안드로이드 스마트폰 어플리케이션을 위한 테스트 용이성 분석 연구

장우성*, 손현승*, 김우열*, 김영철*
*홍익대학교 컴퓨터정보통신공학과
e-mail:{jang, son, john, bob}@selab.hongik.ac.kr

A Study on Analysis of Testability for Android Smart-phone Application

Woo-Sung Jang*, Hyun-Seung Son*, Woo-Yeol Kim*, R. Young-Chul Kim*
*Dept. of CIC, Hongik University, Jochiwon, Korea

요 약

스마트폰 어플리케이션은 소프트웨어의 평가를 구매자가 쉽게 확인 및 작성할 수 있어 품질이 매출에 직접적으로 영향을 끼쳐 소프트웨어의 품질을 향상시키기 위해서 테스트가 요구된다. 하지만 기존의 스마트폰 어플리케이션은 테스트 용이성을 고려하지 않고 개발되어 테스트를 위해 많은 비용이 증가한다. 본 논문은 이 문제를 해결하고자 소프트웨어 설계 단계에서 모델변환을 적용하여 테스트 용이성을 향상 시키는 방법을 제안한다. 대상 모델은 UML의 클래스 다이어그램이고 테스트 용이성 측정을 위해서 Binder[7]방법을 사용한다. 적용사례로 안드로이드 기반의 소프트웨어인 SnakePlus를 구현하고, 이를 대상으로 설계 모델을 모델변환을 하여 테스트 용이성을 향상 시킨다.

1. 서론

최근 소프트웨어의 품질이 중요해지면서, 소프트웨어 개발 분야는 보다 다양하고 향상된 테스트 방법을 요구하고 있다. 최근 이슈가 되는 스마트폰의 경우 이러한 소프트웨어의 테스트가 중요하다. 아이폰, 안드로이드폰에 설치되는 소프트웨어의 경우 고객과 개발자의 거래가 앱스토어를 통해 일대일 관계로 이루어지고, 소프트웨어의 평가가 앱스토어 내에 공개되므로 소프트웨어 품질의 저하는 결국 소프트웨어의 판매 수에 직접적인 영향을 끼치게 된다. 이처럼 최근에 소프트웨어 테스트가 중요하게 되었지만, 일반적인 스마트폰 소프트웨어는 복잡도가 높아 테스트를 수행하기 어렵다. 처음부터 테스트가 쉬운 구조로 소프트웨어를 설계하기 위해서는 설계자가 많은 경험을 가지고 있어야 한다. 하지만, 경험이 부족한 개발자가 소프트웨어의 테스트를 위해 개발 도중 혹은 완료 후 구조를 변경할 경우 변경 과정에 많은 시간이 소모되어 추가적인 비용이 발생하게 된다.

기존의 테스트 가능성 분야의 연구는 Data Flow Analysis를 사용하여 소프트웨어 테스트 용이성을 측정하는 연구[1], Observerability와 Controllability 개념을 적용하여 도메인 테스트 용이성을 연구[2], 디자인 패턴을 이용하여 테스트 용이성을 개선하는 연구[3] 등의 방법 들이 제시 되었다. 하지만 이들 모두 일반적인 소프트웨어를 대상으로 수행하였고 자동화된 방법이 제공되지 않아 적용에 어려움이 있다.

모델 변환 방법은 플랫폼에 독립적인 메타모델을 설계한 후 필요한 기술 모델을 변경하여 그 모델을 통해 코드 생성을 자동화하는 메커니즘이다. 독립 모델의 재사용성을 높여 모델과 관련된 코드의 생산성을 높일 수 있다[4]. 즉, 스마트폰 개발환경에 모델변환기법을 적용하면 하나의 모델을 이종의 모델로 자동생성이 가능하다[5, 6].

본 논문은 소프트웨어 설계 단계에서 모델변환을 적용하여 테스트 용이성을 향상 시키는 방법을 제안한다. 제안한 방법은 Binder[7]의 테스트 용이성 측정 메트릭의 PAP, PAD, NOC 속성을 모델 변환 규칙으로 만든다. 이 규칙을 모델 변환에 적용하여 안드로이드 설계 모델을 모델변환 한다. 모델의 변환은 테스트 용이성을 체크하는 리스트를 기반으로 하여 수행된다. 이러한 모델 변환 적용으로 테스트 용이성 향상이 자동화가 가능해지고 정확성이 증가되고, 테스트 비용을 감소시킬 수 있다.

