2013. 12. 16. [제76호]

신속한 SW테스트 커버리지 결정을 위한 5단계 가이드

소프트웨어공학센터 경영지원TF팀

Contents

정보통신산업진흥원 부설 nipa 소프트웨어공학센터

Key Message

SW개발 후 테스트를 충분히 수행할 시간여유는 없는 것이 일반적이라고 볼 수 있으며, 이러한 상황 하에서 신속하게 SW테스트 커버리지를 결정하는 것은 테스트 수행시간을 확보할 수 있는 방법 중 하나임. 신속한 SW테스트 커버리지 결정을 위해 FIBLOTS 모델을 활용하고 테스트 범위와 특성을 결정하고 작업리스트를 만드는 등의 방안을 단계별로 제시함.

▶ 서론

- SW프로젝트 결과물에 대한 신속한 테스트 수행을 결정하는 때에는 다음의 두 가지 질문으로부터 시작해봄
  - 첫 번째는 범위의 질문으로 "테스트가능한 모든 것 외에 어떤 특징들이 테스트하기에 적합한 것인가?"임. 테스트 현장에서는 항상 예정된 테스트 범위 이상으로 테스트 할 것들이 있음
  - 두 번째는 테스트 기법과 커버리지의 질문으로 "테스트가 예정된 각각의 기능들을 어떠한 방법을 통해 원하는 결과를 도출할 수 있을 것인가?"임. 기능별 상이한 품질기준들은 각기 다른 기능요소들과 각기 다른 테스트 기법들을 활용하게 만듦
- 아래의 테스트 커버리지 결정을 위한 신속한 사고과정을 통해 앞서 살펴본 질문들의 대답과 함께 적시에 테스트를 실행하기 위해 테스트 실행을 조직하는 것에 대해 살펴보기로 함

▶ STEP 1 : 테스트의 범위를 계산하기

- 어떤 기능들을 테스트할 것인지의 범위에 관한 질문에 Scott Barber의 FIBLOTS의 연상을 사용하는데, 이 방법은 SW 시스템 성능 테스트에 주로 활용됨
- 테스트 범위에 대해 고려할 때 FIBLOTS를 적용하는 방식은 다음과 같음
  - Frequent(빈도) : 어떤 특징들이 가장 자주 사용되는가 (예. 사용자 상호작용 요소나 백그라운드프로세스)
  - Intensive(집중도) : 어떤 요소들에 가장 집중되는가?(검색, 대용량 데이터 처리, 강한 GUI 상호작용)

- Business-critical(비즈니스관점) : 어떤 요소들이 일하는데 필요한 프로세스를 지원하는가(예. 월말작업, 새 계좌 생성)
- Legal(제도적요소) : 어떤 요소들이 제도적 측면에서 요구되는 프로세스를 지원하는가(예. 계약프로세스 등)
- Obivious(명백함) : 어떤 요소들이 기능 수행에 실패했을 때 치명적인 상황을 초래하는 프로세스를 지원하는가
- Technically risky(기술적 리스크) : 어떤 요소들이 시스템의 기술적인 리스크 측면을 지원하거나 상호작용 하는가
- Stakeholder-mandated(이해관계자 위임) : 테스트 결과를 확신하기 위해 고객 등 이해관계자들이 질문하고 답변한 것들은 무엇인가

▶ STEP 2 : 테스트하는 각 요소들을 자세히 이해하기

- 무엇을 테스트할 것인지에 대해 결정하고 나면, 테스트에 포괄하고 싶은 각 요소들의 측면에 대해 이해하는 것이 필요함
- 제품 요소 리스트를 활용하여 테스트 수행에 있어 집중해야할 요소들을 결정하고 상위수준에서 다음 부분들에 대한 커버리지를 결정함
  - Structure(구조) : 물리적 제품이나 코드나 하드웨어 등 검토할 상세 특징들로 구성된 것
  - Functions(기능) : 제품이나 UI, 계산, 에러처리 등의 요소
  - Data(데이터) : 입력, 출력, 생애주기 등의 제품이나 프로세스 요소
  - Platform(플랫폼) : 프로젝트 외적으로 제품이나 특징들을 지원하는 요소
  - Operations(운영) : 공동사용, 낮은 사용, 과도한 사용 등 제품이나 특징들이 사용되는 것
  - Time(시간) : 동시실행(동시접속), 대기시간 등 실행과 시간사이의 관계

