MVC 아키텍처 기반 애플리케이션의 GUI 테스팅†

(GUI Testing for MVC Architecture based Applications)

주희주‡ 이찬근¶
(Heeju Joo) (Changun Lee)

요 약 MVC는 유지보수가 쉬워 효율적인 개발이 가능한 아키텍처이다. 웹과 PC환경의 애플리케이션뿐만 아니라 모바일 애플리케이션 개발에 많이 적용됨에 따라 그에 대한 테스팅 또한 중요하다. 기존에 MVC 아키텍처를 적용한 개발에 초점을 둔 연구는 많았으나 MVC 아키텍처의 특성을 고려한 효율적인 테스팅을 위한 연구는 많지 않았다. 따라서 본 논문에서는 MVC 아키텍처 기반 애플리케이션의 효율적이고 정확한 테스팅을 위한 연구를 진행한다. 모델-뷰-컨트롤러의 융합된 상태를 담았던 기존의 상태 다이어그램을 뷰 상태와 모델-컨트롤러 상태 다이어그램으로 분리하고 테스트 케이스를 작성한 후, 테스트를 진행한다.

키워드 MVC 아키텍처, 상태 다이어그램, 상태 전이, 명세 기반 테스팅, GUI 테스팅, 테스트 케이스

Abstract MVC(Model-View-Controller) architecture is well-known for high maintainability and it makes efficient development possible. Recently, it has been applied in many fields such as web, desktop and mobile applications. Therefore, effective testing for this architecture is strongly needed. Although much research has been done, there was not much efforts for exploiting MVC architecture in GUI testing. Therefore, in this paper, we propose a scheme for efficient and accurate GUI testing for MVC based applications. In this study we separate original state diagrams into view state diagram and model-controller state diagram. Then, we present a case study showing the effectiveness of our proposed scheme.

Key words MVC architecture, state diagram, state transition, specification-based testing, GUI testing, test cases

1. 서 론

MVC(Model-View-Controller) 아키텍처는 UI와 프로그램 로직을 분리하여 유지보수가 쉬운 애플리케이션을 효율적으로 개발할 수 있기 때문에 로컬 애플리케이션과 웹 애플리케이션의 개발에 매우 널리 사용되고 있다. 최근 스마트 폰 시장이 발달됨에 따라 모바일 애플리케이션의 개발에도 그 적용 영역이 넓어지고 있다.

MVC를 이루고 있는 모델-뷰-컨트롤러 중 특히 GUI를 제공하는 뷰는 사용자와 밀접한 관계를 형성하는데, 이 때문에 더욱 뷰를 통한 테스팅이 중요하게 여겨진다. 단순히 GUI만을 살펴보는 테스팅이 아닌 프로그램 내부의 모델 및 컨트롤러의 상호작용의 테스팅 이기 때문에 테스트 케이스 생성에 신중해야 한다. 그러나 현재 MVC 아키텍처를 적용하는 개발 연구는 활발히 진행되는 반면 MVC 아키텍처 기반 애플리케이션에 특화된 효율적인 테스팅 방법에 대한 연구는 미미하다. 따라서 본 논문에서는 MVC 아키텍처로

† 본 연구는 한국연구재단 기초연구사업(과제번호 201100013924)의 지원을 받았습니다.
‡ 학생회원 : 중앙대학교 컴퓨터공학과
soulmateof88@gmail.com
¶ 종신회원 : 중앙대학교 컴퓨터공학부 교수. 교신저자
cglee@cau.ac.kr
논문접수 : 2011년 01월 07일
심사완료 : 2011년 02월 28일

개발된 애플리케이션의 상태 다이어그램을 뷰와 모델-컨트롤러의 두 가지로 분리하여 작성하고 테스트 케이스를 생성함으로써 좀 더 정확하게 명세 기반의 상태 전이 테스팅을 수행하는 방법을 제안한다.

