웹 프리젠테이션 레이어 생성을 위한 XSLT 스타일쉬트 설계

(XSLT Stylesheet Design for Building Web Presentation Layer)

채 정 화 † 유 철 중 †† 장 옥 배 †††
(Jung-Hwa Chae) (Cheol-Jung Yoo) (Ok-Bae Chang)

요 약 웹 기반 정보 시스템의 데이타 구조 및 프리젠테이션 로직으로부터 비즈니스 프로세스 정보를 분리하는 것은 여러 가지 이점을 가져온다. 그러나 웹 애플리케이션에서 비즈니스 프로세스 로직과 프리젠테이션 로직을 분리하는 것은 쉽지 않다. 심지어는 프리젠테이션 로직으로부터 데이타가 분리되지 않는 경우도 있다. 그러므로 비즈니스 프로세스에 대한 추상적인 모델을 정의하고, 프로세스 로직, 데이타 구조 및 프리젠테이션 로직을 분리하는 전략을 이용하여 그것을 동적인 사용자 인터페이스에 매핑시키는 작업이 요구된다.

본 논문에서는 데이타 구조 및 프리젠테이션 로직으로부터 비즈니스 프로세스 정보를 분리하고자 XSLT(Extensible Stylesheet Language Transformations)를 확장하여 프로세스를 인식하는 스타일쉬트를 제안한다. 이를 위하여 비즈니스 프로세스의 추상적인 명세를 제공하고, 비즈니스 모델을 프로세스 관점에서 살펴보고자 페트리네트 표기법을 이용하여 비즈니스 모델 요소 및 상호작용 활동을 추출한다. 이것은 웹 애플리케이션의 프로세스 구조로부터 상호작용 웹 문서의 동적인 부분, 즉 비즈니스 프로세스에서 사용자와 상호작용하는 부분을 분리하기 위한 시도이다. 본 연구에서 제안하는 아키텍처의 핵심은 XSLT 컨트롤러로서 이것은 워크플로 엔진과 웹 브라우저간의 인터페이스 역할을 한다. XSLT 컨트롤러는 XSLT 템플릿을 활성화시키는 인터프리터로서 프로세스 상태를 프리젠테이션 레이어에 매핑시킨다. 이렇게 각 부분을 모듈화하는 것은 사용자 인터페이스를 변경하여도 프로세스나 데이타의 논리적인 표현에 영향을 미치지 않도록 하며, 레이아웃 변형 명세서와 독립적으로 프로세스 로직을 변경할 수 있도록 한다. 즉, 웹 애플리케이션을 독립적인 방법으로 개발할 수 있어 개발을 보다 용이하게 하고 유지보수가 용이해진다.

키워드 : 웹 애플리케이션, 프리젠테이션, 사용자 인터페이스, 전자상거래

Abstract In the Web-based information systems, separating the business process logic from the data and presentation logic brings about a wide range of advantages. However, this separation is not easily achieved; even the data logic may be not separated from the presentation layer. So, it requires to define an abstract model for business processes, and then to map the model into the user's dynamic interface using the logic separating strategy. This paper presents a stylesheet method to recognize the process by extending XSLT (Extensible Stylesheet Language Transformations), in order to achieve the logic separation. To do this, it provides an abstract specification of the business process, and a scheme that extracts business model factors and their interactions using a Petri-net notation to show the business model into the process point of view. This is an attempt to separate users' interaction from the business process, that is, dynamic components of interaction Web document from the process structure of Web applications. Our architecture consist mainly of an XSLT controller that is extended by a process control component. The XSLT controller is responsible for receiving the user requests and searching the relevant templet rule related to different user requests one by one. Separation of concerns facilities the development of service-oriented Web sites by making if modular. As a result, the development of service-oriented Web sites would be very easy, and can be changed without affecting the other modules, by virtue of the modularization concept. So, it is easy to develop and maintain the Web applications in independent manner.

Key words : Web applications, presentation, user interface, XSLT

† 비 회 원 : 전북대학교 전산통계학과
jhchae@chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터과학과 교수
cjyoo@chonbuk.ac.kr
††† 종신회원 : 전북대학교 전자정보공학부 교수
okjang@chonbuk.ac.kr
논문접수 : 2003년 9월 16일
심사완료 : 2003년 11월 19일

1. 서 론

초기의 웹이 정적인 정보만을 제공하는 단순한 애플리케이션을 위한 도구였다면 최근에는 대량의 데이타를 기반으로 한 복잡한 애플리케이션을 위한 플랫폼으로 변화되고 있다. 웹 애플리케이션의 컨텐츠에 대한 논리적인 구조나 데이타를 레이아웃(layout)에 독립적인 형태로 표현하기 위한 다양한 연구가 진행되어 왔다. XML이 성공한 요인 중에 하나가 바로 문서의 레이아웃과 구조를 분리했다는 점이다. 레이아웃의 생성은 XSLT[1]와 같은 특별한 문서처리 언어를 이용할 수 있다. 이렇게 함으로써 애플리케이션 고유의 데이타 스키마를 형성할 수 있도록 하는 이점을 가져온다. 이것은 곧 프리젠테이션 로직을 변경하여도 문서의 논리적인 구조에는 영향을 미치지 않는다는 의미이다. 즉, 동일한 논리적인 문서로부터 서로 다른 매체에 대한 다양한 레이아웃이 생성될 수 있다.

전자상거래 사이트와 같은 서비스 지향 웹 애플리케이션의 기능을 고려하는 경우 문서의 내용과 데이타의 표현만이 해결해야 할 과제의 전부는 아니다. 그와 같은 웹 애플리케이션은 사용자와 백엔드 시스템과의 상호작용을 지원하는 동적인 문서를 나타내야 한다. 최근에는 CGI 스크립트, ASP, 서블릿 및 JSP와 같은 동적인 웹 페이지를 구축할 수 있는 다양한 접근 방법이 제시되고 있다. 이러한 기술을 사용하여 구현한 애플리케이션에서는 비즈니스 프로세스가 애플리케이션 코드 안에 포함된다. 반면 워크플로 관리 시스템(Workflow management system) 분야에서는 복잡한 비즈니스 프로세스의 추상적인 표현법이 개발, 유지보수 및 구현에 매우 유용하다고 주장한다[2]. 또한 프로세스 구조의 독립성을 나타내기 위해 페트리네트(Petri Nets)나 상태 차트(state charts)와 같은 추상 프로세스 모델로 표현한다.

본 논문에서는 프로세스 구조의 추상 명세서로부터 웹 페이지에서의 동적인 부분, 즉 비즈니스 프로세스에서 사용자와 상호작용해야 하는 부분을 분리하는데 목적을 둔다. 이를 위하여 프로세스 및 데이타 구조가 동적인 웹 페이지로 변형될 수 있도록 XSLT를 확장한다.

본 논문에서는 다음과 같은 직교적인 방법으로 개념적 영역을 구분하여 접근한다.

- 컨텐츠(content) & 표현(presentation): 웹 애플리케이션은 컨텐츠 및 사용자와의 상호작용과 관련된 표현으로 구성된다. 컨텐츠는 웹 애플리케이션에 의해 사용되는 도메인에 의존적인 데이타이며 보통 데이타베이스 시스템에 의해 관리된다. 또한 표현은 각 웹 페이지의 레이아웃이나 사용자와의 상호작용과 관련된 것이다.
- 구조(structure) & 행위(behavior): 애플리케이션의 논리적인 관점을 제공하는 것이 구조라면 행위는 사용자와의 상호작용을 의미한다.
- 이러한 분리는 애플리케이션의 각 부분을 모듈화함으로써 서비스 기반 웹 사이트의 개발을 보다 용이하게 한다. 또한 한 부분의 변경으로 인하여 다른 부분까지 변경하지 않아도 된다는 이점을 가져온다.

