웹 응용의 플랫폼 독립적인 구성요소 식별□

정우성°, 이기열, 이병정*, 김희천**, 이종석***, 우치수
서울대학교 컴퓨터공학부, 서울시립대학교 컴퓨터과학부*,
한국방송통신대학교 컴퓨터과학과**, 우석대학교 컴퓨터공학과***
{wsjung°, kylee, wuchisu}@selab.snu.ac.kr, bjlee@venus.uos.ac.kr*,
hckim@knou.ac.kr**, jong1007@core.woosuk.ac.kr***

Identifying Platform Independent Elements of Web Applications

Woosung Jung°, Keeyoull Lee, Byungjeong Lee, Heechern Kim, Jongsuk Lee, Chisu Wu
School of Computer Science & Engineering, Seoul National University
School of Computer Science, University of Seoul*
Dept. of Computer Science, Korea National Open University**
Dept. of Computer Engineering, WooSuk University***

요 약

웹 응용은 서로 의존관계를 가지는 다양한 구성요소들이 복잡하게 뒤섞여 있기 때문에 구성요소를 효과적으로 분석하기 어렵고 개발을 병행하거나 유지보수를 하는 과정에서 많은 비용과 시간을 소요하게 된다. 본 논문에서는 웹 응용의 구성요소들을 아키텍쳐 관점에서 모델링하여 관계형 집합으로 표현하고 종속성을 파악하였다. 또한 관련 구성요소들을 독립적인 영역으로 나누는데 필요한 프로토콜 변수를 파악하고자 하였다. 본 논문에서 제안하는 웹 응용 아키텍쳐 정보는 관계형 집합으로 DBMS에 테이블로 저장되는 경우 SQL문을 통하여 항해나 패턴 분석, 기타 설계단계에서 웹 응용에 대한 다양한 질의를 가능하게 한다. 이러한 구성요소 정보는 기존의 웹 응용으로부터 역공학을 통해 얻을 수도 있으며, 재구조화를 통해 재공학의 도구로 활용될 수도 있다. 향후 추상화 아키텍쳐를 구체적인 웹 기술과 매핑시킬 경우 자동화를 통해 구현에 필요한 기초 코드를 생성하는데 응용될 수도 있다.

1. 서 론

웹 응용은 컴포넌트, 이벤트, 리소스, 사용자 인터페이스, 오퍼레이션, 항해 등 서로 의존관계를 가지는 다양한 구성요소들이 복잡하게 뒤섞여 있기 때문에 효율적인 분석이 어렵고, 유지보수에 많은 비용과 시간을 소요하게 된다.

본 논문에서는 웹 응용의 구성요소들을 아키텍쳐 관점에서 모델링하고, 각 구성요소들간의 종속성(dependency)을 분석하여 크게 4개 영역으로 구분하였다. 또한, 각 영역의 독립성을 보장하기 위해 경계 지점에서 고려해주어야 하는 프로토콜 변수에 대해서도 정리하였다. 웹 응용의 영역을 독립적으로 나누어 줌으로써, 구성요소 개발의 독립성을 향상시키고 결국 유연성이나 확장성이 좋은 아키텍쳐를 구성할 수 있는 기초를 제공할 수 있다.

여기서 기술되는 모델 정보는 웹 응용을 객체기반 관점에서 접근하고 있으나 관계형 정보를 바탕으로 분석된 것이기 때문에 테이블화시켜서 효과적으로 RDBMS에 저장 가능하다. SQL질의문 등을 통해 항해나 패턴 분석, 재사용 요소 파악 등 다양한 아키텍쳐 분석을 할 수도 있다. 또한, 이러한 저장 정보를 활용하여 웹 응용의 명세를 논리적으로 시뮬레이션 시켜 볼 수도 있다. 2장에서는 먼저 웹 응용 모델에 대한 구성요소들을 정의하고, 이들 간의 관계를 개략적으로 살펴본다. 3장에서 아키텍쳐 모델의 각 구성요소들을 보다 상세화하여 독립적인 4개 영역으로 구분하고, 나누어진 영역의 통합에 필요한 프로토콜 변수를 파악한다. 또한 각 영역의 추상적 모델이 구체적인 웹 기술과 어떻게 매핑이 가능한지를 예를 들어 보인다. 4장에서는 일부 주요 영역의 구성요소들에 대해서 관계형 데이터베이스 테이블로 구현해보았다. 5장에서는 결론과 함께 향후 연구 방향에 대해 언급하여 마무리한다.