2장에서 관련 연구를 통해 MVC 아키텍처에 대한 개념과 이를 적용한 개발 및 테스팅 연구 현황에 대해 살펴 볼 것이다. 3장에서는 상태 다이어그램의 분리 및 테스트 케이스 작성에 대해 제안한다. 4장에서는 MVC 아키텍처 기반의 모바일 애플리케이션으로 사례연구를 제시하고 이를 뷰와 상태 다이어그램으로 표현 한 후, 테스트 케이스를 생성하여 테스팅을 수행한다. 마지막으로, 제안하는 테스팅 방법의 차별성 및 향후 연구 방향을 검토하며 논문을 결론지을 것이다.

2. 관련 연구

2.1 MVC 아키텍처

MVC아키텍처는 1970년대 객체지향 프로그래밍의 개념 위에 설계된 언어인 Smalltalk를 이용한 애플리케이션 개발에 적용하기 위하여 도입되었다[3].

MVC 아키텍처는 모델, 뷰, 컨트롤러의 세가지 구성요소가 애플리케이션 내부에서 독립적인 역할을 하며 작업 요청을 통해 상호 작용을 한다. 다음 그림 1은 MVC 아키텍처의 구성 및 관계를 보여준다.

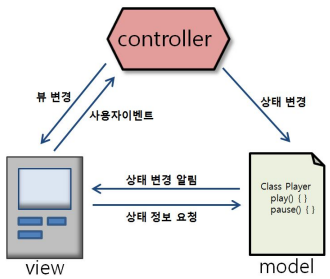

그림 1 MVC 아키텍처[3]

모델은 모든 데이터와 상태 및 로직을 처리한다. 뷰와 컨트롤러에서 데이터 상태를 조작하거나 가져오기 위한 인터페이스를 제공하며, 뷰와 컨트롤러에게 직접 관여하지 않고 독립적으로 존재한다. 뷰는 일반적으로 화면을 표현하는데 필요한 모든 처리를 담당하는데, 필요한 상태 및 데이터는 모델에서 가져온다. 컨트롤러는 사용자로부터 입력을 받아서 그것이 모델에게 어떤 의미가 있는지 파악한다. 컨트롤러는 모델에게 상태 변경을 요청하거나 뷰의 상태 변경을 요청할 수 있다. 사용자는 뷰를 통해서만 애플리케이션과 접촉이 가능하며 뷰는 모델에게 표시해야 할 상태를 요청한다. 모델의 상태가 변경되면, 모델은 뷰에게 상태가 변경되었음을 알린다.

2.2 상태 다이어그램

상태 다이어그램은 이벤트에 따른 객체의 상태 변화를 나타내는 다이어그램이다. 상태 다이어그램을 구성하는 요소로는 시작상태(Initial Pseudo State), 종료상태(Final State), 상태(State) 및 전이(Transition)가 있으며 상태를 변화시킬 수 있는 것은 전이이다. 이런 요소들을 조합하여 하나의 시스템 전체, 시스템의 일부 또는 개별 객체에 대한 동작을 나타내어 개발자와 유저 모두에게 유용한 정보를 제공할 수 있다. 본 논문에서는 상태 다이어그램을 이용하여 애플리케이션을 표현하고, 이를 테스팅에 활용한다.

2.3 MVC 아키텍처 개발 및 테스팅

MVC아키텍처를 적용한 애플리케이션의 개발 방법론에 대해서 많은 연구가 있었다. 웹 플랫폼 서비스 개발에 최적화 되어있는 아키텍처인 만큼 웹 분야에서는 이미 MVC 아키텍처를 이용한 연구가 활발하게 진행되고 있다[2][4]. 또한 EC system, stopwatch, command processing framework 등 다양한 서비스 개발 및 연구에 적용되었으며[1][8], 상태

다이어그램을 뷰와 모델-컨트롤러의 두 가지로 나누어 효율적인 개발에 접근한 연구[5]도 있다. 최근 스마트 폰 애플리케이션의 효율적인 개발을 위해 적극 적용하려는 연구[9]도 진행 중이다. 그러나 활발한 개발 연구와는 달리 MVC 아키텍처 기반의 애플리케이션을 위한 별도의 테스팅 기법에 관한 연구는 비교적 활발하지 못했다. MVC 아키텍처 기반의 웹 애플리케이션을 위한 테스팅에 관한 연구에는 추상 workflow 그래프 모델을 제안하여 state-based/code-based 테스팅을 수행한 연구[7]와, 모델-뷰-컨트롤러 각각을 다른 방법으로 테스팅 하고 각각 다른 접근 제어 방법으로 보안성을 높인 연구[10]가 있다.