적용사례로 안드로이드 기반의 게임 Snake를 확장한 소프트웨어인 SnakePlus를 구현하는 과정에 제시한 3가지 규칙을 적용하고 이를 대상으로 설계 모델을 모델변환을 하여 테스트 용이성을 향상 시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 모델 변환 방법과 테스트 용이성 측정 방법에 대해서 설명한다. 3장에서는 제안한 모델 변환을 이용한 테스트 용이성 향상 기법에 대해서 언급한다. 4장에서는 적용사례로 SnakePlus의 구조에 모델 변환 기법을 적용하여 테스트 용이성 향상을 검증한다. 마지막 장에서는 결론 및 향후 연구를 언급한다.

2. 관련연구

2.1. 모델변환방법

모델 변환은 입력되는 소스 모델을 대상 모델로 변환하는 방법이다. 모델 변환의 종류에는 모델에서 모델, 모델에서 코드, 코드에서 모델 등이 있다. 또한 모델 변환에서

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2010-0012117).

사용되는 모델은 UML 뿐만 아니라 컨트롤 플로우 다이어그램, 데이터 플로우 다이어그램 등 다양한 모델로도 변환이 가능하다. UML은 UML 메타모델이 지원이 되기 때문에 메타모델을 만들지 않아도 되지만 메타모델이 없는 모델일 경우 MOF(Meta Object Facility)를 이용하여 메타모델로 설계가 요구된다. 모델 변환을 수행하기 위해서는 소스 모델, 소스 메타모델, 타겟 메타모델, 변형 정의가 필요하다. 모델 변환 엔진은 이러한 요소들을 사용해서 타겟 모델로 변환한다[8]. 모델 변환에서 중요한 요소로는 변환 룰을 정의하는 언어에 있다.

2.2. 테스트 용이성 측정 방법

테스트 용이성(testability)은 IEEE에서 발행한 소프트웨어 관련 용어집 내에 "시스템 또는 컴포넌트가 테스트 기준을 확립하고 테스트를 쉽게 할 수 있는 정도" 그리고 "요구사항에서 테스트 기준의 확립을 위해 사용된 용어의 정도"라고 정의되어 있다.[9] 테스트의 용이성을 향상시키기 위해서는 설계 단계에서부터 테스트 용이성을 고려하여 설계하는 것이 중요하다. 개발의 중간 단계에서 테스트의 용이성을 고려한다면, 코드 상의 변경이 필요하게 되어 소프트웨어의 개발 비용이 증가하게 된다. 이러한 테스트의 용이성을 측정하기 위한 기준은 여러 가지 측면이 존재한다. 각 관점에 따라 테스트의 용이성 정도는 달라지고, 수정해야할 부분 또한 달라진다. James Bash가 제시한 테스트 용이성 요소는 크게 제어성, 관찰성, 유효성, 간편성, 안정성, 정보로 나뉜다[10].

Binder의 테스트 용이성 측정 메트릭은 소프트웨어의 구조를 여러 측면에서 측정하여 산술적인 측정치를 산출해낸다. 측정치는 값이 높을수록 테스트 용이성이 낮다는 것을 의미한다. 아래의 <표 2>는 Binder의 구조적 테스트 용이성을 측정하기 위한 메트릭이다.

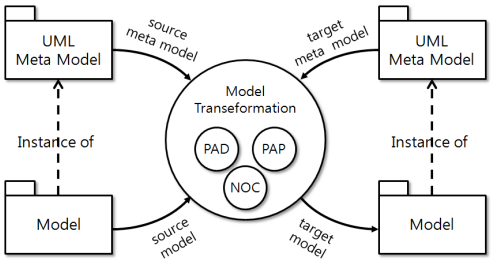
<표 2> 테스트 용이성 측정 메트릭[3]

