웹 응용의 항해 모델 검사[1)]

김택수[0] 박상현 이병정* 김희천** 우치수
서울대학교 컴퓨터공학부, 서울시립대학교 컴퓨터과학부*, 한국방송통신대학교 컴퓨터과학과**
{dolicoli[0], zez4shy, wuchisu}@selab.snu.ac.kr, bjlee@venus.uos.ac.kr*, hckim@knou.ac.kr**

Checking Navigation Model of Web Application

Taeksu Kim[0] Sanghyun Park Byungjeong Lee* Heechern Kim** 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**

요 약

웹 응용 개발 과정에서 페이지와 항해를 모델링하기 위해 설계 단계에서 항해 모델을 사용한다. 하지만 페이지와 항해의 수가 증가함에 따라 모델의 구조는 복잡해지고 검증이 어려워진다. 본 연구에서는 항해 모델의 검증을 위해 항해 규칙을 정의하고 항해 규칙을 이용한 항해 모델 검증 방법을 제시한다. 또한 사례 연구를 통하여 항해 모델 검증의 예를 보인다.

1. 서 론

여타의 응용 프로그램과 달리 웹 응용(Web application)은 페이지를 통해서 사용자와 상호 작용을 한다. 사용자는 페이지를 호출함으로써 요청(request)을 발생시키며 페이지들을 통해 비즈니스 논리를 처리하며 정보를 획득한다. 즉 웹 응용은 페이지들과 그들 간의 항해 관계들로 구성된다고 할 수 있다. 이러한 웹 응용의 특징을 감안할 때 웹 응용 개발 과정에서 항해 모델을 설계하는 것은 중요한 과정의 하나라 할 수 있다.

이전 연구에서 항해가 웹 응용의 중요한 행위적 특성이라는 점에 착안하여 UML[5] 상태 기계 다이어그램(state machine diagram)과 시퀀스 다이어그램(sequence diagram)을 확장하여 항해 모델을 제안하였다[4]. 하지만 이 모델은 페이지와 항해의 수가 늘어날수록 복잡도가 증가하고 모델 내의 항해 오류를 검출하기 어렵다. 이런 이유로 작은 단위의 항해 모델은 유용하게 활용될 수 있으나 복수의 모델을 통합한 전체 항해 모델을 검증하기에는 부적합한 측면이 있다. 모델링 방법을 제시한 기존의 다른 연구들에서도[1, 2, 3] 항해 모델 검증에 대한 논의는 중요하게 다루어지지 않는 경향이 있었다. 본 연구에서는 이러한 기존 항해 모델의 단점을 보완하기 위해 항해 규칙을 이용한 모델 검증 방법을 제시한다.

논문의 구성은 다음과 같다. 2장에서는 항해 모델에 대해 간단히 소개하며, 3장에서 항해 규칙을 이용한 모델 검증 방법을 제시하고 항해 설계 과정을 제안한다. 4장에서는 사례 연구를 통해 항해 설계 기법의 실제 활용에 대해 살펴보며 마지막 5장에서는 연구의 결론을 도출하고 향후 연구 과제를 제시한다.

2. 항해 모델

2.1 항해 모델의 표현

웹 응용의 항해를 표현하기 위해 두 개의 모델을 이용한다. 페이지들과 그들 간의 항해 관계를 나타낸 '뷰 관점 항해 모델'과 하나의 항해를 위해 사용되는 컴포넌트 간의 데이터 흐름을 표현한 '데이터 전송 관점 항해 모델'이 그것이다.[4]

'뷰 관점 항해 모델'에서는 사용자들에게 제공되는 화면을 기준으로 페이지들을 정의하고 그들의 물리적 포함 관계와 연결 관계를 UML 상태 기계 다이어그램을 확장해 표현한다. '데이터 전송 관점 항해 모델'은 비즈니스 논리를 처리하는 페이지와 컴포넌트간의 데이터 흐름을 나타내는데 UML 시퀀스 다이어그램을 확장하여 데이터의 흐름을 기술한다.