본 논문의 나머지 내용은 다음과 같다. 2장에서는 관련 연구를 분석하고, 3장에서는 비즈니스 프로세스의 추상 명세시 고려해야 할 점에 대해 살펴본다. 4장에서는 시나리오를 이용하여 비즈니스 모델을 도출하여 보고, 5장에서는 프리젠테이션 레이어 생성을 위한 아키텍처를 설명한다. 6장에서는 예를 통하여 XSLT 템플릿의 여러 유형을 살펴보고, 7장에서는 연구결과에 대한 평가를 하였으며, 8장에서는 결론 및 향후 연구에 대해서 언급한다.

2. 관련연구

네트워크 기반 정보 시스템으로 단순히 정적인 컨텐츠만을 전달하기 시작한 이래 이제 웹은 매우 복잡한 애플리케이션을 위한 플랫폼이 되었다. 이 장에서는 웹의 동적인 행위를 가능하게 하는 서로 다른 솔루션에 대해서 살펴보도록 한다. 특히 이미 서론에서 제안한 것과 같이 분리의 관점에서 비교 평가할 것이다.

웹 애플리케이션은 HTTP를 통해 단 방향적인 커뮤니케이션을 하고, 일반적으로 가볍고 무상태적인 클라이언트를 갖는다. 웹 애플리케이션은 일반적으로 n-계층(layered) 구조로 이루어진다. 이 n-계층 구조는 사용자 인터페이스, 프리젠테이션 로직, 비즈니스 로직, 하부구조 서비스 및 데이타 계층 등으로 분리되며 이러한 애플리케이션 로직은 물리적이라기보다는 논리적 즉, 기능적으로 분리되어 있는 구조이다.

계층 아키텍처는 사용자의 관점에서 시스템을 바라보는 아키텍처의 특수한 형태이다. 이것은 전면에 사용자 그리고 후면에는 기본적인 데이타 등으로 전면에서 후면(front-to-back)의 분리를 이끌어낸다. 전형적인 3 계층의 아키텍처에서 사용자 인터페이스와 관련된 계층은 비즈니스 로직과 분리된 층에 있고 비즈니스 로직 계층은 다시 데이타 접근 로직과 분리된 층에 있게 된다. 각 층들은 일반적으로 논리적으로뿐만 아니라 물리적인 분리를 의미하기 때문에 다른 머신에 존재할 수도 있다. 따라서 웹 애플리케이션은 논리적으로 뿐만 아니라 물리적으로도 서로 나뉘어져 자신의 일을 수행하기 위해 다양한 층들 사이에 분산적 통신이 이루어지게 된다.

최근에는 웹 애플리케이션의 개발에 있어서 프리젠테이션 레이어와 컨텐츠의 분리를 위한 다양한 시도가 있

다. XML은 문서의 컨텐츠와 레이아웃을 분리하는 마크업 언어이기 때문에 이러한 문제를 해결하기에 적합한 구조를 갖고 있다.

프리젠테이션 로직으로부터 비즈니스 프로세스를 분리하려고 하는 경우의 여러 해결책들을 살펴보자. CGI 스크립트 및 서블릿과 같은 솔루션은 개발자가 프리젠테이션 로직과 비즈니스 프로세스를 구분하는데 아무런 도움을 주지 못한다. 서블릿의 경우 전체 페이지가 하나의 서블릿 안에 구성된다. 즉, HTML 태그와 프리젠테이션 코드가 자바 코드 안에 같이 포함되어 개발자가 기존 페이지에 어떤 변경을 하는 경우 서블릿을 수정하고 재컴파일해야 한다. 이와 같은 문제는 개발뿐만 아니라 유지보수나 차후 시스템을 진화시키는 경우에도 심각한 문제가 아닐 수 없다. ASP나 JSP와 같은 서버측 스크립팅 기법들이 해결책으로 등장하였다. JSP[3]는 XML로 표현되는 잘 정의된 인터페이스와 함께 자바빈즈(*JavaBeans*)와 같은 소프트웨어 컴포넌트 안에서 애플리케이션 로직을 캡슐화하는 기능을 수행한다. 그러므로 애플리케이션 로직이 프리젠테이션 로직으로부터 분리되고 결국 애플리케이션 로직과 프리젠테이션 로직은 서로에게 영향을 미치지 않으며 변경될 수 있다. 그러나 JSP는 고수준의 추상화 레벨에서 프로세스 정의를 지원하지 않기 때문에 JSP에 매핑되는 프로세스와 관련된 사용자 인터페이스를 정의하기 어렵다. 클라이언트 측 스크립팅은 매핑을 가능하게는 하지만 그렇게 하기 위해서는 간단한 비즈니스 프로세스에서 조차 복잡한 과정을 필요로 한다.

웹 애플리케이션 설계는 대부분 컴포넌트 기반의 모델링 기법에 기반을 두고 있다. PlayFwas 시스템[4]은 웹 애플리케이션의 프리젠테이션 레이어에 초점을 둔 프레임워크로서 JSP에 대한 템플릿 소스 코드와 자바 서블릿, 자바빈즈 컴포넌트의 추상 클래스로 구성된다. WMEE[5]는 J2EE 애플리케이션을 위한 UML 기반의 코드 생성기 엔진이며, ATG Dynamo[6]는 개인용 웹 포탈이나 전자상거래 사이트를 개발하기 위한 애플리케이션 서버이다. 본 논문의 접근방법과 유사한 연구로는 Web Service User Interface(WSUI)가 있다. WSUI[7]는 상호작용 및 프리젠테이션 로직과 네크워크 서비스를 연결시켜 웹 컴포넌트 모델을 제안하였다. 이 연구에서는 컴포넌트의 프리젠테이션 로직이 비즈니스 로직과 독립적으로 작동되는 XSLT 스타일쉬트를 정의하였다. 이것은 부분적으로 우리의 목적과 일치하는 것이라고 할 수 있으나 한계점을 갖고 있다. 즉, 컴포넌트의 레이아웃이 컴포넌트 안에 존재하는 상태가 아닌 반응하는 이벤트의 견지로 정의하였다는 것이다. [8]의 연구에서는 본 논문과 비슷한 원리를 이용했는데, 여기서는 동적인 HTML 페이지를 설계하여 이를 명시적으로 프로세스 로직과 바인드하였다는 차이점이 있다. 지금까지의 많은 연구에서 웹을 보는 관점은 사용자와의 상호작용하는 경우 오직 데이타를 내비게이션(navigation)하기 위한 목적으로만 인식되어져 왔다[9].

본 연구에서는 웹 애플리케이션을 생성하기 위한 자동 코드 생성기와 같은 다른 프로젝트나 연구와는 달리 웹 프리젠테이션 레이어 생성을 위하여 프로세스 기반의 XSLT 스타일쉬트 설계에 초점을 둔다.