2. 구성요소 정의 및 아키텍쳐 개요

웹 응용은 그림 1과 같은 다양한 구성요소들의 관계로 이루어져 있다.

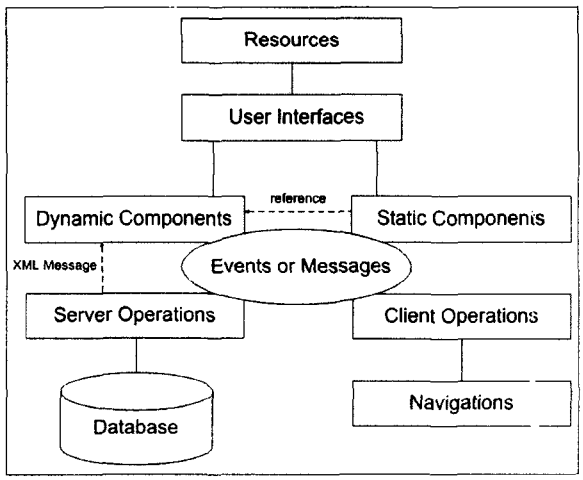

그림 1 웹 응용 아키텍쳐 개요

2.1. C: 컴포넌트(Components)

웹의 구성요소들 중에서 버튼(Button), 콤보박스(ComboBox) 등의 최소단위를 기본 컴포넌트(Primitive Component)라 정의한다. 여기서의 컴포넌트는 WCML(Web Composition Markup Language)[1]에서와 같은 개념으로 웹에서 태그(tag)로 표현되는 기본 컴포넌트를 포함하여 이들을 묶어서 그룹화(grouping)시킨 것도 컴포넌트로 생각한다. 그러므로 최상위 컴포넌트는 페이지(page)에 해당하며, 이러한 컴포넌트들은 대부분 포함(inclusion) 관계이거나 재사용(reuse)을 위한 상속(inheritance)관계가 되기 때

□ 본 연구는 한국과학재단 목적기초연구(R01-2002-000-00135-0)지원으로 수행되었음.

문에 하나의 페이지는 계층구조를 기반으로 이루어진다. 그렇기 때문에 구현 단계에서는 XML로 표현하기 쉽다[1]. 서버로 부터의 응답(response)에 따라 컴포넌트 정보가 바뀌는 동적 컴포넌트와 그렇지 않은 정적 컴포넌트로 나뉜다.

2.2. O: 오퍼레이션(Operations)

계산, 비즈니스 로직 등 실제 실행과 관련한 정보를 다루는 부분으로, 실행 위치에 따라 클라이언트 오퍼레이션(Client Operation)과 서버 오퍼레이션(Server Operation)으로 나뉜다.

클라이언트 오퍼레이션은 VBScript, JavaScript와 같은 클라이언트 스크립트(Client-side Script)가 대표적이며, 서버 오퍼레이션은 JSP(Java Server Page), ASP(Active Server Page), PHP(HTML Preprocessor)의 서버 스크립트(Server-side script)나 COM+, EJB 등의 컴포넌트 모듈이나 웹 서비스(Web Services)를 예로 들 수 있다. 서버 오퍼레이션은 DB Query를 수행하기 위한 경우가 대부분이며, 이때 ODBC, ADO/ADO.NET 등의 기술이 사용된다. 클라이언트 오퍼레이션은 이벤트 또는 다른 오퍼레이션에 의해 호출되며, 서버 오퍼레이션은 메시지를 통해 돌어오는 클라이언트의 요청(request)에 의해 응답한다.

2.3. N: 항해(Navigations)

항해는 오퍼레이션의 특수한 경우로 볼 수 있으며, 웹 응용의 구성이나, 웹 응용을 시뮬레이션함에 있어서 필수적인 정보이기 때문에 오퍼레이션과는 별개의 구성요소로 다룬다. 본 논문에서는 항해의 정의를 특정 컴포넌트가 다른 컴포넌트로 대치(replacement)되는 것으로 확장시켰다. 기존의 항해는 사용자 관점에서 페이지를 옮겨가는 것으로 보았기 때문에 복수개의 프레임이 존재하고 이들의 페이지 변경이 동시에 이루어지는 경우는 표현하기 어려웠다. 여기서는 항해도 컴포넌트 관점에서 다룰 수 있도록 일반화된다.