2.2. 항해 모델의 요소 정의

웹 응용은 페이지들을 정점(vertex)으로 하고 그들 간의 관계인 항해를 간선(edge)으로 가지는 그래프 구조로 기술한다.

$$W = G(P, N)$$
$$P :$$
$$N :$$

항해의 근원페이지, 목적페이지와 경로, 경로 항해 집합을 다음과 같이 정의한다.

정의 1. 근원페이지, 목적페이지 페이지 p에서 페이지 q로의 항해 n에 대해, p를 n의 근원페이지라 하고 *source(n)*으로 표시한다. q를 n의 목적페이지라 하고 *target(n)*으로 표시한다.

정의 2. 경로 유한 연쇄 $np(p,q) = \{n_1, n_2, n_3, \cdots, n_m\}$가 다음의 조건을 만족할 때, $np(p,q)$는 페이지 p에서 페이지 q로의 경로라고 정의한다.

$$source(n_1) = p,$$
$$target(n_m) = q,$$
$$target(n_i) = source(n_{i+1}) \ where \ 1 \le i \le m-1$$

정의 3. 경로 항해 집합 페이지 p에서 페이지 q로의 경로 $np(p,q) = \{n_1, n_2, n_3, \cdots, n_m\}$의 모든 항으로 이루어진 집합을 경로 항해 집합이라고 하고 *pns(p,q)*로 나타낸다.

정의 4. 경로 집합 페이지 p에서 페이지 q로의 모든 경로 항해 집합들의 집합을 p에서 q로의 경로 집합이라고 정의하고 *PS(p,q)*로 나타낸다.

각 페이지와 항해는 식별자(identifier)를 가지고 있다고 가정하고 페이지 식별 함수와 항해 식별 함수를 다음과 같이 정의한다.

1) 본 연구는 한국과학재단 특정기초연구 (R01-2002-000-00135-0) 지원으로 수행되었음.

정의 5. 페이지 식별 함수 다음 조건을 만족하는 함수 *page*를 페이지 식별 함수라고 한다.

$$page : string \rightarrow P$$
$$page(id) = id$$

정의 6. 항해 식별 함수 다음 조건을 만족하는 함수 *nav*를 항해 식별 함수라고 한다.

$$nav : string \rightarrow N$$
$$nav(id) = id$$

3. 항해 모델의 검증

3.1. 항해 규칙

페이지와 항해를 설계할 때 제약 조건을 반드시 지켜야 하는 항해가 존재하는 경우가 있다. 이 제약 조건을 항해 규칙이라고 정의한다. 항해 규칙은 크게 다음의 네 가지 유형으로 분류할 수 있다.

● 페이지 직접 연결 조건

페이지 직접 연결 조건이란 서로 다른 두 페이지 p와 q에 대해 p에서 q로의 경로는 p에서 직접 q로 연결되는 항해 이외에는 없어야 한다는 조건을 의미한다. 식별자가 각각 x와 y인 두 페이지 간의 항해에 페이지 직접 연결 조건이 있다면 *DL(x, y)*라고 표현하고 그 정의는 아래와 같다.

$$\forall n \in N,$$
$$(source(n) = page(x) \Leftrightarrow target(n) = page(y))$$

● 마지막 페이지 조건

일부의 페이지의 경우 더 이상의 항해를 허용하지 않는 경우가 있다. 이 때 사용되는 조건이 마지막 페이지 조건이다. 식별자가 x인 페이지에 마지막 페이지 조건이 적용된다면 *LP(x)*라고 표현하고 그 의미는 다음 정의와 같다.

$$\forall n \in N, (source(n) \neq page(x))$$

● 필수 항해 포함 조건

페이지 p에서 q로의 모든 경로는 반드시 하나 이상의 항해 n을 거쳐야만 한다는 조건을 말한다. 식별자가 각각 x와 y인 두 페이지 간에 필수 항해 포함 조건이 있고 반드시 식별자가 z인 항해를 거쳐야 한다면 *IN(x, y, z)*라고 표현하고 다음과 같이 정의한다.