3. MVC 애플리케이션의 테스팅 방법 제안

3.1 상태 다이어그램 분리

MVC 애플리케이션의 실행 중에 상태를 파악하는 것은 쉽지 않으나 코드를 분석하여 상태를 도출하는 것은 비교적 쉽게 수행할 수 있다[5]. 따라서 본 논문에서는 애플리케이션의 코드를 잘 알고 있는 개발자가 상태 다이어그램을 작성하고 분리한다고 가정한다. 상태 다이어그램 작성 시 상태 및 전이를 정확히 작성하여 테스팅 과정에서 놓치는 오류가 없도록 한다.

3.2 분리된 테스트 케이스 작성

분리하여 작성된 상태 다이어그램을 기반으로 테스트 케이스를 작성하는 과정은 비교적 간단하다. 각각의 '상태'가 Pre_Condition, Post_Condition이 되며, '전이'는 Transition으로 매칭 된다. 모든 경우를 조합하여 테스트 케이스를 작성한다. 다음 그림 2는 상태 전이와 테스트 케이스 간 매핑의 예를 보여준다.

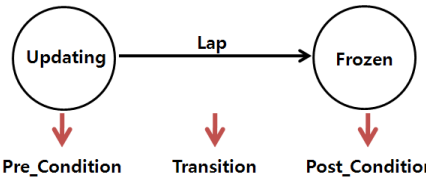

그림 2 상태 전이와 테스트 케이스 항목의 매핑

4. MVC 애플리케이션의 테스팅 방법 제안

4.1 Case Study

본 논문에서는 MVC 아키텍처로 개발된 애플리케이션의 테스트 과정을 보이기 위해 예제 프로그램을 이용한다. 예제 프로그램은 안드로이드 OS에서 음악을 재생하는 'SongPlayer' 애플리케이션이며 다음 그림 3과 같은 UI를 제공한다.

그림 3 'SongPlayer'의 UI

프로그램 내부의 모델-뷰-컨트롤러 역할 및 상호연결은 다음과 같다.

- 뷰: 'SongPlayer' 사용자에게 보여지는 모든 부분을 담당한다. 시간을 체크하는 Timer와 TimeLine, 노래 제목이 나타나는 부분, 그리고 정지/재생/일시 정지 버튼이 뷰에 해당한다.
- 컨트롤러: 사용자 이벤트를 입력 받아 모델에게 상태 변경을 요청한다. 예를 들어, 사용자가 '정지'버튼을 눌렀을 경우 이벤트를 받아 모델에게 전달하여 재생 중인 노래를 멈추도록 요청한다.

• 모델: 실제 'SongPlayer'의 모든 로직을 담당한다. 컨트롤러로부터 상태 변경을 요청 받고, 변경된 상태를 뷰에게 전달한다. 재생 중인 노래를 일시 정지 했을 때, 그 상태를 뷰에게 전달하여 화면에 '일시 정지' 버튼 대신 '재생'버튼이 보이도록 한다.

4.2 상태 다이어그램 생성

예제 프로그램의 상태 다이어그램은 그림 4와 같다.