2.4. E: 이벤트(Events)

컴포넌트만으로는 정적인 구조밖에 표현할 수 없다. 웹 응용이 동작하는 것은 사용자와의 인터렉션 때문이다. 이벤트는 컴포넌트에 동적 정보를 바인딩시켜주는 역할을 한다. 즉, 마우스 클릭(MouseClick)이나 마우스 움직임(MouseMove) 등의 사건들을 컴포넌트에 정의해 놓고, 이러한 이벤트 발생 시 수행할 오퍼레이션들을 바인딩시킴으로 해서 원하는 동작을 얻게 되는 이벤트 기반 구조로 이루어져 있다[2].

2.5. M: 메시지(Messages)

이벤트와 유사하지만, 이벤트는 발생 위치가 클라이언트 지역인 반면 메시지는 서버 오퍼레이션을 수행하기 위해 클라이언트와 서버 사이에서 이루어진다. 주로 HTTP 요청/응답에 해당하며, 웹 서비스(Web Services)의 경우는 SOAP(Simple Object Access Protocol)에 해당한다[3].

2.6. U: 사용자 인터페이스(User Interfaces)

웹 응용에서는 시각적인 디자인 역시 중요한 비중을 차지하며, 비즈니스 로직이나 내부 아키텍쳐의 변경 없이 사용자 인터페이스만 변경되는 경우도 빈번하다. 최근에는 스킨(skin)이라는 개념으로 웹 응용의 디자인을 동적으로 변경시키는 것도 일반화되었다. 사용자 인터페이스가 독립적으로 제공될 경우 관련 유지보수가 쉬워질 뿐 아니라 다양한 클라이언트 환경에 맞도록 동적인 화면 구성이 가능해진다[4]. 효과적인 웹 응용 분석을 위해 사용자 인터페이스는 분리 작업될 수 있어야 하는데, 컴포넌트 정보가 XML로 구성될 경우 UI정보를 XSLT를 통해 표현함으로써 가능해진다.

2.7. R: 리소스(Resources)

사용자 인터페이스가 컴포넌트의 크기나 배치등과 관련한 논리적 인터페이스라면 리소스는 인터페이스에 적용되는 디자인 자체를 의미한다. 사용자 인터페이스에 종속되며, 아키텍쳐 분석과는 독립적이다.

2.8 D: 데이터베이스(Database)

웹 응용에서 서버 오퍼레이션이 이루어지는 대부분의 이유는 서버의 데이터베이스에 접근하기 위해서이다. 오퍼레이션에 필요한 정보를 처리할 수 있는 구조로 DB스키마가 설계되어야 하기 때문에 서버 오퍼레이션에 종속적이지만, 분석과는 독립적이다.