Encapsulation metric	
LCOM	오직 하나의 메서드에 의해서 사용되는 인스턴스 변수의 그룹의 수
PAP	public 또는 protected C++ 클래스의 데이터 멤버의 비율
PAD	클래스 내 public 또는 protected 데이터멤버로의 외부접근의 갯수
Inheritance metric	
NOR	프로그램에 의해 사용되는 뚜렷한 클래스 계층의 갯수
FIN	클래스를 파생시킨 클래스(슈퍼클래스)들의 갯수
NOC	특정한 부모 클래스로부터 파생된 클래스들의 갯수
DIT	클래스 내부의 계층의 레벨
Polymorphism metric	
OVR	오버로드 되지 않은 콜의 비율
Complexity metric	
WMC	클래스 내 메서드들 수행의 순환복잡도의 합계
RFC	클래스 내 메서드들이 시행된 갯수
CBO	한 클래스에서 다른 클래스의 메소드나 데이터 멤버에 접근하는 수

3. 제안한 테스트 용이성 향상 기법

3.1. 모델 변환을 이용한 테스트 용이성 향상 방법

테스트의 용이성을 향상시키기 위해서는 소프트웨어의 구조를 변경하여야 한다. 하지만 설계가 완료된 소프트웨어는 비용의 소모가 커지기 때문에 다시 구조를 변경하기가 쉽지 않다. 이러한 문제를 해결하기 위해서는 수작업이 아닌 프로그램에 의한 자동화 된 작업이 필요하다. 자동화된 작업은 개발 비용을 감소시키고, 수작업에 비해 좀 더 안정적으로 소프트웨어 구조를 변경한다. 본 논문은 타겟인 안드로이드 플랫폼의 소프트웨어 구조를 모델 측면에서 변화시키는 방법을 통해 테스트 용이성을 향상하였다. 모델이 다른 모델로 변환하기 위해서는 UML 메타 모델을 통해 변환을 하게 된다. 변환에 필요한 조건은 Binder의 테스트 용이성 측정 메트릭을 통해 해결하였다. 테스트 용이성 수치를 산출해낼 수 있는 Binder의 테스트 용이성 측정 메트릭에서 테스트 용이성을 떨어뜨릴 수 있는 케이스를 찾고, 모델을 변환을 통해 이러한 케이스를 제거한다. 이러한 방법은 코드의 정형적인 변환이 가능하게 되어 프로그램화 할 경우 테스트 용이성을 자동적으로 증가 시킬 수 있다. 모델 변환에 대한 그림은 아래의 (그림 1)과 같다.

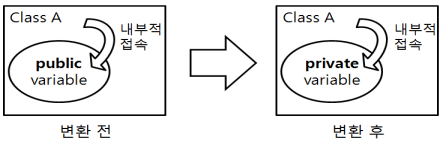


(그림 1) 모델 변환도

3.2. 테스트 용이성 향상을 위한 모델 변환 규칙

3.2.1. PAP 속성 규칙

PAP 속성은 접근 권한이 public 또는 protected인 클래스 내의 데이터 멤버의 비율을 말한다. 이 비율이 증가할수록 다른 오브젝트들에게 보이는 변수가 증가할 수 있으며, 결국 높은 PAP 속성은 클래스 내의 부작용이 더 많이 생겨날 수 있음을 의미한다. PAP 속성의 테스트 용이성을 증가시키기 위해서는 클래스 내부의 클래스 변수들을 검사하여 하위 클래스 또는 외부 클래스에서 사용되지 않고 오직 자신의 클래스 내부에서만 사용되는 클래스 변수의 속성을 private로 변경한다. 이를 통해 다른 오브젝트들에게 보이는 클래스 변수의 개수를 최소화 할 수 있다.

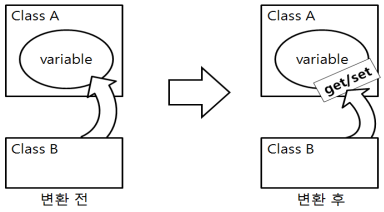


(그림 2) PAP 속성 향상 기법

3.2.2. PAD 속성 규칙

PAD 속성은 외부 클래스가 클래스 내 public 또는 protected 데이터멤버로 접근하는 개수를 말한다. PAD 속성이 높을수록 캡슐화의 위반이 높다는 것을 의미하고, 이것은 즉 클래스 내 부작용의 확률이 높아진다는 것을 의미한다. PAD 속성의 테스트 용이성을 증가시키기 위해서는 클래스 내부의 클래스 변수들을 검사하여 외부에서 접근하는 클래스 변수의 경우 캡슐화를 한다. 캡슐화가 적용