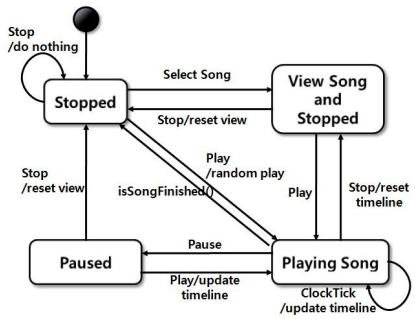

그림 4 'SongPlayer'의 상태 다이어그램

상태 다이어그램으로 예제 프로그램의 상태와 전이에 대해 한눈에 볼 수 있다. 애플리케이션의 상태는 크게 'Stopped(정지)', 'View Song and Stopped(노래 제목은 보이지만 플레이어 정지 상태)', 'Playing Song(재생)', 'Paused(일시 정지)' 4가지로 나타내어지며 각 상태로의 전이는 그림 4에 제시되어 있다.

본 논문에서 제시하는 MVC의 효율적인 테스팅을 위해 모델-컨트롤러 간의 상태와 뷰의 상태 두 가지로 분리하여 그린 상태 다이어그램은 다음 그림 5, 그림 6과 같다.

그림 5에서와 같이, 'SongPlayer'의 모델-컨트롤러 간의 상태 다이어그램은 음악을 재생하는 Player 부분과 타이머 뷰의 상태를 변화시키는 Timer 부분의 동시 상태로 표현된다. 사용자 눈에 직접 보이진 않지만, 내부적으로 프로그램의 상태를 변화 시키는 로직의 상태를 볼 수 있다.

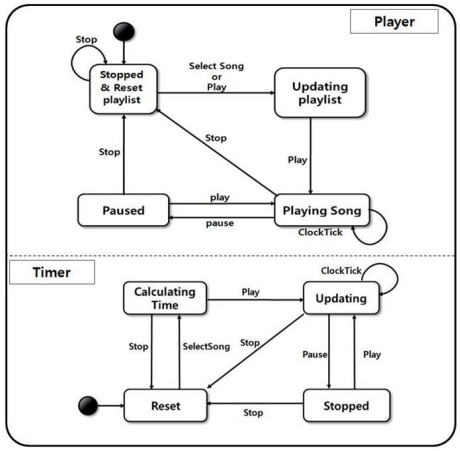

그림 5 'SongPlayer'의 모델-컨트롤러 상태 다이어그램

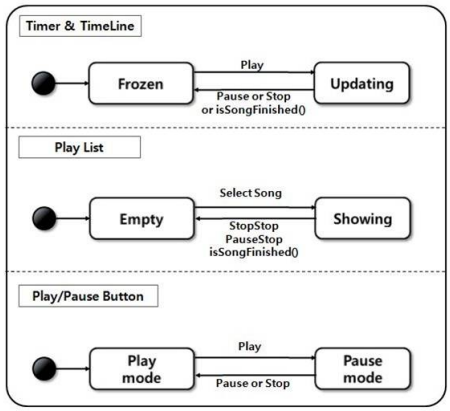

그림 6 'SongPlayer'의 뷰 상태 다이어그램

그림 6는 'SongPlayer'의 뷰 상태 변화를 보여준다. 뷰는 노래 시간을 나타내는 Timer & TimeLine 부분과 노래 제목을 나타내는 Play List 그리고 Play/Pause Button의 동시 상태로 나뉘어진다. 내부 로직을 알 필요가 없이, 사용자 눈에 보이는 뷰의 상태 변화만을 분리해 나타내었다. 이제 상태 다이어그램을 두 가지로 분리하였다. 뷰 상태 다이어그램은 유저 GUI 테스팅을 위한 테스트 케이스 생성에 이용되며, 내부 로직 테스팅을 위한 테스트 케이스는 모델-컨트롤러 상태 다이어그램으로부터 생성 된다.

4.3 분리된 테스트 케이스

상태 다이어그램을 분리함으로써 뷰만의 상태를 좀 더 자세히 볼 수 있게 되었다. 이는 MVC 아키텍처 기반 애플리케이션의 GUI 테스팅을 위한 중요한 시나리오가 된다. 뷰 상태 다이어그램으로부터 도출한 테스트 케이스의 일부는 다음 표 1과 같다.

