테스트 주도 개발(TDD)에서의 모바일 응용 소프트웨어 성능 테스트 방안

김희진°, 최병주, 윤석진
이화여자대학교, 한국전자통신연구원
heejinkim@ewhain.net°, bjchoi@ewha.ac.kr, sjyoon@etri.re.kr

Performance Testing for Mobile Application Software in Test-Driven Development

Heejin Kim°, Byoungju Choi, Seokjin Yoon
Ewha Womans University, Electronics and Telecommunications Research Institute

요 약

테스트 주도 개발(TDD)에서는 무엇보다 테스팅의 중요성이 강조되고 있으며, 효율적인 단위 테스트를 통해 소프트웨어를 신속하게 개발할 수 있도록 자동화된 테스트 프레임워크를 지원한다. 본 논문은 소프트웨어를 개발하는 과정에서 소프트웨어의 기능뿐만 아니라 비기능적인 요소(non-functional factor)인 성능도 함께 고려하여 모바일 응용 소프트웨어를 개발하도록 하는 성능 테스트 방안을 제안한다.

본 논문에서는 모바일 응용 소프트웨어 성능 테스트 현황과 이슈를 살펴보고, 모바일 응용 소프트웨어 성능 테스트를 위해 필요한 성능 특성들을 분석하여, 테스트 주도 개발의 특징을 반영한 성능 테스트 방안에 대해 제시하고자 한다.

1. 서 론

최근 몇 년 동안 주목을 받고 있는 테스트 주도 개발(Test-Driven Development; TDD)은 자동화된 테스트로 개발을 이끌어가는 점진적인 소프트웨어 개발 방법으로[1], 작동하는 깔끔한 코드를 만들기 위해 효율적인 단위 테스트의 중요성을 강조하며, 소프트웨어를 신속하게 개발할 수 있도록 JUnit[2]과 같은 자동화된 테스트 프레임워크를 지원한다.

테스트 주도 개발에서는 개발해야 할 소프트웨어 기능을 작은 단위로 나누어 테스트를 먼저 작성하고, 단위 테스트를 바탕으로 소프트웨어를 개발한다. 여기에서 단위 테스트(Unit Test)는 테스트 대상이 되는 코드 기능의 아주 작은 특정 영역을 실행해 보는, 개발자가 작성한 코드 조각을 의미한다[3]. 이러한 테스트 주도 개발은 디버깅에 낭비하던 시간을 극적으로 줄여주고, 더 나은 코드 디자인을 제공함으로, 빠른 개발 속도 및 신뢰성 있는 소프트웨어를 개발할 수 있도록 도와준다[4].

일반적으로 단위 테스트를 통하여 명세된 대로 기능이 제대로 잘 작동된다는 것을 확신하게 되면, 성능 테스트를 통해 통합된 코드와 비기능적인 시스템 요구사항을 비교하여 테스트를 수행한다[5]. 즉, 제품이 완성되기 전에는 성능 테스트가 거의 수행되지 않는다. 하지만, 개발과정에 기능 테스트 결과뿐만 아니라, 성능 테스트 결과들도 반영하여 소프트웨어를 개발한다면, 소프트웨어의 품질 향상 및 신뢰성을 높이는데 상당한 도움을 줄 수 있을 것이다[6]. 따라서 개발 후반에 집중되어 있는 성능 테스트를 개발 단계로 앞당겨 소프트웨어를 개발하는 과정에서 소프트웨어의 비기능적인 성능요소도 함께 고려하는 테스트 주도 개발에서의 성능 테스트 방안을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 테스트 주도 개발과 테스트 우선 성능 기법에 대해 기술하고, 3장에서는 모바일 응용 소프트웨어 성능 테스트 현황 및 이슈에 대해 기술하고, 4장에서는 성능 특성 및 성능 테스트 방안에 대해 기술하고, 5장에서는 결론 및 향후 과제에 대해 기술한다.

2. 관련 연구

2.1 테스트 주도 개발(TDD)

테스트 주도 개발(TDD)은 테스트를 기반으로 하는 점진적인 소프트웨어 개발 방법으로써, Kent Beck에 의해 소개되었으며, 테스트 우선 개발(Test-First Development)으로도 알려져 있다.