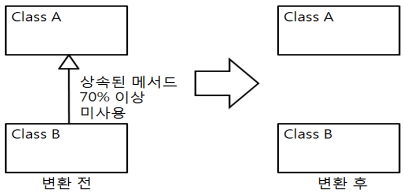
된 클래스 변수는 private 형태로 바뀌게 되어 PAP 속성의 테스트 용이성을 증가시킨다. 동시에 외부의 접근에 대한 캡슐화의 위반을 줄여준다.



(그림 3) PAD 속성 향상 기법

3.2.3. NOC 속성 규칙

NOC 속성은 특정한 부모 클래스로부터 파생된 클래스의 개수를 말한다. NOC 속성이 높을수록 부모 클래스를 수정할 때 다시 테스트 되어야 하는 자식 클래스가 늘어남을 뜻한다. NOC 속성의 테스트 용이성을 증가시키기 위해서는 부모 클래스에서 사용되는 메서드의 개수와 최종적인 자식 클래스에서 사용되는 메서드의 개수를 비교하여, 부모 클래스에서 물려받은 메서드가 최종적인 자식 클래스에서 대부분 사용되지 않을 경우 부모 클래스의 메서드를 자식 클래스의 메서드에 생성한 후 서로 간의 상속 관계를 제거한다. 이를 통해 비능률적인 상속을 최대한 제거하여 특정한 부모 클래스로부터 파생된 클래스들의 개수를 최소화 한다.

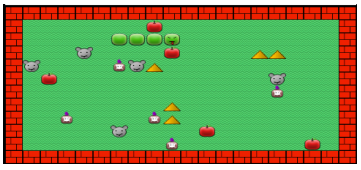


(그림 4) NOC 속성 향상 기법

4. 적용사례

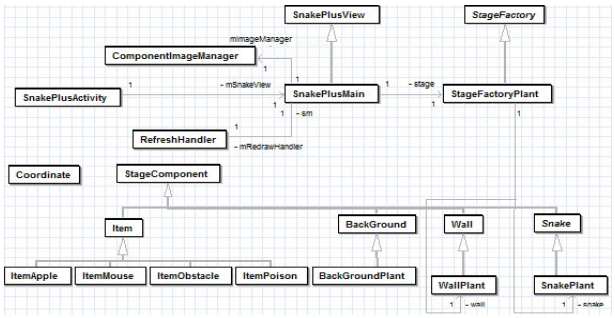
4.1. 개발한 SnakePlus의 모델변환

SnakePlus는 1980년대 인기 아케이드 게임인 뱀 게임을 모티브로 만든 안드로이드폰 전용 게임이다. 기존의 간단한 뱀 게임에 다양한 그래픽 요소를 추가하고, 터치 기능을 이용하여 총 여덟 방향으로 이동이 가능하며, 다양한 아이템을 추가하여 재미 요소를 추가하였다. SnakePlus의 실행화면은 아래의 (그림 5)와 같다.



(그림 5) SnakePlus 실행 화면

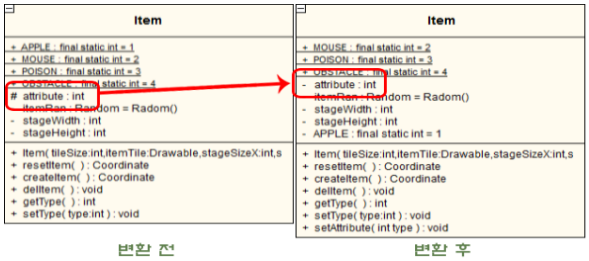
SnakePlus는 안드로이드 플랫폼에 맞추어 개발이 되었기 때문에 코드가 자바로 구성이 되어있으며, 총 20개의 클래스가 존재한다. BackgroungPlant, WallPlant, SnakePlant, ItemApple, ItemMouse, ItemPoison, ItemObstable 클래스들은 모두 StageComponent 클래스를 상속받고 있으며, 객체의 정보를 담당한다. StageFactoryPlant 클래스는 객체들을 직접적으로 컨트롤하여 좌표를 움직이게 하고, 객체의 좌표를 계산하여 게임의 진행 및 종료 여부를 결정한다. SnakePlusView는 사용자의 입력을 받아 StageFactorty 클래스에 입력 정보를 넘겨주고, 타이머를 통해 0.1초 간격으로 StageFactoryPlant 클래스를 실행한다. 클래스들의 전체적인 구조도는 아래의 (그림 6)과 같다.