▶ STEP 3 : 테스트 과업을 합리적인 방법으로 구조화하기

- 일반적으로 테스트 과업목록을 리스트나 스프레드시트에 구조화하기 시작함
  - 무엇을 테스트할 것인가를 이해하고 결정하게 되면 어떻게 테스트할 것인지를

생각하기 시작함

- 실행할 테스팅 항목과 범위 등을 시각화 할 수 있을 때까지는 실제로 테스팅이 시작된 것이 아님에 따라 다음의 항목 등을 예시로 목록화 하는 것이 필요함
  - 런타임 분석 툴과 같은 도움이 될 전문화된 소프트웨어가 필요한가?
  - 코드를 작성하거나 네트워크 실패와 같은 어떤 활동을 조정해야하는가?
- 각 테스트 항목들을 정리하고 난후 테스트 항목들을 그룹화하여 명세서를 만들기 시작함
  - 명세서를 작성하고 수행항목을 정리하고 난후 그것들을 실행하기 위해 완료되어야 할 작업이나 장해요소들을 확인하기 시작함
  - 어떤 항목들은 다른 팀과의 협업이 요구되는 등 명세화되기 전에는 발견할 수 없는 요소들이 곳곳에 숨어있기 때문에 매우 중요한 프로세스임
- 일반적으로 다음의 두 조건이 만족되면 테스트를 준비하기 시작함
  - 1. 테스팅을 실행하기 위해 적정 수준의 테스팅 대상 소프트웨어가 있음
  - 2. 테스팅 실행을 위한 항목이 정의되어 있음

▶ STEP 4 : 테스트 대상 소프트웨어를 빠른 시간 내 검토해볼 것

- 테스팅 초기에 많은 기준을 가지지 않음에 따라 테스트 대상 소프트웨어를 가능한 신속하게 획득하여 검토하는 것이 필요함
  - 테스트하려는 대상SW 검토해보면 테스트 아이디어는 종종 바뀌기도 하며, 일차적인 검토를 끝나자마자 개발자에게 피드백을 주거나 계획된 테스트 항목을 재구성하기 시작할 수도 있음
  - 물론 일반적으로 성능테스트를 시작하기 전에 어느 수준까지는 기능적으로 안정되어있음에 따라 이런 원칙을 모든 테스트에 적용하지는 않음
- 이러한 원칙은 개발팀에게 도움이 되는 자산이 되기도 하는 데 개발팀이나 품질관리팀 모두 언제나 버그가 없는 코드(잘 짜여지고, 단위 테스트가 실행되고, 동료검토가 완료된)를 원하지만 현실을 그렇지 않음을 알 수 있음
  - 때때로 1차 검토에서 제공된 피드백이 팀에게 가치가 있음

▶ STEP 5 : 기능요소에서부터 테스트를 시작할 것

- SW를 테스트 할 때 무엇부터 시작할 것인가는 중요한 결정사항임
  - 일반적으로 시스템의 각 부분들이 어느 수준까지(기본기능이 확정되고 상대적으로 안정된 수준) 제대로 작동하는 지에 대한 확신이 생기기전까지는 엔드투엔드 테스팅(시스템이나 서브시스템의 다양한 부분에 흘러가는 데이터로 하는 테스팅) 같은 것을 시작하지 않음
- 일반적으로 적어도 하나의 "안정된" 인터페이스를 얻기 전까지는 자동화나 성능테스트를 시도하지 않는 것이 바람직함
  - 인터페이스는 웹서비스, 사용자 인터페이스나 메소드 호출 같은 것이 될 수도 있음
  - 이것이 적어도 한 두 번의 기초 테스트를 거쳤거나 개발팀으로부터 빠른 시일 내에 인터페이스에 주요 변화 계획이 없다는 것을 확인내지는 동의를 받는 것이 필요할 수도 있음
- 마지막으로 명세서 항목에 따른 테스트 실행이 한차례 완료되기 전까지는 회귀분석 테스트를 시작하지 않음
  - 회귀 테스트(regression test)는 마지막에 해야 할 작업으로 간주됨

참고 자료

1. http://testingthoughts.com/blog/59
2. http://www.softwaretestingmentor.com/test-design-techniques/decision-coverage/