동향 브리핑

소프트웨어공학센터 경영지원TF팀

탐색적 테스팅 vs. 스크립트 테스팅 : 선택적인가, 병행해야 하는 것인가?

2014. 10. 14. [제109호]

Contents

정보통신산업진흥원 부설
nipa 소프트웨어공학센터

Key Message

탐색적 테스팅(Exploratory Testing)은 테스트 스크립트 또는 테스트 케이스를 문서로 작성하지 않고 경험에 바탕을 두고 탐색적으로 기능을 수행해 보면서 테스트하는 기법이며 이와 반대되는 개념은 스크립트 테스팅(scripted testing) 임. 본 원고에서는 효율적이며 효과적인 테스팅 수행을 위해 이 두 테스팅 기법의 차이를 비교하고 균형적인 실행 방안을 제시함.

▶ 탐색적 테스팅(Exploratory Testing)은 무엇인가?

- 우리가 살고 있는 기술적인 세계에서 정밀한 측정과 정확한 사고방식에 대한 필요성이 빠른 속도로 증가하고 있음
  - "만약 측정할 수 없다면, 관리할 수 없다"[1]는 오래된 속담은 분석과 대시보드가 잘 활용할 때에 새로운 의미를 지님
  - 소프트웨어 테스트 측면에서 탐색적 테스팅의 구조와 프로세스에 직관이나 본능이 어느정도 필요하기 때문에 개발자들은 불편함을 호소하기도 함
- 탐색적 테스팅(Exploratory Testing)이란 용어는 켐 카너(Cem Kaner)가 처음 만들어 사용했음
  - 그는 탐색적 테스팅을 "프로젝트 동안에 병행으로 진행되는 상호 보완 활동으로 테스트 관련 학습, 테스트 디자인, 테스트 실행과 테스트결과 해석을 처리함으로써 작업의 품질을 지속적으로 최적화시키는 개별 테스터들의 개인적 자유와 책임을 강조하는 소프트웨어 테스팅 스타일"이라고 정의했음[2]
  - 쉽게 말해, 탐색적 테스팅은 테스터들이 즉석에서 배우고, 테스트를 적절하게 조정하고, 그 과정에서 종종 기대하지 않은 소프트웨어 문제를 발견하는 것임
  - 최종 결론은 사용자의 경험과 그것을 만든 개발자의 경험을 더 잘 반영하는 소프트웨어의 진화하는 모습임
- 탐색적 테스팅의 장점은 좀 더 전통적인 방식인 스크립트 테스팅에 비해 빠르게 훑어볼 수 있다는 것임(quick look)

1) "If it can't be measured, it can't be managed"
2) http://kaner.com/?p=46

▶ 탐색적 테스팅은 스크립트 테스팅(Scripted Testing)과 어떻게 다른가?

