# Windows 드라이버 테스트 프로세스 적용 사례

최정희*, 박상현**, 이명수**, 이상근**
*고려대학교 컴퓨터정보통신대학원 소프트웨어공학과
**고려대학교 컴퓨터 · 정보통신공학과
e-mail :{ leowang, condols, lms9711, yalphy}@korea.ac.kr

# Application of Windows Driver Test Process

Jeong-Hee Choi*, Sang-Hyun Park**, Myong-Soo Lee**, Sang-Keun Lee**
*Dept. of Software Engineering, Graduate school of Computer Information Communication, Korea University
**Division of Computer and Communication Engineering, Korea University

**요 약**

소프트웨어 제품에서 Windows 드라이버의 결함은 BSOD 를 발생시키거나 제품의 전체 기능을 마비시켜 제품 신뢰도에 악영향을 미치게 된다. Windows 드라이버는 제품과 연관되는 부분이 많아 별도의 테스트가 어려워, 제품의 알파 테스트 단계에서 검증 단계를 거치게 된다. 이 단계에서는 테스트는 드라이버만의 기능 검증이 어렵고, 드라이버 결함 원인 파악 및 수정 기간의 연장을 가져온다. 이로 인한 전체 프로젝트 일정에도 영향을 주게 된다.

이에 본 연구에서는 드라이버의 신뢰성을 확보를 위한 Windows 드라이버 테스트 프로세스를 제시하였다. 전체 개발 프로세스 내에 드라이버에 관한 테스트 프로세스를 구체화하고, 드라이버 별도의 테스트 단계를 두는 방안을 제안했다. 제품 알파 단계 이전에 드라이버의 결함을 발견하여 제품 테스트 기간에 발견되는 드라이버 결함 발견이 감소되는 것을 증명하였다.

## 1. 서론

품질 비용은 크게 적합비용(Conformance cost)과 부적합비용 또는 실패비용(Nonconformance cost or failure cost)으로 구성된다[1,2,3].

적합 비용은 예방비용(prevention cost)과 평가비용(appraisal cost)으로 구성된다. 예방비용은 절차나 방법을 정립, 구성원 교육, 툴 도입, 품질에 대한 계획을 세우는데 필요한 비용이다. 평가비용은 코드 검증, 리뷰, 테스트 수행하는데 필요한 비용이다. 부적합 비용은 내부실패비용(internal failure cost)과 외부실패비용(external failure cost)으로 나눈다. 내부실패비용이란 실행한 테스트 케이스가 실패로 판명되었을 때 발생하는 비용으로, 프로그래머가 개발한 코드에서 발생한 결함을 수정하고 있다면 내부실패비용을 유발하고 있는 것이다. 외부실패비용은 테스터가 발견한 결함보다는 고객이 발견한 결함과 관련된 비용으로 프로그래머나 테스터가 발견한 결함으로 인해 발생하는 내부실패비용보다 더 크다. 외부실패비용은 내부실패비용에 기술지원을 위한 비용 및 결함 수정 후 외부로 배송하는 과정에서 발생하는 비용을 추가한 것이기 때문이다[2,3,4].

소프트웨어 제품에서 Windows 드라이버의 결함은 BSOD 를 발생시키거나 제품 전체 기능 동작에 이상을 발생시킨다[7,8]. 즉, Windows 드라이버의 외부실패비용은 다른 모듈에 비해 큰 비용을 차지한다. 기술지원 비용 및 결함 수정에 발생한 비용뿐만 아니라 회사 이미지 손상, 제품의 신뢰도 하락까지 수반하게 된다.

그러나, Windows 드라이버는 개발자의 디버깅이나 유닛 테스팅(Unit Testing)수준에서 많은 연구가 이루어지고, 전체 개발 프로세스에서 고려되고 있지 않다.

본 논문에서는 Windows 드라이버 결함의 외부실패비용을 줄이고, 제품의 신뢰성을 확보할 수 있는 하나의 방안으로 Windows 드라이버 테스트 프로세스라는 세부 개발 프로세스를 제안한다. 이 프로세스의 적용 사례에 관한 활동들과 결과를 보여준다.