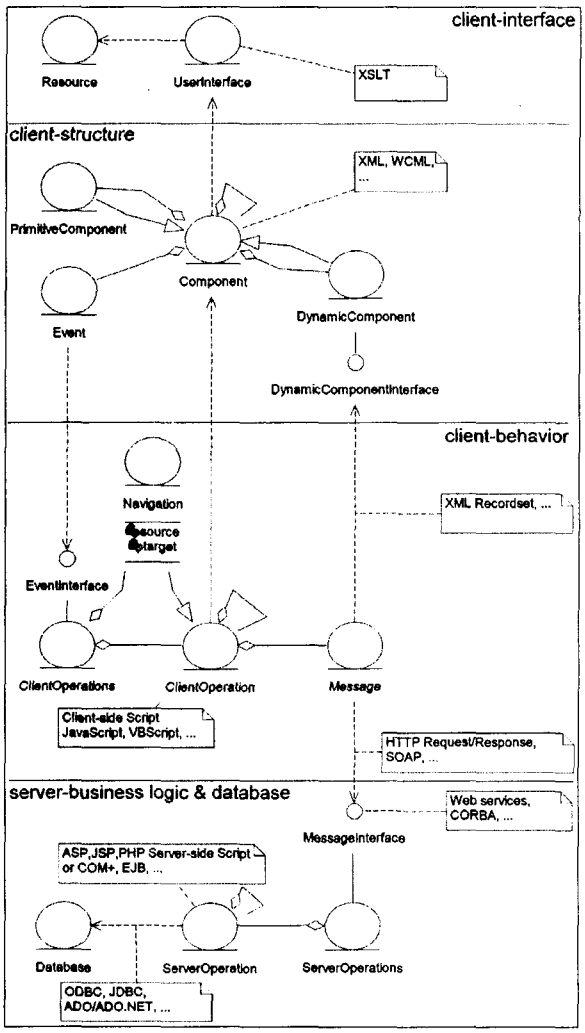

그림 2 웹 응용 아키텍쳐 세부 설계

3. 아키텍쳐 세부 설계 및 영역간 프로토콜 변수 분석

웹 응용은 플랫폼 독립적으로 앞서 언급한 8가지 구성요소들을 가지게 되며, 이러한 구성요소들은 다시 4개의 독립적인 영역으로 나뉘어 진다. 3-Tier구조에서는 데이터베이스가 서버 오퍼레이션과 분리하여 설계되며, 이 경우 서버 오퍼레이션은 응용 서버의 비즈니스 로직에 해당한다. 웹 응용 아키텍쳐의 세부 설계가 그림2에 표현되었으며, 각 구성요소에 해당하는 웹 기술의 예가 노트(note)로 기술되었다.

각 구성요소들에 대해 보다 구체적인 설계가 이루어질 수도 있으나 크게 클라이언트 영역에서는 인터페이스, 컴포넌트 구조 및 행위에 대한 정보가 기술되고, 서버 영역에서는 비즈니스 로직과 데이터베이스에 대한 설계가 이루어짐을 알 수 있다. 각 레벨은 개

발과정에서 해당 전문인력별로 업무가 분리되어질 수 있는 범위라고 볼 수 있다.

각 레벨의 구성요소들을 독립적으로 개발하고 통합하기 위해서는 레벨 사이의 사용관계(점선)를 분석하여 미리 정의되어진 프로토콜을 구해야한다. 사용관계를 파악해 볼 때, 인터페이스는 컴포넌트에 종속되고, 컴포넌트에 정의된 이벤트에 클라이언트 오퍼레이션이 종속된다. 또한 서버 오퍼레이션은 클라이언트로부터 요청되는 메시지에 종속적임을 알 수 있다. 하지만, 반대로 오퍼레이션에서 참조하는 컴포넌트들이 존재하기 때문에 아키텍쳐의 종속에 대한 출발점은 컴포넌트와 오퍼레이션 모두에 있음을 알 수 있다. 각 영역은 다른 영역과의 통합을 위해 최소한의 프로토콜이 필요한데, 이러한 변수들이 그림3에 나타나 있다. Rid, Cid, Eid, Mid, Cd_id는 각각 리소스, 컴포넌트, 이벤트, 메시지, 동적 컴포넌트의 고유 ID를 의미한다.