(그림 6) SnakePlus 클래스 구조도

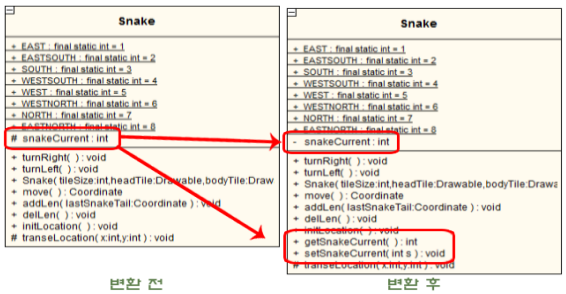
SnakePlus의 클래스들을 PAP, PAD, NOC 속성을 위주로하여 모델 변환할 경우 Item, Snake, SnakePlusMain, SnakePlusView 클래스가 테스트 용이성이 낮은 조건에 해당되어 모델 변환이 이루어지게 된다.

Item 클래스는 PAP 속성에 의해 attribute 변수 타입이 protected 형태에서 private 형태로 변환된다. 이에 대한 그림은 아래의 (그림 7)과 같다.



(그림 7) Item클래스의 변환 전후

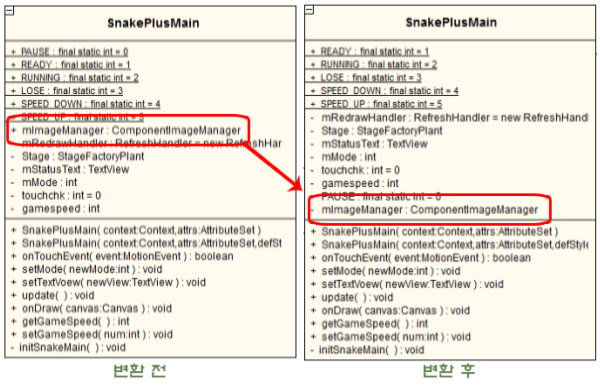
Snake 클래스는 PAP 속성에 의해 snakeCurrent 변수 타입이 protected 형태에서 private 형태로 변환되고, snakeCurrent는 다른 클래스에서 참조가 되는 변수이기 때문에 PAD 속성에 의해 get/set 메서드가 추가된다. 이에 대한 그림은 아래의 (그림 8)과 같다.



(그림 8) Snake클래스의 변환 전후

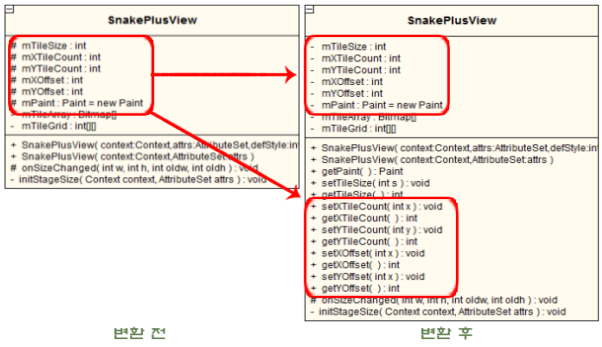
SnakePlusMain 클래스는 PAP 속성에 의해 mImageManager 변수의 타입이 public 형태에서 private

형태로 변환된다. 이에 대한 그림은 아래의 (그림 9)와 같다.



(그림 9) SnakePlusMain클래스의 변환 전후

SnakePlusView 클래스는 PAP 속성에 의해 mTileSize, mXTileCount, mYTileCount, mXOffset, mYOffset, mPaint, mTileArray, mTileGrid 변수들의 타입을 private 형태로 변환한 후 PAD 속성에 의해 변수들의 get/set 메서드를 추가한다. 이에 대한 그림은 아래의 (그림 10)과 같다.