본 논문의 구성은 다음과 같다. 2 장에서는 Windows 드라이버에 테스트에 관한 관련 연구를 설명하며, 3 장에서는 제안하는 기법을 설명한다. 4 장에서는 3 장의 기법을 수행한 결과의 비교 평가, 5 장에서는 본 논문의 결론을 맺는다.

## 2. 관련 연구

메인 OS 개발자들은 결함이 있는 드라이버가 시장에 배포되는 위험을 줄이기 위해 정기적으로 향상된 사양과 드라이버 개발자가 사용할 툴을 배포한다. 마이크로소프트는 SLAM[9]과 DDK(Driver Development Kit)[10]와 같은 드라이버 개발 패키지를 포함하는 많은 테스팅 도구들을 제공한다. 그러나, 향상된 기능에도 불구하고, 결함이 있는 드라이버가 여전히 시스템 실패(failures)의 주요한 원인이 되고 있다[11].

Windows 드라이버는 세가지 수준의 복잡성이 있다. 첫 번째는 하드웨어와 통신하는 것이고, 두 번째는 플러그 앤 플레이와 전원, 운영체제의 요구사항을 구

현하는 것이다. 세 번째는 다른 드라이버와 통신하는 것이다[6]. Windows 드라이버 개발에서는 다른 일반적인 어플리케이션 개발에 비하여 디버깅과 테스팅이 강조된다. 마이크로소프트에서 드라이버 개발자에게 제안하는 테스팅 도구에는 다음과 같은 것이 있다.

• Driver Prefast[6]: 드라이버 개발 시 사용되는 정적 소스 분석 툴이다. 컴파일 시 수행돼 컴파일러와 런타임 테스트로 발견되지 않는 코딩 에러를 보고한다.

• SDV(Static Driver Verifier)[6,9]: 정적 검증 툴로서 운영체제에 드라이버 내부 호출 경로 에뮬레이션과 심볼 수준에서 소스 코드를 실행한다. SDV 는 윈도우 내부 정보와 드라이버가 윈도우 인터페이스를 사용해야 하는 방법을 내장하고 있다.

• 어플리케이션 베리파이어(AppVerifer)[6]: 비관리화 코드(unmanaged code)로 작성된 유저 모드 어플리케이션 용도의 동적 검증 툴이다. 유저모드 드라이버나 커널 모드 드라이버를 수반하는 유저 모드 어플리케이션 에러를 찾는데 사용할 수 있다. 어플리케이션 베리파이어는 표준 애플리케이션 테스트나 표준 드라이버 테스트 동안 발견하기 어려운 미묘한 프로그래밍 에러를 찾아낸다. 어플리케이션이 실행중인 동안 애플리케이션을 감시하면 애플리케이션에 다양한 스트레스를 주면서 테스트하고 애플리케이션이 실행이나 설계에 존재하는 잠재적인 에러 보고서를 생성한다.

## 3. 제안하는 방법

Windows 드라이버 테스트 프로세스를 적용하기 위한 세부 방안으로 1 단계 개발자의 유닛 테스팅 프로세스 구체화, 2 단계 QAE 가 드라이버 모듈 테스팅 수행을 기본적인 틀로 추진하였다. 이와 같은 프로세스를 적용하기 위해서는 Windows 드라이버 개발 프로젝트가 필요하였고, 새로 개발되거나 메이저 버전 업이 이루어지는 드라이버 개발 프로젝트들 중에 하나로 선정하였다.

선정된 드라이버는 커널 단에서 필터 드라이버를 관리하고, 유저 어플리케이션과 메시지 통신을 담당하는 기능을 수행한다. 기존 1.0 에서 2.0 으로 업그레이드하는 프로젝트이다. 해당 타겟(Target) 드라이버를 A 라고 칭하도록 한다.

### 3.1 유닛 테스팅 프로세스 구체화

Windows 드라이버 분야별 대표 개발자들과 각각 개별 면담 및 전체 회의를 통하여 개발 환경 및 테스팅 수준을 파악하였다.

테스팅 방법에 대한 공통적인 규칙(Rule)은 존재하지 않았으며, Code Static Analysis 는 팀 내의 규약으로 정하여 수행하며 SDV, PREfast 를 테스팅 도구로 사용하며, 소스 커버리지 80%정도의 목표치 달성하도록 하고 있었다. 개발팀 내의 내부적인 규칙은 가지고 있으나, 강제성이 없어 다른 제약 조건이 있을 경우는 지켜지지 못하고 있었다.