3. 비즈니스 프로세스의 추상적 명세

워크플로 관리 기술은 활동(activity)의 이행과 비즈니스 프로세스 로직을 분리하는데 공헌한 바가 크다. GUI 툴이나 워크플로 정의 언어 등을 이용하여 비즈니스 프로세스에 대한 고수준의 추상적인 명세가 시도되었다. 이러한 단계의 핵심 결과는 활동의 제어 및 데이타 흐름의 의존관계 등을 정의한 것이다. 활동은 프로세스 정의의 일부분이 아니며 독립적으로 정의된다. 워크플로 엔진은 실행시간에 프로세스 명세에 정의된 의존관계에 따라 활동을 수행한다. 워크플로 관리 기술은 프로세스 정의와 활동의 이행이 독립적으로 유지보수되고 변경될 수 있음을 증명하였다.

3.1 워크플로 이행시 상호작용 웹 문서의 역할

상호작용 웹 문서, 즉 웹 페이지가 사용자에게 제공되면 그 때부터 사용자와의 상호작용에 의해서 워크플로 활동이 이행된다. 웹 문서는 사용자가 워크플로 관련 데이타를 제공하도록 하며 다음 프로세스 실행을 결정한다. 사용자와의 상호작용은 워크플로 엔진에게 워크플로 관련 데이타 전달을 초기화시키며 활동의 완료를 알려준다. 또한 웹 문서는 상호작용하는 사용자에게 데이타를 보여주는 역할도 한다. 만일 상호작용 웹 문서가 워크플로 활동을 이행하였다면 데이타는 이러한 활동의 목적과 관련된 결과를 사용자에게 보여준다. 이외에도 사용자는 데이타 내비게이션을 위하여 웹 문서와 상호작용할 수도 있다. XSLT와 같은 레이아웃 생성을 위한 메커니즘을 이용하여 데이타 구조를 프리젠테이션 및 내비게이션에 대한 레이아웃과 분리시킬 수 있다.

상호작용 웹 문서는 워크플로 활동을 이행하는데 있어서 크게 두 가지 역할을 수행한다. 컨텐츠와 프리젠테이션 레이어를 분리하는 견지에서의 웹 문서는 프리젠테이션을 하는 매체로서의 역할을 하며, 워크플로 구조와 활동 이행을 분리하는 견지에서의 웹 문서는 활동 이행의 역할을 한다고 볼 수 있다. 이러한 두 가지 역할은 상호작용 웹 문서를 생성하기 위해 사용되는 메커니즘에 반영되어야 한다. XSLT는 단순히 구조적인 데이타로부터 프리젠테이션 레이어의 포맷을 생성하는데 이

용된다. 본 논문에서는 프로세스 구조와 활동 이행을 분리하는 목적으로 XSLT를 확장하여 활동 이행을 위한 상호작용 기능을 생성할 수 있도록 한다.

전자상거래 애플리케이션에서 데이타가 사용자에게 보여지는 것은 워크플로 프로세스의 현재 상태에 의존한다. 그러므로 개발하는 스타일쉬트에 이러한 의존관계를 표현하기 위한 XSLT를 사용자 인터페이스 개발자에게 제공해야 한다. 또한 워크플로 활동 이행시 상호작용 웹 문서가 워크플로 엔진과도 상호작용할 수 있도록 XSLT를 확장한다.

3.2 비즈니스 프로세스의 핵심 요소

전자상거래와 같은 서비스 지향 웹 애플리케이션의 설계는 일반적으로 비즈니스 모델과 프로세스 모델을 기반으로 한다[10]. 비즈니스 모델은 비즈니스 파트너간의 물품(정보) 교환과 관련 있는 것이라면 프로세스 모델은 비즈니스 통신 측면에서 운영적이거나 절차적인 부분에 초점을 둔다. 비즈니스 모델이 전자상거래 시스템이 "무엇"을 하는지 정의한다면 프로세스 모델은 "어떻게"를 정의한다고 할 수 있다. 이것은 시스템을 설계하는 프로세스가 두 가지 주요 단계로 구성됨을 말한다.

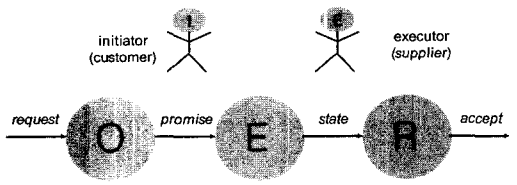

그림 1 트랜잭션 패턴

그림 1은 트랜잭션의 일반적인 패턴을 보여주는데, 비즈니스 트랜잭션은 일반적으로 두 행위자(actor)에 의해 수행된다[10]. 트랜잭션을 시작하여 완료를 수행하는 행위자와 목표에 실제적인 수행을 담당하는 행위자가 있다. 전자를 고객(customer) 또는 제출자(initiator)라고 하며, 후자를 공급자(supplier) 또는 실행자(executor)라고 한다. 트랜잭션은 주문 단계, 실행 단계 및 완료 단계의 세 단계로 구성된다. 주문 단계는 제출자의 요청과 실행자의 이행 약속이 이루어지는 제출자와 실행자 사이의 상호작용이다. 이 단계는 두 행위자간에 동일 목표에 대해 서로 합의하는 과정이다. 실행단계는 합의한 이행 약속으로부터 목표를 수행하는 과정이다. 이 단계는 실행자에 의해서 이행되는 과정이다. 완료단계는 다시 제출자와 실행자 사이의 상호작용으로 이루어지며 목표의 종료를 알리는 통지를 받고 결과에 대해 만족한다는 제출자의 동의가 이행된다. 이 단계는 수행한 결과에 대해 상호 동의에 도달하는 과정이다. 트랜잭션의 완료 시점이 목표 행위가 완료된 시점이 아니라 제출자가 결과에 대해 동의하는 시점이라는 것이 중요하다. 이것은 상호작용하는 임무에서 이벤트의 우위를 강조하는 것이며, 유형물뿐만 아니라 무형의 행위나 정보도 같은 방식으로 다룬다는 의미이다. 모든 트랜잭션은 초기화되며 그 종료는 처음 제출자가 완료하는 시점이다. 이런 방식으로 트랜잭션은 독립적인 하나의 체인을 구성하게 되는데 이것을 비즈니스 프로세스라고 한다.

비즈니스 프로세스의 두 가지 핵심 요소는 서로 의존관계를 갖는 활동과 활동 간에 전달되는 데이타이다. 그러므로 사용자 인터페이스 설계자는 발생되는 활동의 유형 및 활동과 관련된 데이타 측면에서 비즈니스 프로세스를 이해해야 한다. 부가적으로 활동을 수행하는 행위자와 활동 이행시 호출되는 매개변수(parameter)에 대해서도 알아야 한다. 본 논문에서는 이러한 관점에서 예를 통하여 살펴보기로 한다. 즉, 사용자 인터페이스 설계시 먼저 비즈니스 행위자, 활동 및 그들의 관계를 개략적으로 도출하고, 비즈니스 프로세스 관점에서 비즈니스 협력관계 및 각 프로세스를 보다 세밀히 표현할 필요가 있다.