테스트 주도 개발 주기는 그림 1과 같이 작성해야 하는 프로그램에 대한 테스트를 먼저 작성하고 이 테스트를 통과할 수 있도록 실제 프로그램의 코드를

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음.
[2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]

작성한다. 테스트를 실행하고, 코드의 수정 과정을 걸쳐 모든 테스트를 실행해서 테스트가 성공하는 것을 확인하면, 코드의 외적 행위는 그대로 유지하면서 내부 구조를 변경하는 과감한 리펙토링(Refactoring) 과정을 통해 지속적으로 디자인을 정제하게 된다.

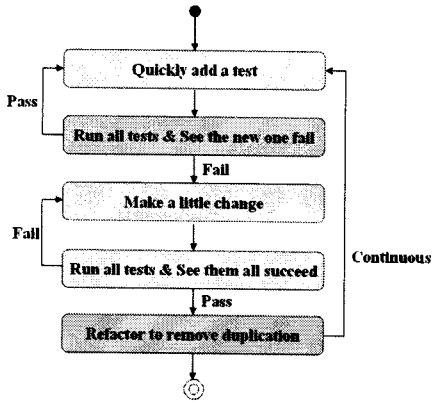

그림 1. 테스트 주도 개발 주기

이러한 개발 과정을 통해 점진적인 디자인의 성장과 함께 점진적인 코드의 완성을 이끌어내며, 최적화된 단위 테스트를 하게 된다.

테스트 주도 개발에서는 테스트를 자동화하는 것이 필수조건이다. 테스트가 자동화되어 있지 않으면 테스트를 실행하는 것이 귀찮기 때문에 테스트를 자주 실행하지 않게 되며 개발 주기에 맞춰 진행하는데 장애가 된다. 그래서 자동으로 테스트를 수행할 수 있는 환경을 갖추는 것이 중요하다. 해당 프로그램 언어를 지원하는 JUnit, cppUnit, NUnit, PyUnit과 같은 테스트 프레임워크는 테스트 구조화 및 자동화된 효율적 단위 테스트를 일반 개발 작업에 추가할 수 있는 효율적 메커니즘을 개발자에게 제공한다. 그 중 성능 테스트와 관련된 대표적인 테스트 프레임워크로 JUnitPerf[7]와 JMeter[8]가 있다.

2.2 테스트 우선 성능 기법(TFP)

테스트 우선 성능 기법(Test-First Performance)는 테스트 주도 개발 접근 방법과 성능 테스트를 병합한 기법으로 단위 테스트와 성능 테스트를 위한 JUnit 테스트 프레임워크를 사용한다[9].

TFP에서는 테스트 주도 개발 방식과는 달리 테스트 대상이 제대로 동작하는지 판단하도록 돕는 JUnit assert methods를 사용하는 대신, 성능 시나리오(performance scenarios)를 사용하며, 그것들은 성능 로그 파일을 생성하기 위해 사용된다. 또한 개발자들과 성능 설계사는 성능 문제 및 영역을 식별하기 위해 그 로그 파일들을 이용한다.

TFP에서는 성능 영역 식별, 성능 목표 수립 및 테스트 케이스를 설계하는 Test Design 단계, 성능 로그 파일을 생성하여 결함을 찾는 Critical Test Suite Execution단계, 모든 테스트 케이스를 수행하여 성능 결함을 찾는 Master Test Suite Execution 이러한 세 단계의 프로세스로 나눠서 수행된다.

3. 모바일 응용 소프트웨어 성능 테스트 현황 및 이슈

3.1 모바일 응용 소프트웨어

일반적으로 모바일 응용 소프트웨어는 모바일 단말기 상에서 동작하는 소프트웨어로써 J2ME 및 WIPI 모바일 플랫폼을 기반으로 개발된다. 이는 VM(Virtual Machine) 환경에서 단말기로 다운로드 받은 후에 실행하여 사용되며, 다음과 같은 측면에서 일반 소프트웨어와 차별화된 특징을 갖는다.