이와 같은 현황을 토대로 드라이버 개발팀내의 유닛 테스팅 프로세스 구체화를 1 단계 방법으로 선정하였다. 소프트웨어 개발 프로세스의 초기 단계에서 산출물에 대한 결함을 효과적으로 발견함으로써 개발 과정 나중의 단계까지 남아서 많은 비용과 결함을 발생키는 잠재적인 결함을 감소시킬 수 있기 때문이다[5].

개발자는 자신의 코드의 검증을 위해 개발자 테스트 단계를 거치고 있지만, 구체적인 테스트항목 및 방법이 개발 프로세스 내에 규정되어 있지 않았다. 개발자 유닛 테스팅을 강화하기 위해 정적 분석과 동적 분석을 수행을 하고, 코드 커버리지 측정 도구를 통하여 내부적인 정한 목표율을 달성하도록 하게 하였다. 코드 커버리지의 목표율은 기존의 규칙과 같이 80%로 협의 하였고, 이 목표율을 달성하지 못할 시에는 타당한 이유를 별도로 기술하도록 하였다. (그림 1) 과 같은 내용의 개발자 유닛 테스팅 체크 리스트 문서를 만들고, 팀 내의 프로세스가 아닌 Windows 드라이버 개발 프로세스로 사내 전체 프로세스 중의 하나로 강제사항이 되도록 하였다.

**개발자 유닛 테스팅 체크 리스트**

1. 개요

| 프로젝트 명 | | 담당자 | |
|---|---|---|---|
| 적용제품 | | 작성일 | |

2. 모듈 체크 리스트

추가되거나, 변경된 모듈 리스트를 작성하고, 점검 항목에 대한 수행 여부를 표기한다. 단, Code Coverage 항목에는 달성한 퍼센트를 기입한다. Code Coverage 기준은 80% 이상으로 하고 달성하지 못한 경우는 이유를 비고에 기술하도록 한다.

| 모듈명 | 기능 | Static Analysis | Dynamic Analysis | Code Coverage | 비고 |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |

3.OS 커버리지

테스트된 OS, 추후 테스트가 필요한 OS, 테스트가 필요치 않는 OS 를 기입한다.

4. 추가 테스트 리스트

기본적인 모듈 점검 이외에 추가적으로 수행한 테스트 내용을 기입한다.

5. 예상되는 리스크

예상되는 리스크나 개발자 테스트에서 확인이 불가능하였던 사항을 기입한다. QAE 테스트 수행 시 고려될 수 있도록 한다.

6. 테스트 툴

QAE에 전달할 테스트 툴이 별도로 개발된 경우 툴에 대한 설명을 기입한다. 테스트 툴 명, 위치, 사용법, 특징 등을 설명한다. 별도의 문서를 작성하여 첨부할 수 있다.

(그림 1) 개발자 Unit Testing 체크 리스트

### 3.2 드라이버 모듈 테스트 수행

Windows 드라이버 제품과 연관되는 부분이 많이 별도의 테스트가 어려워, 제품의 알파 테스트 단계에서 검증이 이루어졌다. 이 단계에서 드라이버만의 기능 검증이 어렵고, 결함이 발견되면 원인 파악 및 수정기간의 연장을 가져왔다. 제품 알파 테스트 단계 이전에 해당 드라이버의 기능만을 별도로 테스트 할 수 있도록 QAE 가 드라이버가 모듈 테스트를 수행하도록 하였다.

테스트 방법으로 크게 2 가지를 나누었다. 첫 번째는 기존의 발생했던 버그 리스트를 확인하여 재 검증 단계로 삼고, 두 번째는 타겟 드라이버만의 기능 검증 단계를 거치도록 하였다.

첫 번째는 기존에 발생했던 버그 리스트 확인을 위해서 이슈트래킹 시스템에 등록된 내용을 분석하고 테스트 케이스를 추출하여 회귀 테스트(Regression Test)를 수행하도록 하고, 두 번째는 타겟 드라이버의 SRS(Software Requirement Specification)을 분석하여 기능 테스트(Function Test)가 수행될 수 있도록 하였다.