3.3 XSLT의 확장

본 논문에서는 정적인 형태로 표현되는 데이타와 동적인 형태를 보여주는 프로세스를 구분한다. 자주 변경되지 않는 정적인 정보들은 XML 파일에 포함되며, 이것은 사용자 인터페이스 설계자가 재사용할 수 있는 형태의 정보이다. 이 절에서는 XSLT 구문을 이용하여 변동 가능한 로직들을 추출하고 표현하기 위하여 XSLT 스타일쉬트 메커니즘을 확장한다. 즉, 이러한 메커니즘을 통하여 이용자와의 상호작용에 의해서 자주 변동되는 로직들을 정의한다. 이렇게 정의된 XSLT 템플릿은 입력 데이타와 저장된 데이타 등에 따라 그 결과를 프로세스에게 다시 돌려준다. 다음은 어떠한 형태로 XSLT를 확장해야 하는지 기본 규칙을 살펴본다.

규칙 1. 워크플로 프로세스 및 데이타의 상태에 따른 문서 생성

프로세스 및 데이타의 상태에 따라 다음 프로세스에 반환할 문서를 결정하기 위한 구문을 나타내기 위하여 다음의 간단한 문장을 예로 들어 보자.

문장 "If age < 20, then greeting = "young man" else greeting = "old man""에서 "If age < 20"은 조건(condition)이 되고, greeting = "young man"은 행동(action)이 된다. greeting = "old man"도 action이 되는데, 전자의 action을 편의상 then action이라고 한다면, 후자의 action은 else action이 될 수 있다. 따라서 위 문장은 하나의 조건과 두 개의 행동으로 구성된다. 이와 같은 조건과 행동의 기본 모형은 그림 2와 같이

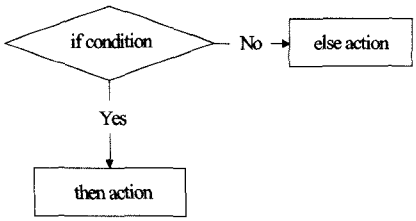

그림 2 조건과 행동의 기본 모형

나타낼 수 있다.

사용자 인터페이스 설계자는 비즈니스 프로세스에서 발생되는 활동의 유형 및 활동과 관련된 데이타를 이해해야 한다. 또한 활동을 수행하는 행위자와 활동 이행시 호출되는 매개변수에 대해서도 알아야 한다. 워크플로 프로세스 및 상태를 나타내는 매개변수를 x_1, ..., x_n이라고 하고, 각 상태를 평가하는 술어를 $P(x_1, ..., x_n)$이라고 나타내면 전체 술어 집합은 $C = \{P(x_1, x_2, ..., x_n)\}$로 나타낼 수 있다. 그러므로 워크플로 프로세스 및 데이타의 상태에 따라서 문서를 생성하기 위한 XSLT 확장 구문은 다음과 같이 나타낼 수 있다.

<xslt-e:for-each select="Exp(C)">

... </xslt-e:for-each>

이 구문은 워크플로의 실행 결과에 따라서 XSLT 문서를 해석하도록 하는데, 그림 3과 같이 C가 참(true)인 경우마다 <xslt-e:for-each> 구문에 의해서 템플릿이 매번 인스턴스화된다. XSLT 컨트롤러는 조건을 검토하여 적합한 XSLT 템플릿 인스턴스를 반환한다.

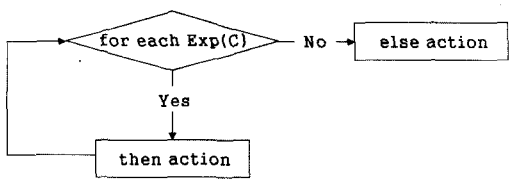

그림 3 워크플로 프로세스 상태 판별

반면 술어가 특정 매개변수에 대해서 만족하는지를 테스트하기 위해서는 다음과 같은 구문을 이용할 수 있다. 여기서 <xslt-e:if> 하위의 요소들은 술어가 참(true)으로 판정되는 경우에 포함될 수 있다.

<xslt-e:if test="Exp(C)">

... </xslt-e:if>

규칙 2. 워크플로 상태로부터의 데이타 검색

위의 <xslt-e:for-each> 구문을 처리하는 경우, 처리 결과는 매개변수 x_1, ..., x_n의 값에 따라 결정된다. <xslt-e:for-each> 구문 내부에서 워크플로 변수가 특정 값을 만족하는 경우 필요한 데이타를 검색하기 위한 구문은 다음과 같이 나타낼 수 있다.

<xsl:variable name="Exp(xk)" select=" ... "/>

즉, 프로세스의 현재 상태로부터 추가적인 정보를 얻고 싶은 경우나 특정 활동이 몇 번 수행되었는가를 알고 싶은 경우 등 위와 같은 구문을 작성할 수 있다. 또한 워크플로 상태로부터 직접 데이타를 검색하기 위해서 다음과 같은 구문을 이용할 수 있다.

<xslt-e:variable name=" ... "

select="Exp(query)"/>

또는 다음과 같이 명시적으로 구문 내에 삽입할 수도 있다. 여기서 query는 XPath와 Query를 이용하여 작성할 수 있다.

<xslt-e:value-of select="Exp(query)"/>

그림 4는 현재 상태로부터 원하는 데이타를 검색하여 XSLT controller에 반환하는 과정을 그림으로 나타낸 것이다.

구문<xslt-e:value-of select="/History/Book/Deliverd/Count">은 'Book'이 'Deliver'된 수를 나타내주는 것으로서, 이 구문을 실행하면 히스토리 데이타를 검사하여 프로세스에게 그 결과를 다시 반환한다.

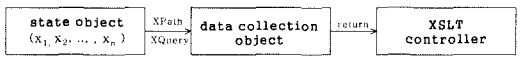

그림 4 상태 객체로부터 데이타 검색

규칙 3. 비즈니스 활동과 사용자 인터페이스 요소의 바인딩

HTML 폼의 동적인 요소들의 집합을 $E = \{e_1, e_2, ..., e_n\}$이라고 하고, 워크플로 활동 집합 $A = \{a_1, a_2, ..., a_n\}$라고 하자. $B : E \rightarrow A$는 비즈니스 활동을 사용자 인터페이스 요소와 연결하는 함수를 정의한다.

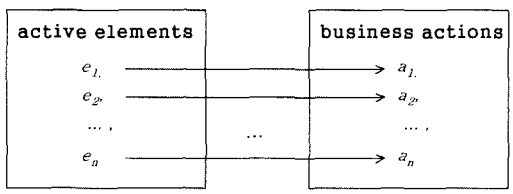

그림 5 비즈니스 활동과 HTML 요소와의 바인딩

이것을 XSLT 구문으로 나타내면 다음과 같다.

<xslt-e:submit action="a(x_1, ..., x_n)"

actionElement="id">

... </xslt-e:submit>

이 구문은 워크플로 활동과 사용자 인터페이스의 폼이나 링크와 같은 동적인 요소들을 연결한다. xslt-

```
<xslt-e:submit action="deliver(Payment($CreditCardForm.CreditCardInformation))"
      actionElement="pay" from="Customer" to="Portal">
   <form name="CreditCardForm"> Credit Card Information :
      <input type="text" name="Credit Card Information"/>
      <input type="submit" value="Submit Credit Card Information actionID=pay"/>
   </form>
</xslt-e:submit>
```

그림 6 <xslt-e:submit>의 사용 예

e:submit는 actionElement에 의해서 식별되는 HTML의 폼 요소와 액션 a를 묶어주는 구문이다. submit는 사용자 인터페이스 설계자가 직감적인 형태의 동적인 요소를 생성하도록 하며, 사용자가 입력한 값을 워크플로 엔진에게 전달하도록 추상화를 제공한다. 여기에서 action 속성은 수행될 행동을 명시한다. actionElement 속성은 고유의 식별자를 갖고, INPUT 속성의 actionID 값에 따라서 해당 action이 수행된다. 그림 6의 경우를 예로 들면 사용자 인터페이스의 Register 버튼을 클릭하는 이밴트의 발생은 워크플로 엔진에게 관련된 정보를 전달하여 actionElement의 값이 'pay'인 action이 실행된다. 모든 활동은 반드시 한 행위자로부터 다른 행위자로 방향성을 가지게 되는데, from과 to 속성은 각각 행위의 제출자(originator)와 응답자(target)를 나타낸다.