첫째, 생명주기(Life Cycle)가 매우 짧다. 모바일 응용 소프트웨어는 치열한 경쟁 상황에서 시장 적시성을 만족하기 위해 3~4개월의 짧은 개발기간에 개발된다. 충분한 설계와 테스트가 요구되는 안정성보다는 시장성을 우선시하므로 S/'W의 품질과 신뢰성이 떨어진 상태로 출시되고 있는 실정이다.

둘째, 모바일 응용 소프트웨어는 기술의 한계로 PC에서처럼 지속적인 업데이트가 불가능하기 때문에 개발 완료 후, 제품화되어 한번 탑재되면, 추가적인 업데이트가 거의 없는 단발성의 특징을 가지므로 탑재 이전에 모바일 응용 소프트웨어에 대한 엄격한 테스트가 필요하다.

이러한 특징들을 지닌 모바일 응용 소프트웨어를 개발할 때 기존의 전통적인 개발 방법보다 설계 및 구현 시간을 단축해 주며, 모바일 응용 소프트웨어의 실행 중에 수행환경의 변화에 따라 기능을 동적으로 재구성 해주는 아키텍처 기반 모바일 응용 소프트웨어 개발 환경의 개발하는 것에 목표를 두고 수행 중에 있는 MOPAD(Mobile Platform based Application Development: 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발) 프로젝트 연구에 참여하여 연구하고 있다. 그 중에서 RAD 지향 모바일 응용 소프트웨어 개발 프레임워크에서의 단위 테스트 지원 도구에 대한 연구를 진행 중에 있으며, 모바일 응용 소프트웨어 성능 테스트에 초점을 두고 있다.

3.2 모바일 응용 소프트웨어 성능 테스트 현황 및 이슈

국내 모바일 응용 소프트웨어 개발은 매우 복잡한 구도를 가지고 있다. 이동통신회사와의 계약을 통해 수많은 소프트웨어 회사들이 개발을 진행하는 것 이외에도 이동통신회사에서 내부적으로 소프트웨어

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음.
[2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]

개발을 진행하기도 하고, 다수의 하청 개발회사가 각 이동통신회사별로 개발에 참여하고 있다. 이로 인해 테스트에 대한 책임소재 및 테스트 주체가 불분명한 경우가 많고, 단말기 자체의 프로세싱 파워와 메모리, UI, 데이터 입력 등이 제한적이기 때문에 테스트가 일반적인 소프트웨어의 그것과는 다른 부분이 다수 존재하며, 실행속도 및 안정성과 관련된 제품의 품질 향상을 위해 성능요소의 테스트가 중요한 이슈로 부각되고 있지만, 성능 테스트는 아직 매우 미흡한 실정이다[10].

모바일 응용 소프트웨어 개발 일정이 촉박하여 대부분의 테스트는 GUI 및 기능과 키 이벤트 위주의 블랙박스화 된 테스트를 수행함으로써 결함 발생 시 결함 원인 식별, 발생 위치 추적 및 해결의 어려움을 겪는다. 또한 에뮬레이터(Emulator)를 사용한 모바일 응용 소프트웨어 개발 시, 에뮬레이터와 실제 타겟(Real Target)인 모바일 단말기에서의 성능 테스트 결과가 각각 서로 다르기 때문에 신뢰하지 못해 에뮬레이터에서 성능 테스트는 거의 수행하지 않고, 대부분의 성능 테스트는 개발 후반에 집중된 제품 중심 테스트로 실제 타겟에 탑재되어 수행된다. 이 경우 Fault 발견 시, 문제의 발생 원인을 발견하기 어렵기 때문에 오류를 수정하는데 많은 시간이 걸린다.

다음은 모바일 응용 소프트웨어 개발 시, 성능과 관련한 주요 이슈를 정리한 것이다.

- 메모리

휴대폰 메모리의 용량의 제약으로, 메모리를 효율적으로 사용하는 프로그램의 개발이 필요함.