드라이버 A 의 만의 테스팅을 위해서는 개발자에게 별도의 테스팅 툴을 요구해야 할 필요가 있었다. 타겟 드라이버 A 의 개발자에게 개발 리소스(Resource)뿐만 아니라, 프로젝트 초반에 테스팅 툴 도구 개발을 위한 별도의 리소스를 미리 확보하였다. 개발자와 협의하여 구현기간 대비 1/10 정도의 리소스를 확보하였다.

회귀 테스트 케이스 추출은 기존 이슈트래킹 시스템에 올려진 버그 중 Windows 드라이버 개발자가 처리한 총 256 건을 선정하여 이슈를 분류하였다.
기존에 등록된 이슈가 너무 방대하고, 본 연구와 관련 있는 이슈인지 여부를 구분하는데 많은 시간이 소요되었다. 본 연구와 가장 관련이 있는 제품 테스트를 하였던 QAE 에게 요청하여 기존 프로젝트에서 최대한의 필터링을 통하여 해당 이슈를 선정하였다. <표 1>은 이슈의 최초 현상으로 분류한 결과이며 <표 2>는 이슈의 처리 결과로 분류한 내용이다.

**<표 1> 이슈 현상의 분류**

| 순번 | 이슈 현상분류 | 개수 |
|---|---|---|
| 1 | 공유이상 | 2 |
| 2 | 기능오작동 | 40 |
| 3 | 기능 Off | 32 |
| 4 | 네트워크느림 | 3 |
| 5 | 부팅불가 | 11 |
| 6 | 설치오류 | 6 |
| 7 | 시스템느려짐 | 2 |
| 8 | 시스템오류 | 1 |
| 9 | 시스템정지 | 8 |
| 10 | 시스템종료불가 | 3 |
| 11 | 요청 | 10 |
| 12 | 자동재부팅 | 19 |
| 13 | 타프로그램이상/정지 | 9 |
| 14 | 파일속성변경 | 1 |
| 15 | 프린팅불가 | 2 |
| 16 | BSoD | 67 |
| 17 | ErrorMessage | 12 |
| 18 | Internet 연결불가 | 14 |
| 19 | Mail 불가 | 7 |
| 20 | MS 보안경고 | 7 |
| | 총 | 256 |

공유 이상의 경우는 제품 제거 후에 공유 폴더로의 접근이 불가능하거나, 공유 폴더에서의 복사가 3 배 이상 느려진 경우가 발생하였다. 가장 많은 현상인 BSOD 는 제품 설치 중이나 설치가 완료되고 BSOD 가 발생하였다. 설치 후에 시스템이 부팅이 되지 않은 경우, 프린팅이 되지 않는 경우, 자동으로 재부팅이 되는 경우 등 Windows 드라이버의 결함은 제품이 아닌 시스템 자체의 사용이 불가한 현상이 많았다.

**<표 2> 이슈의 처리 결과 분류**

| 순번 | 이슈의 처리 결과 | 개수 |
|---|---|---|
| **1** | **기능수정** | **4** |
| **2** | **버그** | **84** |
| 3 | 설명부족 | 1 |
| **4** | **악성코드** | **6** |
| 5 | 엔진 | 5 |
| **6** | **원인파악불가** | **49** |
| **7** | **원인 파악 후 종료** | **4** |
| 8 | 재설치 해결 | 6 |
| 9 | 정보공유 | 2 |
| 10 | 정상 | 5 |
| **11** | **타사 제품과 충돌** | **50** |
| 12 | Known Bug | 4 |
| 13 | MS 버그 | 4 |
| 14 | PC 환경 | 26 |
| 15 | Vista 인증 | 6 |

버그 이외에는 원인파악불가로 처리되거나 타사 제품과의 충돌의 가장 많은 처리 결과가 나왔다. 개발자가 원인을 파악하기 전에 사용자가 PC 를 포맷하였거나 추 후에는 다시 현상이 발생하지 않은 경우가 있었으며, 비슷한 기능을 하는 드라이버 제품과 서로 충돌되어 발생한 경우가 많았다.

