, P. "StateWebCharts : A formal description technique dedicated to navigation modelling of web applications," *DSVIS'2003*, Funchal, 2003.
[18] M. Guell M., et al. "Modeling interactions and navigation in web applications," *Lecture notes in computer science*, 1921, 115-127, 1996.
[19] May H. et al. "View integration and cooperation in databases, data warehouses and web information systems," *Lecture notes in computer science 3730*, 213-49, 2005.
[20] Garcia, J. et al., "Model-driven approach to design user interfaces for workflow information systems," *J. of Universal Computer Science*, 14(19), 3160-3173, 2005.
[21] M.Laikorpi, P.Selonen, T.Systa, "Towards a model-driven process for designing restful web services," ICWS '2009, pp.173-190, 2009.
[22] S. Auer, et al., "Dbpedia: A nucleus for a web of open data," ISWS 2008, LNCS, Vol.4825, pp.722-735. Springer Berlin, 2008.
[23] J. Yu, et al, "Understanding mashup development," IEEE Internet computing, vol.12, issue 5, pp.44-52, 2008.
[24] R. Ennals, M.Garofalakis, "MashMaker: mashups for the masses," Proc. SIGMOD'07, pp.1116-1118, ACM Press, Beijing, China, 2007.
[25] S.Yu, J.Woodard: Innovation in the programmable web: characterizing the mashup ecosystem. ICSOC 2008, LNCS 5472, pp.136-147, 2009.
[26] J..Magazinius, et. al, "A lattice-based approach to mashup security," ASIACCS'10, April 13-16, 2010, Beijing, China., 2010.
[27] M.Mahemoff, *Ajax design patterns*, published by O'Reilly media Inc., 2006.
[28] L.Li, W.Chou, "Micro-Resource: A microformat framework for dual restful web services," WEBIST 2010.
[29] E.Benson, et.al, "Sync Kit: A persistent client-side database caching toolkit for data intensive websites," WWW 2010, North Carolina, USA, 2010.
[30] A.Jhingran, "Enterprise information mashups: integrating information, simply," VLDB 2006.

이 은 정
e-mail : ejlee@kyonggi.ac.kr
1988년 서울대학교 계산통계학과(학사)
1990년 한국과학기술원 전자계산학과(공학석사)
1994년 한국과학기술원 전자계산학과(공학박사)
1994년~2000년 전자통신연구원 선임연구원
2001년~현 재 경기대학교 전자계산학과 부교수
관심분야 : XML 처리 기술, 웹서비스