표 1 뷰 상태 테스트 케이스

No.	Pre_ Condition	Transition	Post_ Condition	P/F
1	Frozen	Play	T_Upd	
2	Frozen	Play	TL_Upd	
3	Updating	Pause	T_Stp	
4	Updating	Stop	T_Stp	
5	Updating	isSongFinished	T_Stp	
…	…	…	…	…
15	Pause_md	Stop	Play_md	

모델-컨트롤러 상태 다이어그램은 내부 로직 개발자 및 테스터를 위한 시나리오를 제공하며, 테스터는 뷰를 통해 사용자 입력 값을 컨트롤러와 모델에게 전달한다. 모델-컨트롤러간의 상태 다이어그램으로부터 도출한 테스트 케이스는 다음 표 2와 같다.

표 2 모델-컨트롤러 상태 테스트 케이스

No.	Pre_ Condition	Transition	Post_ Condition	P/F
1	Stopped	Stop	DoNth	
2	Stopped	Select Song	Upd_PL	
3	Stopped	Play	Upd_PL	
4	Upd_PL	Play	Playing	
5	Playing	Pause	Paused	
…	…	…	…	…
15	t_Cal	Play	t_Upd_ing	
16	t_Cal	Stop	t_reset	
17	t_Upd_ing	ClockTick	t_Upd_ing	

4.4 테스트 결과

테스트를 위해 개발자와 베타 테스터에게 테스트 케이스를 제공하였다. 뷰 테스팅의 결과는 테스터의 판단을 통해 성공/실패 여부를 결정하였으며, 모델-컨트롤러 테스팅의 결과는 테스팅을 위해 코드를 약간 수정하여 로그가 출력되게 하였고 로그 출력 결과를 통해 테스트의 성공/실패 여부를 결정하였다.

결과의 비교를 위해 분리되지 않은 상태로 수행된 테스트(테스트 1)와 분리된 상태로 수행된 테스트(테스트 2)를 진행하였으며 두 테스트는 동일한 환경에서 진행되었다. 테스트 1은 케이스가 많지 않아 테스트 시간이 비교적 적었으나 미처 발견하지 못한 오류가 비교적 많았다. 다음 그림 7은 두 테스트에 대한 수행 결과 비교 그래프이며, 이를 바탕으로 표 3에서 두 가지 테스트의 장단점을 비교해 보았다.

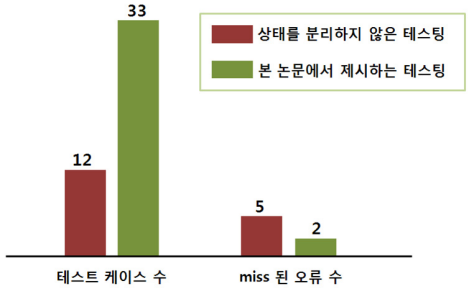

그림 7 테스트 수행 결과 비교

표 3 제안하는 테스팅과의 비교

	상태를 분리하지 않은 테스팅	본 논문에서 제안하는 테스팅
장점	· 테스트에 드는 비용(시간)이 짧다. · 모델/뷰/컨트롤러의 상태를 동시에 테스팅 할 수 있다.	· 테스트 Fail시 수정 할 부분을 빠르게 인지할 수 있다. · 뷰 테스팅의 진행이 쉽다.
단점	· 세밀한 테스트 케이스 생성이 어렵다. · 테스트 Fail시 수정 할 부분을 인지하는 데에 비용(시간, 노력)이 많이 필요하다.	· 테스트에 드는 비용(시간)이 길다. · 모델-컨트롤러의 테스팅 준비 작업을 위해 기존 소스 코드를 수정해야 한다.

5. 결 론

본 논문에서는 MVC 아키텍처 기반 애플리케이션을 효율적으로 테스팅 하기 위해, 상태 다이어그램을 분리하여 분리된 테스트 케이스를 도출하는 테스팅 방법을 제안하였다. MVC 아키텍처는 모델-뷰-컨트롤러의 세 가지 구성요소가 애플리케이션 내부에서 독립적인 역할로 상호작용하는 구조를 가지며, UI와 프로그램 로직을 분리하여 유지보수가 쉬운 애플리케이션을 효율적으로 개발할 수 있기 때문에 로컬 애플리케이션과 웹 애플리케이션의 개발에 매우 널리 사용되고 있다.