기존 이슈의 처리 결과의 분류 중 설명부족, 엔진, 재설치 해결, 정보 공유, 정상, Known Bug, MS 버그, PC 환경, Vista 인증을 제외한 나머지(197 개/전체 256 개)를 가지고 테스트 케이스 추출을 위한 버그를 분류하였다. 이슈의 처리 결과에 따른 내용 상세 분류의 내용을 통하여 드라이버 A 기능과 관련이 있는 9 가지의 테스트케이스를 추출하였다.

SRS 분석을 통해 드라이버 A 만의 기능 테스트 케이스 16 가지를 추출하였다. 테스트 툴 개발 단계 전에 테스트 수행을 위한 테스트 툴 요구사항을 도출하고 드라이버 A 모듈 테스트 위한 테스트 툴을 인계받았다. 테스트 툴에 한계가 있는 경우는 제품으로 테스트를 수행하기로 하였다.

## 4. 비교 평가

테스트 수행환경은 기존 제품 QAE 와 개발자와 협의를 통해 <표 3>과 같이 선정하였다.

**<표 3> 테스트 수행 환경**

| Arch | OS | CPU | Mem | HDD |
|---|---|---|---|---|
| X86 | Windows 98 SE | P4 1.80Ghz | 256MB | 20GB |
| X86 | Windows XP Professional | P42.4Ghz | 756MB | 40GB |
| X86 | Windows Vista | Core2 1.86Ghz | 2GB | 250GB |
| X64 | Windows Vista | | | |

회귀 테스트 중 다른 제품과의 충돌 프로그램과의 테스트는 Windows XP Professional, Windows Vista 에서 수행하였다. Not Test(N/T)는 테스트 툴도 지원불가하며,

제품으로 테스트 할 수 없는 경우이다.

<표 4> 테스트 수행 결과표

| 구분 | 테스트 환경 | TC 수 | 테스트 결과 | | |
|---|---|---|---|---|---|
| | | | PASS | FAIL | N/T |
| 회귀 테스트 | Windows 98 SE | 9 | 6 | 1 | 2 |
| | Windows XP Pro | 9 | 8 | 1 | 0 |
| | Windows Vista | 9 | 6 | 2 | 1 |
| | Windows Vista x64 | 9 | 5 | 1 | 2 |
| 기능 테스트 | Windows 98 SE | 16 | 14 | 0 | 2 |
| | Windows XP Pro | 16 | 11 | 3 | 2 |
| | Windows Vista | 16 | 11 | 3 | 2 |
| | Windows Vista x64 | 16 | 11 | 3 | 2 |
| 총 | | 100 | 72 | 15 | 13 |

<표 4>는 테스트 수행 결과이다. 드라이버 A 가 새로운 개발이 아닌 버전 업그레이드 프로젝트로 많은 오류가 발견되지는 않았다. 타겟 드라이버 A 에 대한 검증뿐만 아니라 다른 모듈의 기능에서 오류가 발견되었다. 또한, 툴에 대한 명확한 요구사항이 도출이 되지 않아 Not Test 건수의 비중이 높았다.

현재 드라이버 A 를 적용한 새로운 제품이 출시되었다. 알파 단계부터 RC 단계까지 테스트한 결과 타겟 드라이버 A 에 의한 오류는 발견되지 않았다. 타겟 A 를 포함하는 드라이버 A'의 이슈트래킹의 이슈등록 건수를 비교하여 보았다. 이슈트래킹 시스템의 분류가 A'로 등록되어 전체 이슈와 A'와 이슈 등록건수로 비교하였다.

<표 5> 결과 비교표

| 제품 | 드라이버 A 버전 | 제품 Release 이전 | | | 제품 Release 이후 | | |
|---|---|---|---|---|---|---|---|
| | | 전체 이슈 등록건수 | A' 이슈 등록건수 | A' 이슈 비중 (%) | 전체 이슈 등록건수 | A' 이슈 등록건수 | A' 이슈 비중 (%) |
| 가 | 1.0 | 4,444 | 205 | 4.6 | 5,055 | 703 | 13.9 |
| 나 | 2.0 | 791 | 9 | 1.1 | 10 | - | 0 |

<표 5>의 드라이버 A 버전 2.0 은 본 연구에서 적용한 Windows 드라이버 개발 프로세스를 적용한 버전이고, 1.0 은 이전과 같은 방법으로 별도의 Windows 개발 프로세스를 적용하지 않은 버전이다.