4. 비즈니스 모델의 프로세스 뷰

이 장에서는 간단한 서비스 기반 웹 애플리케이션 시나리오를 통하여 비즈니스 모델 요소 및 비즈니스 프로세스를 도출한다. 비즈니스 프로세스를 설명하기 위한 다양한 기법이 있으나 본 논문에서는 프로세스 명세 표기법인 페트리네트를 이용하여 포탈 사이트의 비즈니스 프로세스를 명세하여 본다.

다음은 신발을 판매하는 웹 포탈 사이트에 대한 가상의 시나리오이다.

웹 사이트는 각종 상품을 보여주고 그날의 히트 상품을 예측하는 등의 창구가 있고, 은행 및 물품 공급자 등과 관련된 백엔드 층으로 구성되어 있다. 사이트 방문자는 신발의 상세한 내용을 포함하는 카달로그 페이지를 이용할 수 있으며 구입 스크린에서 신발을 구입할 수 있다. 방문자는 자유롭게 카달로그 정보를 볼 수 있지만 신발을 구입하고자 한다면 회원에 가입해야 한다. 회원 가입은 무료이고 회원 가입을 위해서는 신용 카드 정보가 있어야 한다. 이 웹 사이트의 핵심 기능들을 정리하면 다음과 같다.

- 이용자는 물품을 구입하기 위하여 로그인해야 한다.
- 이용자는 카달로그에서 물품을 선택하여 구입할 수 있다.
- 등록하지 않은 고객이 물품을 구입하고자하면 회원에 가입하도록 요구한다.
- 비회원은 카달로그 정보를 이용할 수는 있지만 회원에 가입해야만 구입할 수 있다.

그림 7은 위의 시나리오에 대한 비즈니스 모델을 나타낸 것이며, 그림 8은 상태 전이 과정을 가시적으로 표현하기 위하여 프로세스를 페트리네트[11]로 명시한 것이다.

비즈니스 모델은 개발하고자 하는 시스템의 기본적인 비즈니스 측면을 서술하는 것으로서 어떤 행위자가 관련되어 있고, 행위자간에 제공되어지는 것은 무엇이며, 어떤 활동이 수행되는가 등을 식별하는 것이다. 반면 프로세스 모델은 프로세스의 절차적 측면과 프로세스에서 수행되는 활동의 제어 흐름을 서술하는 것이다. 본 논문에서는 단계별로 질문에 응답하는 과정을 통하여 비즈니스 모델 요소를 추출하고, 이 정보를 바탕으로 비즈니스 프로세스를 도출해 본다. 사용자 인터페이스 설계자는 그림 9와 같은 질문에 응답함으로써 비즈니스 프로세스 및 필요한 정보들을 추출할 수 있다. 본 논문에서는 고객과 웹 사이트간의 상호 작용을 다루며 물품 공급자(Supplier)와 관련된 부분은 제외시킨다.

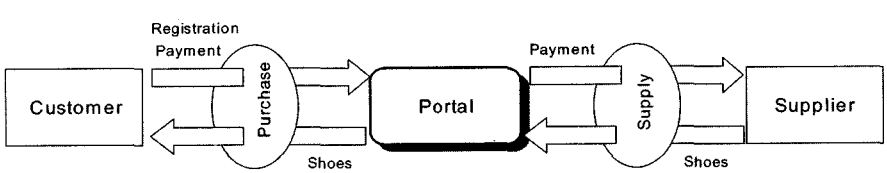

그림 7 신발 판매 웹 사이트의 비즈니스 모델

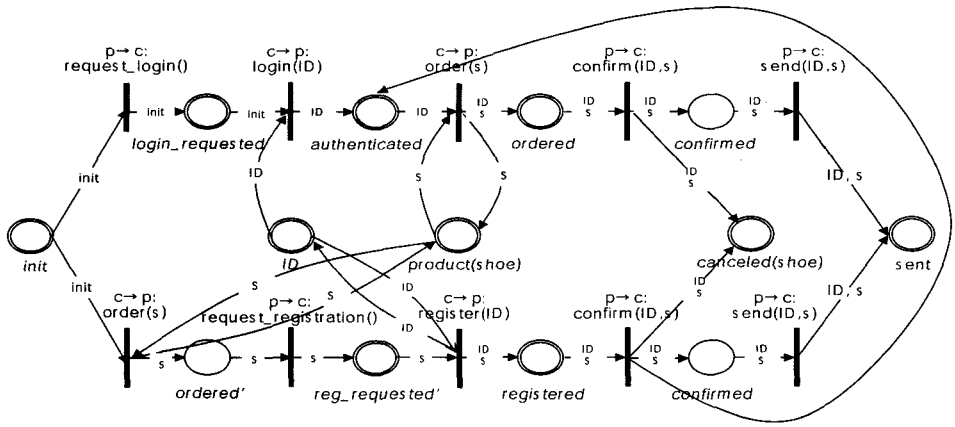

그림 8 신발판매 웹 사이트의 주문 페트리네트 모델

가. 행위자는 누구인가? Customer, Portal Site
나. 고객은 누구인가? Customer
다. 교환되는 유형 또는 무형의 물품에는 무엇이 있는가? Registration, Confirmation, Payment
라. 상호 작용되는 활동에는 무엇이 있으며, 그 활동과 관련된 행위자는 누구인가?

상호작용 활동	초기 행위자	대응되는 행위자
Request	Customer	Portal Site
Confirm	Portal Site	Customer
Payment	Customer	Portal Site
Register	Customer	Portal Site

마. 각 활동이 완료된 상태는 어떻게 나타나며, 각 활동과 관련하여 필요한 정보에는 무엇이 있는가?

상호작용 활동	초기 행위자	대응되는 행위자	필요한 정보
Registered	Customer	Portal Site	Name, ID, Email, Address
Confirmed	Portal Site	Customer	Name, ID, Email, Address
Requested	Customer	Portal Site	Shoe(Product Code, Size, Number)
Confirmed	Portal Site	Customer	Shoe(Product Code, Size, Number)
Payment	Customer	Portal Site	Credit Card Information, Name
Confirmed	Portal Site	Customer	Shoe(Product Code, Size, Number), Time

그림 9 비즈니스 모델 요소 추출을 위한 질문 및 응답

위와 같이 추출된 비즈니스 모델 요소와 상호작용 활동에 대한 유즈케이스 다이어그램은 그림 10과 같다.