$$\forall pns \in PS(page(x), page(y)), \ (nav(z) \in pns))$$

● 항해 불포함 조건

페이지 p에서 q로의 모든 경로는 반드시 항해 n을 거치지 않아야 하는 조건을 말한다. 식별자가 각각 x와 y인 두 페이지 간에 항해 불포함 조건이 있고 반드시 식별자가 z인 항해를 거치지 않아야 한다면 *XN(x, y, z)*으로 표현하고 다음과 같이 정의한다.

$$\forall m \in N, \ \forall pns \in PS(page(x), page(y)),$$
$$(m \in pns \Rightarrow m \neq nav(z))$$

3.2. 항해 모델 검증

항해 규칙이 설정되면 이를 이용해 항해 모델의 오류를 검증할 수 있다. 항해 모델링의 결과 추출된 페이지들의 집합을 ***P***, 항해들의 집합을 ***N***이라고 하고 모든 항해 규칙들의 집합을 ***R***이라고 하자. ***R***의 모든 규칙에 대해 각 규칙의 유형에 따라 다음과 같이 검증을 시행할 수 있다.

● 규칙의 유형이 페이지 직접 연결 조건인 경우

항해 규칙이 *DL(x, y)*로 주어진 경우 항해 모델의 모든 항해들에 대해 식별자가 y인 페이지로 향하는 항해의 근원페이지를 검사한다. 만약 근원페이지의 식별자가 x가 아니라면 해당 모델은 페이지 직접 연결 조건을 위반한 것이라고 할 수 있다. 그림 1은 페이지 직접 연결 조건에 위배되는 항해 모델의 예이다. 항해 규칙이 *DL('main.html', 'event.html')*로 정의되어 있다면 'event.html'로 가는 항해는 반드시 'main.html'에서 출발해야 하지만 모델에서는 'notice.html'에서 연결되는 항해가 발견된다.

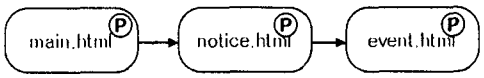

그림 1. 페이지 직접 연결 조건에 위배되는 항해 모델의 예

● 규칙의 유형이 마지막 페이지 조건인 경우

항해 규칙이 *LP(x)*로 주어진 경우 항해 모델의 모든 항해들에 대해 근원페이지의 식별자가 x인 항해의 존재 여부를 검사한다. 만약 그러한 항해가 존재한다면 해당 모델은 마지막 페이지 조건에 위배된 모델이라고 판단할 수 있다. 그림 2는 마지막 페이지 조건에 위배되는 항해 모델의 예이다. 항해 규칙이 *LP('event.html')*로 정의되어 있다면 어떤 항해도 'event.html'에서 출발할 수 없지만 항해 모델에서는 'joinForm.html'로 연결될 수 있는 항해가 존재한다.

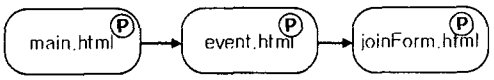

그림 2. 마지막 페이지 조건에 위배되는 항해 모델의 예

● 규칙의 유형이 필수 항해 포함 조건인 경우

항해 규칙이 *IN(x, y, z)*로 주어진 경우, 항해 모델의 모든 페이지 중 식별자가 x인 페이지를 먼저 검사한다. 그래프 탐색 알고리즘을 이용해 식별자가 y인 페이지로의 경로를 추출하고 모든 경로가 식별자가 z인 항해를 포함하는지를 조사하면 항해 모델을 검증할 수 있다. 그림 3은 필수 항해 포함 조건을 위반한 항해 모델의 예이다. 항해 규칙이 *IN('main.html', 'userInfo.jsp', 'login')*으로 주어진 경우 'main.html'에서 'userInfo.jsp'로의 모든 경로는 반드시 식별자가 'login'인 항해를 거쳐야만 하지만 모델에서는 식별자가 'login'인 항해를 거치지 않는 경로가 존재하게 되어 오류가 있음을 알 수 있다.