A 드라이버를 탑재한 '가'는 클라이언트 제품 '나'는 서버 제품이라는 환경에서 차이가 있지만 Windows 드라이버 기능상으로 큰 차이가 없다. '가'제품은 2006 년 5 월 릴리즈 후 많은 패치 및 버전 업그레이드 된 제품으로 등록 이슈가 많고, 클라이언트 제품이므로 서버 제품에 비하여 외부 등록 건수가 많다. 제품 '나'는 2008 년 7 월 릴리즈 후로 등록된 이슈는 현재 없다. 제품 '나'의 릴리즈 이후에 A 드라이버에 오류가 발견되는지에 대한 결과는 추후 추적이 더 필요하다. 하지만, <표 5>의 결과로 보면 기존에 비하여 드라이버에 관한 이슈가 제품 테스트 단계에서 발견된 비중이 1/4 로 줄어듦을 확인할 수 있었다.

<표 4>와 <표 5>의 결과를 바탕으로 포스트모템(postmortem)을 진행하였다. Windows 드라이버 개발자, 제품 PM 및 프로세스 담당자를 대상으로 테스트 결과를 설명하고, 새로운 드라이버 테스트 프로세스를 제안하였다.

기존의 개발 프로세스 내에 Windows 드라이버 개발팀내의 유닛 테스팅 프로세스 및 QAE 드라이버 모듈 테스트 수행이라는 세부 항목을 추가하고, 새로운 드라이버가 개발되거나 모듈이 크게 변경되는 경우는 새로운 드라이버 테스트 프로세스를 따르도록 하여 프로세스 내재화가 이루어질 수 있도록 하였다.

향후에는 드라이버에 관련한 공통적인 테스트 케이스를 더 추가하여 드라이버의 기본 기능 검증 단계로 삼고, 타겟 드라이버에 대한 면밀한 기능 검증으로 드라이버 자체의 품질 향상을 도모할 수 있을 것으로 판단된다.

## 5. 결론

본 연구에서는 Windows 드라이버 테스트 프로세스를 제안하였으며 이 프로세스 도입 사례를 통한 활동과 프로젝트 결과를 보여주고 있다.

이 프로세스의 효과성이 검증되기 위해서는 소프트웨어 개발과 유지보수 전체 단계에서의 프로세스 관리가 필요하기 때문에 높은 수준의 프로세스 내재화가 이루어질 경우, 광범위한 데이터를 기반으로 효과성의 분석이 이루어 질 수 있을 것이다.

개발이나 테스팅 단계에서의 더 많은 리소스 확보가 필요하기 때문에 개발팀과 프로젝트 관리자 층의 합의를 이끌어 내는 것이 중요할 것이다. 향후 더 많은 도입 사례와 정량적인 효과성에 대한 연구를 통해 Windows 드라이버 모듈의 품질 향상의 기반이 구축되어야 할 것이다.

## 참고문헌


[1] J. Campanella, Principles of Quality Costs. ASQ Quality Press, Milwaukee 1999.

[2] P. Crosby, Quality is Free. New York: McGraw-Hill.

[3] Quality Assurance Institute, Guide to the CSTE common body of knowledge 2006.

[4] R. Black, Managing the Testing Process Second Edition, Welly, New York, 2002.

[5] T. Gilb and D. Graham, Software Inspection, Addison-Wesley Reading Mass, 1993.

[6] P. Orwick, G. smith, C. Buchmiller and A. Pearson, Developing drivers with the windows driver foundation, Microsoft Press 2007.

[7] Michael M. Swift, Brian N. Bershad, Henry M. Levy, The reliability of commodity operating systems, ACM SIGOPS, 2003.

[8] A. Ganapathi et al., Windows xp kernel crash analysis, In Proc. LISA, 2006.

[9] T. Ball et al., Salm and static driver verifier: Technology transfer of formal methods inside Microsoft, In IFM, 2004.

[10] W. Oney, Programming the MS Windows Driver Model, Microsoft Press, 2003.

[11] C. Sarbu, A. Johansson, N. Suri, Runtime Behavior-based Profiling of OS Drivers, lecture notes in computer science, 2008.