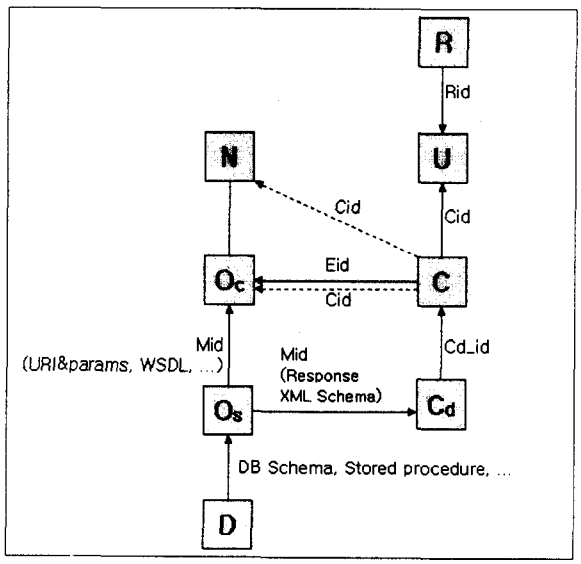

그림 3 구성요소간 프로토콜 변수

4. 관계형 데이터베이스를 이용한 아키텍쳐 표현

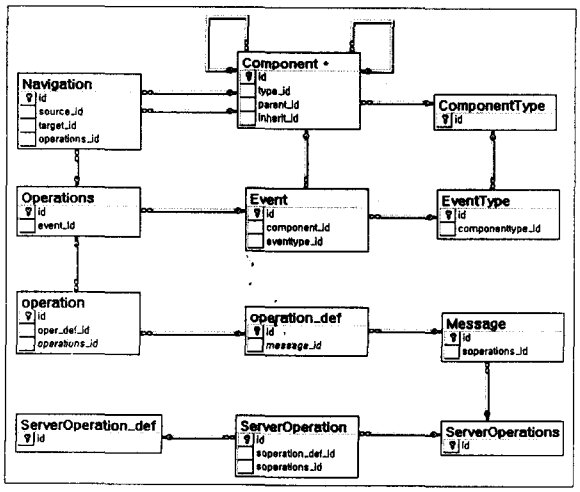

그림 4 웹 응용 아키텍쳐 관계형 데이터베이스 스키마

지금까지의 구성요소들중 리소스, 사용자 인터페이스, 데이터베이스를 제외한 주요 구성요소로 표현된 웹 응용은 그림4와 같은 구성요소로 이루어져 있다고 볼 수 있다.

그림 4는 관계형 데이터베이스 테이블로 주요 구성요소들을 상세히 스키마로 기술한 것이다. 컴포넌트의 이름이나 기타 속성을 제외하고 기본키(Primary Key)와 외래키(Foreign Key)만으로 테이블간의 관계를 구현해보았다. 웹 아키텍쳐를 표현하기 위한 DB스키마를 보면 각 구성요소들의 테이블에 저장되는 레코드값들이 관련한 프로토콜 변수에 종속되어 있기 때문에 이 값이 정해질 경우 독립적으로 관리될 수 있음을 확인할 수 있다. 또한, 아키텍쳐가 관계형 데이터베이스로 표현되어 있기 때문에 SQL문을 이용하여 아키텍쳐에 대한 다양한 질의가 가능하다. 그림 5는 이 스키마를 이용하여 가장 링크가 많이 되어 있는, 즉 항해의 목적(target) 컴포넌트로 사용되는 경우가 가장 많은 페이지 컴포넌트의 id와 링크 개수를 구하는 SQL 질의문의 예를 보여준다.

```
select top 1 Component.id, count(Component.id)
from Component, Navigation
where Component.id = Navigation.target_id
group by Component.id order by count(Component.id) desc
```

그림 5 SQL문을 이용한 웹 응용 아키텍쳐 질의 예

5. 결론 및 향후 계획

본 논문에서는 웹 응용의 구성요소들을 분석하여 추상적 단계의 아키텍쳐로 모델링하여 독립적인 4개 영역으로 구분해 보았다. 또한, 각 영역의 독립성을 고려하여 통합하기 위해 필요한 프로토콜 변수들에 대해서도 살펴보았다. 다양한 웹 기술에 공통적으로 적용될 수 있는 논리적 설계를 통해 웹 개발 및 아키텍쳐의 독립성을 높이고, 다양한 분석이 가능하도록 하였다.

아직까지는 이러한 객체 이벤트 기반의 웹 모델이 충분히 검증되지 않았으나 추후 웹 아키텍쳐 모델을 이용한 실제 구현을 통해 지금까지의 모델을 검증하고, 계속적으로 추가적인 구성요소들의 속성이나 관계들을 일반화시켜 나갈 계획이다. 최종적으로는 추상화 레벨의 웹 응용 설계가 실제 구현으로 이어질 수 있는 방법을 연구 및 구현하는 것이며, 이 과정에서 각 영역별 구성요소들의 유지보수성 향상을 위해 독립성이 유지될 수 있는 방법에 관한 연구 및 질의를 통한 패턴 분석 등의 연구가 병행되어야 한다.

6. 참고 문헌

[1] M. Gaedke, C. Segor, and H. W. Gellersen, "WCML: Paving the Way for Reuse in Object-Oriented Web Engineering," *Symposium on Applied Computing*, pp. 748-755, 2.

[2] A. Carzaniga, E. D. Nitto, and D. S. Rosenblum, "Issues in Supporting Event-based Architectural Styles," *Proceedings of the third international workshoo on Software architecture*, pp. 17-20, 1998.

[3] F. Curbera, M. Duftler, and R. Khalaf, "Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, Vol. 6, Iss. 2, pp. 86-93, 2002.

[4] G. Menkhaus, "Architecture for Client-independent Web-based Applications," *Technology of Object-Oriented Languages and Systems*, pp. 32-40, 2001.