5. 프리젠테이션 레이어의 생성을 위한 아키텍처

그림 11은 본 논문에서 제안하는 XSLT 템플릿이 어떻게 동적인 웹 페이지를 생성하는가를 보여주는 아키텍처이다. 아키텍처의 핵심은 XSLT 컨트롤러로서 이것은 워크플로 엔진과 웹 브라우저 간의 인터페이스 역할을 한다. 즉, XSLT 컨트롤러는 상태 의존적인 방법으로 해당 XSLT 템플릿을 활성화시키는 인터프리터로서 프로세스 상태를 프리젠테이션 레이어에 매핑시키기 위한 도구이다.

사용자 인터페이스 설계자는 Configuration XML 파일과 XSLT 템플릿을 작성한다. Configuration XML 파일은 정적인 정보와 프로세스 및 사용자 인터페이스 관련 정보 등을 포함하고, XSLT 템플릿은 상태에 따라 표현되는 여러 가지 정보들을 갖고 있다. XSLT 컨트롤러는 프로세스의 상태에 따라 해당 XSLT 템플릿을 XSLT 스타일쉬트로 활성화시키는데, 이 때 XML 파일이 적용되어 결과가 생성된다.

XSLT 컨트롤러는 사용자 인터페이스 설계자가 설계한 XSLT 템플릿을 일반적인 XSLT 파일로 변환시키는 역할을 수행한다. 이 과정이 처리되는 순서는 그림

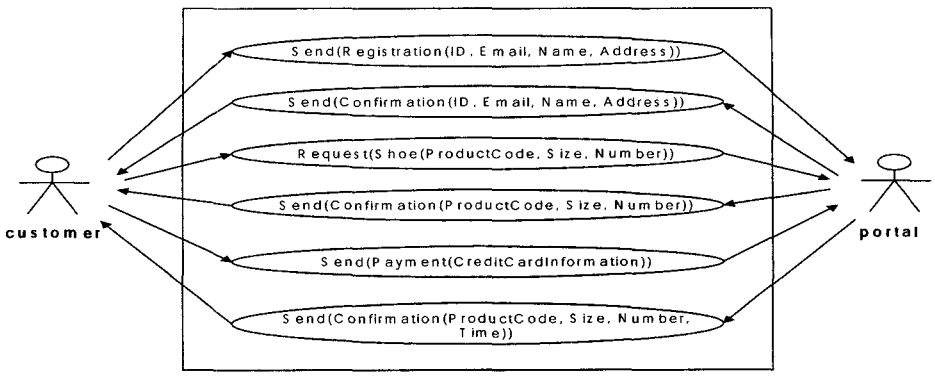

그림 10 비즈니스 모델 요소에 대한 유즈케이스 다이어그램

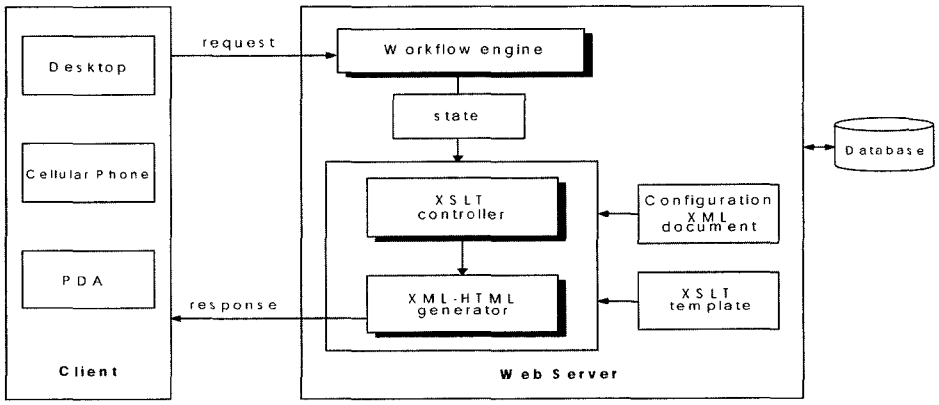

그림 11 XSLT 템플릿 기반 웹 프리젠테이션 생성 아키텍처

12와 같다. 먼저 XSLT 컨트롤러는 먼저 상태 조건을 판별하고, 이 조건에 따라 해당 XSLT 파일을 인스턴스화한다. 이 때 필요한 상태 변수를 스타일쉬트에 삽입한다. 다음으로 생성된 XSLT 스타일쉬트를 기반으로 XML 파일을 변환한다.

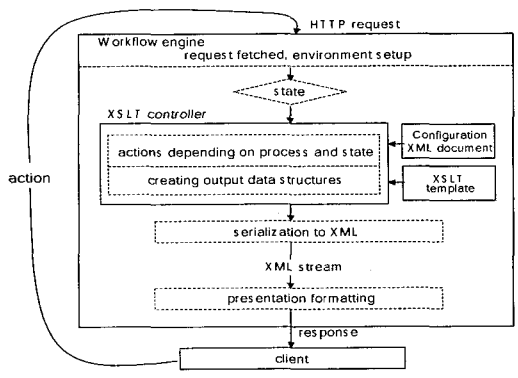

그림 12 웹 프리젠테이션 생성 단계

6. XSLT 템플릿

이 장에서는 본 논문에서 제안한 XSLT 템플릿이 적용되는 과정을 예를 통하여 살펴보도록 한다. 여기서는 앞장에서 소개한 신발 판매 웹 사이트 예제를 이용하도록 한다.

6.1 비즈니스 모델 요소 및 프로세스 추출

이 단계는 사용자 인터페이스 설계자가 필요한 정보를 추출하는 과정이다. 도출되는 과정은 4장에서 이미 설명하였으므로 여기에는 결과만 간단히 기술한다. 사용자 인터페이스 설계자는 다음과 같은 정보를 알아야 한다.

가. 프로세스와 관련된 행위자

- Customer
- Portal site

나. 교환되는 물품(교환되는 물품과 관련된 인자 명시)

- Registration(ID, Email, Name, Address)
- Confirmation(ProductCode, Size, Number)
- Payment(CreditCardInformation)

다. 이행되는 활동(관련된 행위자 명시)

- send(Customer→Portal, Registration(ID, Email, Name, Address))
- send(Portal→Customer, Confirmation(ID, Email, Name, Address))
- request(Customer→Portal, Shoe(ProductCode, Size, Number))
- send(Portal→Customer, Confirmation(ProductCode, Size, Number))
- send(Customer→Portal, Payment(Credit Card-Information))
- deliver(Portal→Customer, Shoe(ProductCode, Size, Number), Time)

6.2 Configuration XML 파일

XML 파일은 웹 문서에 의해 생성되는 모든 정적인 정보들을 포함한다. 이것은 사용자 인터페이스 설계자가 재사용할 수 있는 형태의 정보로서, 예를 들면 웹 사이트 방문자에게 보여지는 환영 메시지나 웹 사이트에 의해 제공되는 서비스에 대한 알림 메시지, 에러 메시지 등과 같이 자주 변경되지 않는 정적인 메시지 등이 해당된다. XML 파일에 들어갈 수 있는 또 다른 정보로는 웹 사이트가 제공하는 카달로그 정보가 있을 수 있으며, 웹 사이트의 논리적인 구조 및 웹 페이지 레이아웃의 구조도 XML 파일에 포함된다. XML 파일은 사용자 인터페이스 설계자의 의도에 따라 간단한 형태의 문서에서부터 매우 복잡한 문서 형태로 나타낼 수 있다. 그림 13은 6.1의 비즈니스 프로세스에 대한 XML 파일을 나타낸 것이다.