- 개발 현장에서 탐색적 테스팅의 적용이 어려운 가장 큰 이유는 스크립트로 실행되지 않는 형태이기 때문임
- 스크립트 테스팅은 테스터 자신이나 다른 테스터가 작성한 경로를 따름
  - 이 스크립트는 문서화된 테스트 케이스이지만 테스트 단계를 포함하며, 스크립트에 배치한 경로에는 편차가 없을 수 있음
  - 스크립트 환경에서 테스터의 역할은 단지 줄에 색을 칠하거나, 각 지시를 n번 따르고, 발견한 것을 보고하는 것임
- 반면, 탐색적 테스팅은 '끊임없이(on the go)' 테스트 경로를 개발하는 테스터에 의존하게 됨
  - 이것이 정확하게 '모든 것으로부터 자유로운(free-for-all)' 테스팅은 아니지만 테스터가 스크립트를 답습하는 대신 최종사용자의 움직임 가정하고 그것을 따르도록 함
- 이 두 방법론 모두 단점은 분명히 존재함
- 스크립트 테스팅은 테스터가 간단하게 원하는 결과를 도출할 수 있는데 이것은 테스팅을 가르치는 선생님에 비유됨
  - 학생들이 무엇을 배울지 고민하기 보다는 원하는 결과를 만들게 하도록 고민함
  - 이러한 접근방식은 테스팅 환경에서 학습을 심각하게 제한할 수 있음
  - 프로그램의 실제 사용자(최종사용자)가 당면하게 되는 많은 잠재적인 문제점은 테스터가 테스트에서 말한 범위를 반드시 벗어나게 되어 있음
  - 결과적으로 버그가 포함된 릴리즈 때문에 최종사용자가 실망하여 돌아설 수 있음
- 반면 탐색적 테스팅은 소프트웨어 설계자와 개발자에게 그들의 제품이 '야생에서(in the wild, 실제 사용환경에서)'에서 어떻게 사용되는지를 볼 수 있는 기회를 줌
  - 제약이 없기 때문에, 탐색적 테스팅은 프로그램을 처음 만들었던 사람들이 전혀 고려하지 못한 영역을 다룰 수 있음
  - 탐색적 테스팅은 프로젝트 소유자가 새로운 시각에서 그들의 작업을 보고, 스크립트 테스트가 완전히 놓쳤던 통찰력을 가지는데 도움이 줌
- 그렇다면, '탐색적 테스팅이 완벽하고 모든 소프트웨어 제조업체들의 바람에 대한 해답이 될 수 있는가'에 대한 대답은 무엇인가?
  - 전혀 아니라는 것임. 사실 탐색적 테스팅이 프로그램의 예상치 못한 단점을 전혀

발견하지 못하는 동안에 테스터가 단순히 해본 적이 없어서 간단한 문제조차 놓칠 수 있음

- 탐색적 환경에서의 문서화는 어떤 경우에는 부족하다는 필연적 결과를 불러옴

- 이에 대한 적절한 해법은 소프트웨어 일부에 스크립트 및 탐색적 테스팅의 조합을 적용하는 것임
  - 이것이 양쪽 진영에 최적이어서, 설계팀은 가능한 다양한 시각을 고려하여 최대한 견고한 품질의 릴리즈를 만들 수 있음
  - 그러나 대부분의 기업들은 둘을 균형 있게 조합할 만큼 여유롭지 못하며, 시간과 자본의 제약 때문에 하나를 선택하는 결정을 해야 함
- 이를 위해, 탐색적 테스팅을 효과적인 테스팅 방법론으로 만드는 자유도는 유지하면서 구조 수준을 제공하는데 도움이 되는 몇 가지 방법들을 알아보고자 함

▶ 유연성을 유지하면서 탐색에 집중하는 테스팅 계획은 어떻게 만들 수 있는가?

- 스크립트 테스팅과 탐색적 테스팅 간의 균형을 잡는 것은 쉽지 않기 때문에 무결성을 유지하기 위해서는 절충이 필요함
- 탐색적 테스터가 주의가 필요해 보이는 영역을 수행할 수 있는 기회를 여전히 열어놓고 테스터들이 집중을 유지하는데 도움이 되는 세 가지 옵션을 간략히 살펴보도록 함

1. 테스트 규정(Test Charters)

- 규정은 탐색적 테스팅 모델에 모양과 형식을 주는 중요한 요소로 단순히 테스트 세션의 목표를 배치함
- 유동적이어서 변할 수 있고, 제거될 수 있고, 테스팅이 진행되면서 추가될 수 있음
- 일부 탐색적 테스터들은 어떤 형식이 그들의 테스트에 적용될 때 어떤 제약을 느낄 수 있음
- 테스터 규정은 테스터들에게 어디로 가야할지를 제한하지 않고 방향을 제시함
- 규정은 테스트가 특정 결과를 도출할 것을 강요하지 않고 기대를 정의함
- 다소 역설적이지만 테스터와 그들의 고객이 프로젝트가 진행되면서 버그와 문제를 발견하는데 도움이 되는 철저한 테스트가 완료되었다고 확신하는데 도움이 되는 필수 단계임

2. 세션 기반 테스트(Session-based Testing)