(그림 10) SnakePlusView클래스의 변환 전후

4.2. 테스트 용이성 측정 결과

Binder의 구조적 테스트 용이성 측정 메트릭을 이용하여 테스트 용이성을 측정한 결과는 아래의 <표 3>과 같다.

<표 3> 테스트 용이성 측정 결과

속성	변환 전	변환 후
LCOM	4	4
PAP	50%	38%
PAD	59	33
NOR	14	14
FIN	7	7
NOC	13	13
DIT	3	3
OVR	0	0
WMC	71	72
RFC	276	303
CBO	128	154

PAP 속성이 50%에서 38%로 감소, PAD 속성이 59에서 33으로 감소, NOC 속성은 값의 변경 없이 유지가 되었다. 그리고 WMC 속성이 71에서 72로 증가, RFC 속성이 276에서 303으로 증가, CBO 속성이 128에서 154로 증가하였다.

5. 결론 및 향후 연구

스마트폰 어플리케이션은 소프트웨어의 평가를 구매자가 쉽게 확인 및 작성할 수 있어 품질이 매출에 직접적으로 영향을 끼쳐 소프트웨어의 품질을 향상시키기 위해서 테스트가 요구된다. 하지만 기존의 스마트폰 어플리케이션은 테스트 용이성을 고려하지 않고 개발되어 테스트를 위해 많은 비용이 증가한다. 본 논문은 이 문제를 해결하고자 소프트웨어 설계 단계에서 모델변환을 적용하여 테스트 용이성을 향상 시키는 방법을 제안한다. 대상 모델은 UML의 클래스 다이어그램이고 테스트 용이성 측정을 위해서 Binder방법을 사용한다. 적용사례로 안드로이드 기반의 소프트웨어인 SnakePlus를 구현하고, 이를 대상으로 설계 모델을 모델변환을 하여 테스트 용이성을 향상 시킨다. 모델 변환을 통하여 테스트 용이성을 개선한 결과로 PAP, PAD 속성의 테스트 용이성이 증가됨을 확인하였다. 하지만 변환 과정 중 많은 get/set 메서드가 생성이 되었고, 이 때문에 WMC, RFC, CBO 속성의 테스트 용이성이 감소됨을 확인하였다.

향후연구로 PAP, PAD, NOC 속성 이외의 다른 속성의 측면에서도 테스트 용이성을 향상시킬 것이며, 안드로이드 플랫폼 이외의 다른 스마트폰의 플랫폼에도 테스트 용이성의 향상을 적용할 수 있도록 확장할 것이다.

참고문헌

[1] Pu-Lin Yeh; Jin-Cherng Lin, "Software testability measurements derived from data flow analysis", Second Euromicro Conference, pp. 8-11, March 1998.

[2] Roy S. Freedman, "Testability of Software Components", IEEE Trans. on Software Engineering, 1991.

[3] 강영남, 최은만, "디자인 패턴 기반 소프트웨어의 테스트 가능성 분석", 한국정보과학회, 2005

[4] Woo Yeol Kim, Hyun Seung Son, R. Young Chul Kim, C. R. Carlson,"MDD based CASE Tool for Modeling Heterogeneous Multi-Jointed Robots", 2009 CSIE, Vol. 7, 775-779, 2009.04.01.

[5] 손현승, 김우열, 장우성, 김영철, "모델 변환을 이용한 안드로이드 어플리케이션 개발", Vol. 7, No. 1, pp. 64-67, KSEJW 2010, 2010.08.19

[6] 손현승, 김우열, 김재승, 김영철, "모델 변환을 이용한 윈도우즈 모바일 어플리케이션 개발", Vol. 37, No. 1, pp. 72-73, 한국정보과학회, 2010.06.30

[7] R. V. Binder, "Design for Testability in Object-Oriented Systems", Communications of the ACM, Vol. 37, No. 9, pp. 87-101, 1994.

[8] K. Czarnecki, S. Helsen, "Feature-Based Survey of Model Transformation Approaches.IBM Systems Journal", Vol. 45, No. 3, pp. 621-64, 2006.

[9] IEEE Std 610. 12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990.

[10] Heuristics of Software Testability by James Bach, Satisfice, Inc.