6.3 XSLT 템플릿

다음은 상태 정보를 기반으로 활성화되는 XSLT 템플릿의 몇 가지 예를 나타낸 것이다.

• 이용자에게 등록을 요구하는 경우

그림 14는 이용자가 등록 폼을 아직 발송하지 않은 상태에 대한 템플릿 예이다. 템플릿은 XML 파일의 <Registration> 요소를 적용하여 사용자 인터페이스에 폼을 생성하여 방문자가 Registration을 전송할 수 있도록 한다. 즉, 이 파일은 등록 폼의 전송 여부에 따라 조건적으로 활성화된다. 여기에서 action 속성은 수행될 행동을 명시한다. actionElement 속성은 고유의 식별자를 갖고, INPUT 속성의 actionID 값에 따라서 해당 action이 수행된다.

사용자 인터페이스의 Register 버튼에 클릭 이벤트의 발생은 워크플로 엔진에게 관련된 정보를 전달하도록 한다. 모든 활동은 반드시 한 행위자로부터 다른 행위자로 방향성을 가진다. from과 to 속성은 각각 행위의 제출자와 응답자를 나타낸다. 회원가입 폼 윗부분에 보이게 되는 헤더는 방문자가 무엇을 해야 하는지에 대한 정보를 알려준다. 헤더 정보는 상태에 따라 서로 다른 정보를 보여준다.

• 프로세스의 현재 상태로부터 추가적인 정보를 얻고자 하는 경우

그림 15는 입력된 값을 웹 문서에 전송하는 방법으로 워크플로 상태를 웹 문서에 매핑시키는 경우이다. 이 예에서는 워크플로 엔진에 의해서 제공되는 히스토리 로그 정보를 이용하여 현재까지 장바구니에 저장한 신발의 수($Shoe.Number)를 보여준다. 이 문서는 또한 이

```
<Configuration_XML>
    <Itmes>
        <Shoe> <Product_Code> ... </Product_Code> </Shoe>
        <Shoe> <Product_Code> ... </Product_Code> </Shoe>
    </Items>
<Components>
    <Mainpage>
        <Header/>
        <Notices/>
        <Registration/>
        <Deliverystatus/>
        <Footer/>
    </Mainpage>
    <Message>
        <Error> ERROR! </Error>
    </Message>
</Components>
</Configuration_XML>
```

그림 13 Configuration XML 파일

```
<xslt-e:if test="/enabled/send/Registration">
    <xsl:template match="HEADER">
        <b>Please register to access e-ShoeStore.</b>
    </xsl:template>
    <xsl:template match="Registration">
        <xslt-e:submit action="deliver(Registration($MemberShipForm.ID,
            $MemberShipForm.Email, $MemberShipForm.Name, $MemberShipForm.Address))"
            from="Customer" to="Portal" actionElement="register">
            <form name="MemberShipForm">
                <input type="text" name="ID"/>
                <input type="text" name="Email"/>
                <input type="text" name="Name"/>
                <input type="text" name="Address"/>
                <a href="none" actionID="register"> Register</a>
            </form>
        </xslt-e:submit>
    </xsl:template>
</xslt-e:if>
```

그림 14 XSLT 템플릿(1)

```
<xslt-e:if test="enabled/send/Shoe">
    <xsl:template match="HEADER"> You have ordered
        <xslt-e:value-of select=$Shoe.Number"/> shoes.
            Enjoy your shopping with following items.
    </xsl:template>
    <xsl:template match="Items">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Shoe">
        <xslt-e:variable match="Code" select="./Product_code/*>
        <xslt-e:submit action="request(Shoe($ProductCode, $Shoe.Number))"
            from="Customer" to="Portal" actionElement="shoeselect">
            <a href="none" actionID="shoeselect">
                <xsl:value-of select=$ProductCode"/>
            </a>
        </xslt-e:submit>
    </xsl-e:template>
</xslt-e:if>
```

그림 15 XSLT 템플릿(2)

용자가 카달로그에서 물품을 선택할 수 있도록 한다. XML 파일의 데이타 집합이 어떻게 워크플로 관련 데이타 값과 매칭되는지를 보여준다.

• 비회원이 물품을 구입하려고 하는 경우

그림 16은 등록하지 않은 비회원이 물품을 구입하려고 하는 경우의 템플릿이다. 방문자는 자유롭게 카달로그 정보를 볼 수 있지만 물품을 구입하고자 한다면 회원에 가입해야 한다. 즉, 이용자가 회원이라면 물품을 구입할 수 있고, 비회원이라면 신용카드 정보를 제시해야 한다. 사용자 인터페이스 설계자는 상태에 따라 어떤 상황이 발생할 것인지를 구분해야 한다.

7. 평 가

본 논문에서는 XSLT를 확장하여 프로세스를 인식할 수 있는 메커니즘을 제안하였다. 본 연구에서는 프로세스를 기반으로 직접 XSLT 페이지를 생성한다. XSLT는 선언적인 명세를 함으로써 명세를 모듈화시키고, 애플리케이션의 진화를 용이하게 한다는 장점이 있다. 본 논문에서의 아키텍처는 애플리케이션 서버와 같은 중량(重量)의 하부구조에 의존하지 않으며 XSLT 컨트롤러

```
<xslt-e:if test="not(enabled/registered)">
    <xsl:template match="Header">
        <b> You are not Member.
            Please give a valid Credit Card Number.
        </b>
    </xsl:template>
    <xsl:template match="Registration">
        <xslt:submit action="deliver(Payment($CreditCardForm.CreditCardInformation))",
                from="Customer" to="Portal" actionElement="payment">
            <form name="CreditCardInformation">
                <textarea name="CreditCardInformation"/>
                <input type="submit"    value="Submit Credit
                        Card Information"   actionId="payment"/>
            </form>
        </xslt-e:submit>
    </xsl:template>
</xslt-e:if>
```

그림 16 XSLT 템플릿(3)

라고 하는 핵심 요소로 구성된다. 이것은 클라이언트 기반의 C2C 비즈니스 프로세스나 P2P 환경의 비즈니스 프로세스를 갖는 애플리케이션에 적합한 경량의 서버이다. XSLT 컨트롤러와 워크플로 엔진은 적절한 결과를 생성하기 위해 유기적으로 구성되어 운영된다. XSLT 컨트롤러는 XSLT 스타일쉬트와 Configuration XML 파일을 처리하고 변형시킨다. 본 논문에서 제안한 템플릿 기반의 웹 애플리케이션 생성 메커니즘의 특성은 다음과 같이 여러 가지 측면에서 살펴 볼 수 있다.