본 논문에서 제안하는 테스팅 방법은 상태 다이어그램을 통해 테스트 케이스를 도출한 명세 기반의 GUI 테스팅으로, 상태 다이어그램을 분리하기 전 보다 더 세밀한 테스트를 진행할 수 있다. 특히 뷰 상태만을 테스팅 하거나 모델-컨트롤러만을 테스팅 할 때 유용하여 MVC 아키텍처 기반의 애플리케이션 테스팅을 효율적으로 수행할 수 있다. 또한, 상태 다이어그램을 기반으로 하기 때문에 특정 개발 환경에 영향을 받지 않는 테스팅이 가능하다. 그러나 상태 다이어그램을 분리하고 그로부터 테스트 케이스를 작성하는 과정에서 시간이 많이 소요된다는 등의 단점을 가진다.

향후 연구 방향으로는 본 논문의 MVC 아키텍처 GUI 테스팅을 Model-Based GUI Testing에 관한 연구[6]로 확장하여 조금 더 체계적인 뷰/모델-컨트롤러 테스트 모델을 설계한다. 또한, 테스트 케이스 생성을 자동화하는 방법에 대해 제안하고 구현하여 테스팅 비용을 감소시키는 연구를 진행할 것이다.

참 고 문 헌

[1] 서순모, 양해술, “MVC Architecture 기반의 EC System용 상품전시 컴포넌트의 설계 및 구현,” 한국정보과학회 2001년도 가을 학술발표논문집 제28권 제2호(1), 2001.

[2] T. Bodhuin et al., “Migrating COBOL Systems to the WEB by using the MVC Design Pattern,” In Proc. of Working Conference on Reverse Engineering, 2002.

[3] E. Freeman and E. Freeman, Head First Design Patterns, O Reilly, 2004.

[4] X. Qiu, “Building Desktop Applications with Web Services in a Message-based MVC Paradigm,” In Proc. of IEEE International Conference on Web Services, 2004.

[5] S. Hansen and T. V. Fossum, “Refactoring Model-View-Controller,” Journal of Computing Sciences in Colleges, Vol. 21, Issue. 1, 2005.

[6] A. Kervinen et al., “Model-Based Testing Throu-gh a GUI,” In Proc. of International Workshop on Formal Approaches to Testing of Software, 2006.

[7] M. Karam et al., “An Abstract Model for Testing MVC and Workflow Based Web Applications,” In Proc. of Advanced Int’l Conference on Telecommunications and Internet and Int’l Conference on Web Applications, 2006.

[8] A. Naderlinger and J. Templ, “A framework for command processing in Java/Swing programs based on the MVC Pattern,” In Proc. of 6th international symposium on Principles and practice of programming in Java, 2008.

[9] 이호중 외 2명 “서비스 기반 모바일 애플리케이션의 MVC 아키텍처 및 적용 사례연구,” 정보과학회논문지 : 컴퓨팅의 실제 및 레터 제16권 제11호, pp.1111-1115, 2010.

[10] Q. Wu et al., “Unit Testing and Action-Level Security Solution of Struts Web Applications Based on MVC,” In Proc. of International Conference on Biomedical Engineering and Computer Science, 2010.

저 자 소 개

주 희 주

2011년 중앙대학교 컴퓨터공학부 학사.
2011년~현재 중앙대학교 컴퓨터공학과 석사과정.

〈관심분야〉 실시간 소프트웨어, 소프트웨어 테스팅.

이 찬 근

1996년 중앙대학교 전자계산학과 학사
1998년 KAIST 전산학과 석사.
2005년 Univ. of Texas at Austin 전산학과 박사.
2005년~2007년 미국 인텔 소프트웨어 엔지니어.
2007~현재 중앙대학교 컴퓨터공학부 조교수.

〈관심분야〉 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