- 2000년에 Jonathan과 James Bach는 세션기반 테스팅의 관리기법을 개발하였음[3)]
  - "소프트웨어 테스트 방법론은 신속한 결함 발견, 창조적인 즉석 테스트 설계, 관리 제어와 측정보고를 제공하는 책임과 탐색적 테스팅을 결합하는 것을 목표로 한다"고 정의 됨
- 이 방법론은 시나리오 테스팅과 함께 사용될 수 있음
  - 단순히 스크립트된 단계를 따르는 스크립트 테스터에게, 세션 기반 테스트는 본질적으로 어느 종류의 작업이 완료가 되고 어떻게 보고할 것인지를 보증하는 정도의 구조를 제공함
  - 각 세션의 시간 정의를 어떤 이들은 단지 20분 동안 지속된다고 하고 다른 이들은 한 시간 동안 지속된다고 함
  - 많은 사람들은 세션이 길수록, 수확 체감의 법칙[4)]을 취할 수 있다고 생각함
- 테스터들은 한 세션에서 더 길게 작업할수록 덮여진 요구된 영역(desired area)이라고 정의된 것과 크게 상관이 없는 것도 알 수 있다고 생각함
  - 세션 기반 테스트에서 요구된 결과(desired result)는 테스트 규정에 매개변수를 전달하는 것이지만 이 규정들은 테스터들이 과정을 진행하면서 진화됨
  - 마지막에 세션 기반 테스팅은 세션 보고를 통해 결과의 구조화 전달을 감안함
- 세션은 테스트 매니저가 연구결과를 완전히 보고하는 것을 확인하는 과정을 통해 Bash의 PROOF 방법을 사용하여 보고되며 단계는 아래와 같음
  - Past(과거): 세션동안에 무슨 일이 일어났는가?
  - Result(결과): 세션동안에 무엇을 이루었는가?
  - Obstacles(장애물): 좋은 테스팅 방식에는 무엇이 있는가?
  - Outlook(전망): 여전히 완료가 필요한 것은 무엇인가?
  - Feelings(기분): 테스터들은 이 모든 것에 대해 어떻게 생각하는가?

- 탐색적 테스트를 효과적이게 만드는 자유도를 유지하면서 범위를 제한하는 방법을 하나 더 살펴보도록 함

---

3) http://www.satisfice.com/sbtm/, Bach는 세션에서 녹음 스튜디오에서 녹음하는 사람의 것을 모방하는 용어로 세션을 사용함

4) 수확체감의 법칙 : 생산요소를 추가적으로 계속 투입해 나갈 때 어느 시점이 지나면 새롭게 투입하는 요소로 인해 발생하는 수확의 증가량은 감소한다는 것

3. 마인드 맵(Mind Maps)

- 탐색적 테스팅에서 마인드맵 방법론은 어느 정도 따라해야할 가치가 있음
  - Jonathan Kohl는 "탐색적 테스팅의 이해(Demystifying Exploratory Testing)"[5]에서 탐색적 테스팅에 있어 마인드 맵의 사용을 권장했음
  - 마인드 맵은 그 자체가 자유로운 사고 운동이기 때문에 탐색적 테스팅과 잘 공존할 것으로 보임
- 중요한 것은 구조화를 통해 탐색적 테스팅동안에 발견된 것을, 테스트에서 도출된 변경을 처리하는 사람들을 지원하는 방식으로 오류를 발견하고 이를 분류하고 조직화하는 것임
- 현실적으로 소프트웨어에서 스크립트 테스팅과 탐색적 테스팅은 근본적으로 방법론이 다를 수밖에 없는 양 극단에 있으나, 기억해야할 것은 하나를 극단적으로 받아들인다면 결과는 최적보다는 미흡하다는 것임
- 이 둘 중 어느 것을 사용해야 하는지에 대한 확실한 공식은 없으나 이 둘의 적절한 적용은 제품의 품질 측면에서 진화를 기대할 수 있을 것임

참고 자료

1. http://kaner.com/?p=46
2. http://www.satisfice.com/sbtm
3. http://blog.bughuntress.com/automated-testing/when-to-use-exploratory-testing-and-what-it-gives-as-compared-to-scripted-testing

5) http://www.stickyminds.com/better-software-magazine/demystifying-exploratory-testing