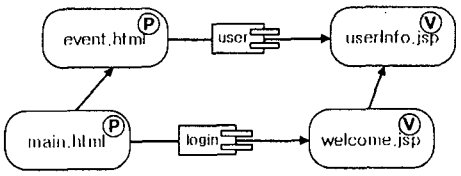

그림 3. 필수 항해 포함 조건을 위반한 항해 모델의 예

● 규칙의 유형이 항해 불포함 조건인 경우

항해 규칙이 *XN(x, y, z)*로 주어진 경우 필수 항해 포함 조건의 경우와 비슷한 과정으로 검증을 실시할 수 있다. 항해 모델의 모든 페이지 중 식별자가 x인 페이지를 먼저 검사한 후 그래프 탐색 알고리즘을 이용해 식별자가 y인 페이지로의 경로를 추출하고 모든 경로 중 식별자가 z인 항해를 포함하는 경로가 존재하는 지 여부를 조사한다. 그림 4는 항해 불포함 조건을

위반한 항해 모델의 예이다. 항해 규칙이 *XN('main.html', 'userInfo.jsp', 'logout')*으로 주어진 경우 'main.html'에서 'userInfo.jsp'로의 모든 경로는 반드시 'logout' 항해를 거쳐서는 안 되지만 모델에서는 'logout' 항해를 거치는 경로가 존재하는 오류를 보인다.

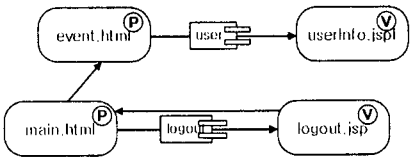

그림 4. 항해 불포함 조건을 위반한 항해 모델의 예

3.3. 항해 설계의 절차

웹 응용 개발 과정에서 항해 설계 과정은 다음과 같은 순서로 진행된다.

- 유즈케이스를 기반으로 항해 규칙을 설정한다.
- 설정된 규칙들을 기존에 작성된 규칙들과 함께 저장한다.
- 단일 유즈케이스에 대해 항해 모델을 작성한다.
- 작성된 항해 모델을 항해 규칙들을 기준으로 검증한다.
- 작성된 항해 모델을 기존에 작성된 항해 모델과 통합하여 항해 규칙들을 적용해 검증한다.
- 모든 검증 작업이 정상적으로 완료되면 구현을 한다.

항해 설계의 첫 단계는 유즈케이스를 분석하여 필요한 필수 페이지를 추출하고 그들 간의 항해 규칙을 네 가지 조건 유형에 맞추어 설정하는 것이다. 항해 규칙은 사용자의 요구 사항을 충족하기 위한 기본적인 제약 조건이므로 유즈케이스로부터 추출이 가능하다. 예를 들어 '로그인 한 사용자는 자신의 정보를 수정할 수 있다.' 라는 '사용자 정보 관리' 유즈케이스의 일부를 통해 필수 항해 포함 조건 *IN('main.html', 'account.html', 'login')*을 추출할 수 있다. 항해 설계의 다음 단계는 유즈케이스 별로 항해 모델을 작성하는 것이다. 항해 모델을 작성할 때에는 물리적 파일과 그들의 포함관계도 함께 고려하여 상세하게 작성한다. 항해 모델이 작성되면 항해 규칙을 기준으로 검증하고 오류가 발견될 경우 항해 모델을 수정한다. 해당 항해 모델의 검증이 완료되면 항해 모델을 이전 단계에 작성된 모델과 통합한 후 다시 검증하는 작업이 이루어진다. 이는 작성 당시에는 오류가 없었던 복수의 항해 모델을 통합하면서 생길 수 있는 오류를 검출하기 위해서 이루어지는 작업이다.

위와 같은 항해 설계 과정을 통해 항해 모델과 유즈케이스 간의 추적성(traceability)을 보장할 수 있다. 또한 단위 항해 모델들 간에 생길 수 있는 충돌을 제거할 수 있다.

4. 사례 연구

항해 설계 과정의 실제적인 적용을 보이기 위해 본 연구에서는 '학내 자료 관리 시스템(UNIVAS)'을 사례 연구로 구현하였다. UNIVAS는 Tomcat WAS와 mysql 데이타베이스를 이용하여 J2EE 환경에서 개발되었으며 이 중 항해 모델은 항해 모델의 설계를 지원하기 위해 본 연구에서 구현한 항해 모델 설계 도구를 이용하여 작성되었다. 항해 모델 도구는 오픈소스 프로젝트 중 하나인 ArgoUML[6]의 모듈을 이용하여 구현되었다.