- 유연성(Flexibility) 및 효율성(Efficiency): 동적인 HTML 페이지를 설계한 후 그것을 명시적으로 프로세스 로직에 바인드하는 기존의 방법과는 달리 본 연구에서는 규칙 기반의 명세로부터 페이지를 생성한다. 이렇게 하여 서로 다른 플랫폼에 대해서 다양한 레이아웃을 생성할 수 있기 때문에 유연성 및 효율성이 있다고 판단된다.
- 모듈성(Modularity): 명세적인 방법을 통한 컨텐츠와 행위에 대한 명확한 분리는 애플리케이션의 각 부분을 모듈화함으로써 서비스 기반 웹 사이트의 개발을 보다 용이하게 하고 한 부분의 변경으로 인하여 다른 부분까지 변경하지 않아도 된다. 본 논문에서는 직교적인 분리의 이론, 즉 '컨텐츠와 표현' 또는 '구조와 행위'의 분리를 기반으로 접근한 시도였는데, 이러한 방법이 반드시 좋은 설계 방법인가 하는 것은 고려해 볼 필요가 있다. 이러한 분리에 의해 얻어지는 이점을 정리하면 다음과 같다.
- 사용자 인터페이스의 변경이 프로세스와 데이타의 논리적인 표현에 영향을 미치지 않는다.
- 레이아웃 변형 명세서에 독립적으로 프로세스 로직의 변경이 가능하다.
- 프로세스 흐름 제어와 사용자 인터페이스 레이아웃의 생성을 클라이언트나 서버에 독립적으로 구성할 수 있다.
- 개발자가 각자의 업무에 집중할 수 있다. 즉, 프로세스나 데이타 구조를 개발하는 동안 레이아웃에는 관여하지 않아도 되며, 레이아웃을 개발하는 경우에도 구조 명세의 부분적인 면만을 이해하면 된다.
- 견고성(Robustness): 생성되는 모든 페이지에 대한 조건 및 값은 실행시간에 결정된다. 그러므로 프로그램의 실행 중에 변경되는 대상들에 대해서 개발자가 해결하는 것이 아니라 아키텍처에 의해서 지원된다.
- 복잡성(Complexity): 일반적인 CGI 스크립트 또는 서블릿 기반의 아키텍처 보다는 접근 방법이 복잡하다고 할 수 있다. 그러므로 결과적으로 애플리케이션이 작은 태스크를 처리하는데 있어서 더 복잡할 수도 있다. 그러나 보다 큰 태스크를 처리하는 경우 통합적인 접근방법이 없는 일반적인 다른 아키텍처가 오히려 더욱 많은 복잡성을 내포하고 있다.
- 성능(Performance): 본 논문의 아키텍처에서와 같이 간접적인 방법을 이용하지 않는 낮은 수준의 다른 일반 접근 방법이 오히려 더 우수한 성능을 나타낼 수도 있다. XSLT 컨트롤러는 먼저 상태 조건을 판별하고, 이 조건에 따라 해당 XSLT 파일을 인스턴스화하며, 생성된 XSLT 스타일쉬트를 기반으로 XML 파일을 변환한다. 이렇게 XML 파일 및 XSLT 파일을 해석하고 처리해야 하는 XSLT 컨트롤러의 역할로 인하여 처리 시간이 비교적 오래 걸린다는 단점을 갖고 있다. 이것은 곧 시스템 성능의 문제이므로 XSLT 처

리 부분에 대한 개선이 필요하다.

본 논문에서는 XSLT를 확장하여 프로세스를 인식할 수 있는 메커니즘을 제안하였다. 본 연구에서는 프로세스를 기반으로 직접 XSLT 페이지를 생성한다. XSLT는 선언적인 명세를 함으로써 명세를 모듈화시키고, 애플리케이션의 진화를 용이하게 한다는 장점이 있다.

8. 결 론

본 논문은 웹 애플리케이션의 프로세스 구조로부터 상호작용 웹 문서의 동적인 부분, 즉 비즈니스 프로세스에서 사용자와 상호작용하는 부분을 분리하기 위한 시도이다. 상호작용 웹 문서는 워크플로 활동을 이행하는데 있어서 데이타를 표현하는 매체로서의 역할과 활동 이행의 역할을 한다. 이러한 두 가지 역할은 상호작용 웹 문서를 생성하기 위해 사용되는 메커니즘에 반영되어야 한다. XSLT는 단순히 구조적인 데이타로부터 프리젠태이션 레이어의 포맷을 생성하는데 이용된다. 우리는 본 논문에서 프로세스 구조와 활동 이행을 분리하는 목적으로 XSLT를 확장하여 활동 이행을 위한 상호작용 기능을 생성할 수 있는 프로세스 상태 기반 XSLT 템플릿을 제안하였다. 이를 위하여 먼저 비즈니스 프로세스의 추상적인 명세를 하였으며, 비즈니스 모델을 프로세스 관점에서 살펴보고자 페트리네트 표기법을 이용하여 비즈니스 모델 요소 및 상호작용 활동을 추출하였다.

이렇게 각 부분을 모듈화하는 것은 서비스 기반 웹 애플리케이션의 개발을 보다 용이하게 하고, 각 부분을 독립적으로 개발할 수 있어서 한 부분을 변경하여도 다른 부분에 영향을 미치지 않는다는 이점을 가져온다. 즉 각각의 개발자가 프로세스를 서술하고, 데이타베이스를 유지보수하며, 사용자 인터페이스를 생성하는 등 고유의 자신의 업무에 집중할 수 있도록 한다.

향후 연구로는 본 논문에서 제안한 아키텍처를 구현하여 웹 애플리케이션의 프리젠테이션을 생성하는 것이다. 또한 구현된 아키텍처의 성능에 대한 테스트가 필요하다.

참 고 문 헌

[1] J. Clerk, WXSL Transformations(XSLT) Version 1.0, http://www.w3.org/TR/xslt, 1999.

[2] WfMC Members, "Workflow Management Reference Model," The Workflow Management Coalition, http://www.wfmc.org/standards/docs.htm.

[3] E. Pelegri-Llopart and L. Cable, "JavaServer Pages Specification," Version 1.1, http://java.sun.com/products/jsp, 1999.

[4] A. Saimi, T. Syomura, H. Suganuma, and I. Ishaida, "Presentation Layer Framework of Web Application Systems with Server-Side Java Technology," Proc. of 24th IEEE Annual International Computer Software and Applications Conference, 2000.

[5] Sygel Wonder Machine Enterprise Edition, http://www.sygel.com

[6] Ejen (Code Generation System), http://ejen.sourceforge.net

[7] E. Anuf, M. Chaston, and D. Moses, "Web Service User Interface WSUI 1.0," http://www.wsui.org/doc/20011031/WD-wsui-20011031.html, 2001.

[8] J. J. Rodriguez and O. Diaz, "Seamless Integration of Inquiry and Transaction Tasks in Web Applications," Proc. of 9th IFIP 2.6 Working Conference on Database Semantics, Hong Kong, 2001.

[9] K. Aberer and A.Wombacher, "A Language for Information Commerce Processes," Third International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, San Jose, California, USA, June 21-22, 2001.

[10] P. Kayaweera, P. Johannesson, and P. Wohed, "From Business to Process Pattern in e-commerce," Proc. of 6th International Workshop on the Language-Action Perspective on Communication Modelling, Montreal, Canada, July 21-22, 2001.

[11] J. L. G. Dietz and J. Barjis, "Petri Net expressions of DEMO Process Models as a rigid foundation for Requirements Engineering," Proc. of 2nd International Conference Information Systems, Stafford, July 4-7, 2000.

채 정 화

1992년 군산대학교 컴퓨터과학과 졸업(이학사). 1999년 전북대학교 교육대학원 전자계산교육전공 졸업(교육학석사). 2002년 전북대학교 대학원 전산통계학과 박사과정 수료. 관심분야는 소프트웨어 컴포넌트 기술, 객체지향 기법, 웹 애플리케이션 기술 등임

유 철 중

정보과학회논문지 : 소프트웨어 및 응용 제 31 권 제 1 호 참조

장 옥 배

정보과학회논문지 : 소프트웨어 및 응용 제 31 권 제 1 호 참조