- 타이밍 관리

각 폰 마다 속도편차가 나는 경우가 많아서, 균일 속도 유지가 큰 문제가 됨
화면의 갱신 시점과 데이터 처리 시점, 키 입력 시점의 Sync에 유의

- 느린 입출력 속도

데이터의 입출력 속도, 이미지의 로드 속도 등이 상당히 느림

- 잦은 상태 전환

프로그램 실행 도중, 전화 착신, 메시지 착신, 임의 타임에 종료가 가능한 점등, PC 환경에 비해 돌발 상황이 많음

- 리소스 관리

사운드 데이터, 그래픽 데이터의 효율적 처리를 위해, 파일 병합 처리 필요

따라서 이러한 이슈들을 해결하기 위해 개발 후반에 집중된 제품 중심 성능 테스트를 개발하는 과정으로 앞당겨, 모바일 응용 소프트웨어를 개발할 때 기능 테스트뿐만 아니라 성능 테스트도 함께 수행하여 성능 테스트 결과를 반영한 개발을 하는 것이 필요하다.

4. 모바일 응용 소프트웨어 성능 테스트 방안

4.1 성능 특성

모바일 응용 소프트웨어의 성능 특성은 모바일 응용 소프트웨어가 모바일 단말기를 제어하면서 작업을 수행하는데 있어 영향을 주는 성능 요소의 집합으로, 성능 테스트에서 측정해야 할 테스트 항목이 된다. 다음 표1에서는 모바일 응용 소프트웨어 성능 특성의 일부분을 제시한다.

표 1. 모바일 응용 소프트웨어 성능 특성 (일부분)

메모리 활용 (Memory Utilization)	메모리 사용 측면에서 메모리의 활용이 효율적으로 이루어지는지에 대한 테스트	
	정확성 (Accuracy)	메모리 사용 중 잘못된 할당과 접근, 보호로 인한 메모리 사용의 정확성에 대한 테스트
	충돌 (collision)	메모리 사용 중 다른 메모리 영역 및 값의 충돌 정도에 대한 테스트
	누수 (Leakage)	메모리 사용 중 메모리의 낭비로 인한 누수 정도 테스트
타이밍 성능 (Timing Performance)	시간 관련 처리가 정확하게 이루어지는지에 대한 테스트	
	지연 (Latency)	작업 요청 후 실제 작업 수행 시작까지의 대기 시간 테스트
	수행 (Execution)	실제 작업 수행 시간 테스트

4.2 성능 테스트 방안

소프트웨어의 품질 향상 및 신뢰성을 높이고, 개발 생산성을 향상시키기 위해 개발 후반에 집중된 제품 중심 성능 테스트를 개발하는 과정으로 앞당겨, 모바일 응용 소프트웨어를 개발할 때 기능 테스트뿐만 아니라 성능 테스트도 함께 수행하여 성능 테스트 결과를 반영하는 성능 테스트 방안이 필요하다. 다음 그림 2는 테스트 주도 개발 프로세스를 기반으로 체계적인 테스트 단계 및 테스트 환경을 정의하고, 성능 테스트 방안을 나타낸 것이다.

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음.
[2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]

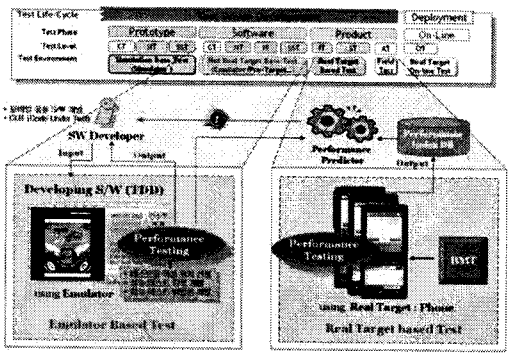

그림 2. 성능 테스트 방안

테스트 절차는 '테스트 기반 문서 검증(Test Basis) 단계, 프로토타입(Prototype) 테스트 단계, 소프트웨어(Software) 테스트 단계, 제품(Production) 테스트 단계'로 구분하고, 테스트 절차에 따른 테스트 환경은 '테스트 문서 기반 테스트(Document Based Test) 환경, 시뮬레이터 기반 테스트(Simulator Based Test) 환경, 에뮬레이터 기반 테스트(Emulator Based Test) 환경, 실제 타겟 기반 테스트(Real Target Based Test) 환경'으로 정의한다. 이러한 테스트 환경 중에서 본 논문에 초점을 두는 환경은 에뮬레이터 기반 테스트 환경이다.