항해 설계 과정의 순서에 따라 유즈케이스로부터 항해 규칙을 추출하고 정리한 후 항해 모델을 작성하였다. 예를 들어, 항해 규칙 *IN('main.html', 'componentDetail.jsp', 'login')*은 '컴포넌트의 상세 정보를 보기 위해 사용자는 로그인을 해야 한다.'는 요구 사항에서 추출한 것이다. 이 항해 규칙에 의하면 'main.html'에서 'componentDetail.jsp'로 항해가 존재하기 위해서는 반드시 'login' 항해를 거쳐야만 한다. 그림 5의 (a)는 작성된 '뷰 관점 항해 모델'의 일부이다. 이 모델을 항해 규칙을 기준으로 검증하면 오류가 없으므로 다음으로 기존의 항해 모델과 통합하여 다시 모델 검증을 실시하였다. 통합된 항해 모델인 그림 5의 (b)를 보면 'main.html'에서 'componentDetail.jsp'로 의 경로가 두 가지 존재하는데, 항해 규칙에 맞지 않는 경로가 존재함을 발견할 수 있었으며 이를 제거하기 위해서 'usingComponents.jsp'에서 'componentList.jsp'로 항하는 항해를 제거하였다.

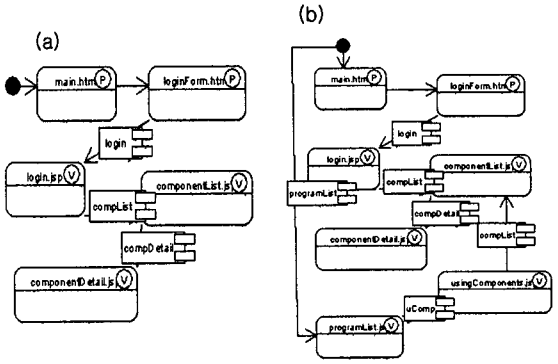

그림 5. UNIVAS 웹 응용의 항해 모델 일부

5. 결론 및 향후 연구

본 연구에서는 항해 모델의 단점인 복잡도 증가로 인한 모델 검증의 어려움을 해결하기 위해 항해 규칙을 이용한 모델 검증의 방법에 대해 제시하였다. 또한 항해 모델의 요소와 항해 규칙의 유형을 정형화하여 정의함으로써 모델 검증을 자동화할 수 있는 이론적 기반을 마련하였으며 모델 검증을 고려한 항해 설계 과정을 제안함으로써 유즈케이스로부터 구현에 이르기까지의 산출물들 간의 추적성을 보장하였다.

본 논문에서 제시된 항해 규칙의 네 가지 유형은 보다 많은 종류의 웹 응용을 분석하면 더욱 확장이 가능할 것으로 예상된다. 다양한 항해 오류의 경우에 대한 분석을 통해 항해 규칙의 유형을 다양화할 필요가 있다. 이와 더불어 항해 규칙을 이용한 자동화된 항해 모델 검증 도구의 구현이 필요하다. 모델로부터 코드를 자동 생성하거나 항해 규칙으로부터 테스트 케이스를 추출할 수 있도록 하는 등 작성된 항해 모델을 구현에 직접적으로 활용할 수 있는 방안에 대한 연구도 필요하다.

참고 문헌

[1] J. Conallen, "Building Web Applications with UML," 2nd ed., Addison Wesley, 2002.

[2] Andreas Kraus and Nora Koch, "A Metamodel for UWE," Technical Report 0301, Ludwig-Maximilians-Universitat, Munchen, 2003.

[3] Rolf Hennicker and Nora Koch, "A Uml-Based Methodolgy for Hypermdeida Design," Proc. of UML 2000 Conference, 2000.

[4] 박상현, 이욱진, 이병정, 김희천, 우치수, "UML 2.0 행위 다이어그램을 확장한 웹 응용의 항해 모델," 한국정보과학회 한국컴퓨터종합학술대회 2005, Vol. 32, No. 1, 2005.

[5] http://www.uml.org/

[6] http://argouml.tigris.org/