그림2에서 보는 바와 같이 에뮬레이터 기반 성능 테스트는 제품 단계에 치우쳐 있는 성능 테스트 활동을 소프트웨어 테스트 단계로 미리 앞당겨 개발을 하면서 성능 테스트를 수행하고 있다. 따라서 개발자가 테스트 주도 개발 방법으로 모바일 응용 소프트웨어를 개발할 때, 측정해야 할 성능 영역을 식별하여, 테스트할 핵심 위치를 선정하고, 성능 테스트 항목 개발 및 성능 테스트 방법을 개발하여 개발자에게 가이드라인을 제공한다.

대체적으로 에뮬레이터와 실제 타겟에서 수행한 테스트 결과가 같다는 가정하에 개발을 하지만, 실질적으로 에뮬레이터에서 수행한 성능 테스트 결과와 실제 모바일 단말기에서 수행한 성능 테스트 결과가 일치하지 않는 경우가 있을 수도 있다. 이러한 경우 소프트웨어 개발자가 모바일 단말기에 탑재해서 테스트해보지 않고도 예측 가능하도록 신뢰할 수 있는 정보를 주기 위한 방안으로 그림 2에서와 같이 개발하기 이전에 성능과 관련된 BMT를 참고하여 실제 타겟 기반 테스트 환경에서 성능 테스트를 수행하고, 그 테스트 결과를 Performance Index DB로 구축하여 성능 예측기 (Performance Predictor)를 통한 성능 결과 정보를 제공받는다.

따라서 개발자는 개발할 때 활용할 수 있는 유용한 성능 정보를 제공받음으로써, 신속한 개발과 품질 향상을 도모할 수 있다.

5. 결론 및 향후 과제

테스트 주도 개발은 디버깅하는 시간을 단축시켜주고, 더 나은 코드 디자인을 제공함으로, 빠른 개발 속도 및 신뢰성 있는 소프트웨어를 개발할 수 있도록 도와준다. 이러한 테스트 주도 개발의 특성을 반영하여 개발 후반에 집중되어 있는 성능 테스트를 개발 단계로 앞당겨 소프트웨어를 개발하는 과정에서 소프트웨어의 비기능적인 성능요소도 함께 고려하는 테스트 주도 개발에서의 성능 테스트 방안은 소프트웨어의 품질 향상 및 신뢰성을 높이고, 개발 생산성을 향상시키는데 도움을 줄 것이다.

본 논문에서는 모바일 응용 소프트웨어 성능 테스트 현황과 이슈를 파악하고, 모바일 응용 소프트웨어 성능 테스트를 위해 필요한 성능 특성들을 추출하였으며, 테스트 주도 개발의 특징을 반영한 성능 테스트 방안을 제시하였다.

향후에는 추출한 성능 특성을 바탕으로 구체적인 성능 항목을 개발하고, 본 논문에서 제안한 성능 테스트 방안의 타당성을 검증하기 위해 사례 연구를 수행할 예정이다.

6. 참고문헌

[1] Kent Beck, "Test-Driven Development By Example", 2003
[2] http://www.junit.org
[3] Hamlet, R.,'Unit testing for software assurance', Computer Assurance, 1989.
[4] L.E. Williams, M. Maximilien, and M.A. Vouk, "Test-Driven Development as a Defect Reduction Practice," Proc. 14th Int'l Symp. Software Reliability Eng., IEEE CS Press, 2003, pp. 34-35.
[5] Shari Lawrence Pfleeger, "Software Engineering: Theory and Practice", 2001
[6] C.-W. Ho et al., "On Agile Performance Requirements Specification and Testing," Proc. Agile 2006 Int'l Conf., IEEE Press, 2006, pp. 47-52.
[7]http://www.clarkware.com/software/JUnitPerf.htm
[8] http://www.jmeter.org
[9] Johnson, Michael J.; Ho, Chih-Wei; Maximilien, E. Michael; Williams, Laurie, "Incorporating Performance Testing in Test-Driven Development," Software, IEEE Volume 24, Issue 3, May-June 2007, pp. 67-73.
[10] E.J. Weyuker and F.I. Vokolos, "Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study," IEEE Trans. Software Eng., vol. 26, no. 12, Dec. 2000, pp. 1147-1156.

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음.
[2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]