소프트웨어 테스팅과 QA에 관한 진실 by 신재문 techbard@TERAMAIL.com

이번에는 오래되었지만 아직도 그 심오한 내용에 공감이 가는 Bill Hetzel의 "The Complete Guide to Software Testing"에 언급된 Testing Principle에 대한 필자의 생각을 풀어보고자 한다. 아래에 원리를 나열하고 그에 대한 필자의 생각을 달았다.

Principle 1 - Complete Testing Is Not Possible

많은 프로그래머들은 자신들의 프로그램을 "충분히" 테스트했다고 생각한다. 이것은 진실이 아니다. 아무리 간단한 프로그램이라 하더라도 테스트 해야 할 경우의 수라는 것은 관리할 수 있는 범위를 넘어선다. 테스터들 또한 상황은 마찬가지이다.

완전한 테스팅이란 없다. 즉, 아무리 테스트를 많이 하더라도 완전한 것이 아니다. 주의해야 할 점은 완전한 테스트와 적절한 테스트 그리고, 불충분한 테스트 간의 경계를 찾는 것인데 이것은 매우 어려운 일이다. 하지만, 전략적 사고를 통해 현재의 상태가 어떤 상태인지 또 어떤 상태로 나아가야 하는지에 대한 판단은 필요하다.

이미 완전한 테스트란 가능하지 않기 때문에, (주의할 점은 그렇다고 아무리 해도 완전할 수 없기 때문에 아무렇게나 해야 한다는 뜻은 절대 아니다.) 남은 것은 적절한 테스트와 불충분한 테스트이다.

이때 생각해 볼 수 있는 것은 테스트의 내용과 범위에 대한 것인데, 여기에는 일반적인 규칙이 있다.

1. 아무렇게나 생각나는 대로(보이는 대로) 동작시켜 보고 테스터 본인이 판단하기에 이상하다고 생각되는 버그를 리포트 하는 것.
2. 요구사항 문서가 존재한다면 문구의 내용을 검증하려는 의도로 (여기서의 검증의 의미는 언급한 기능, 된다고 한 것, 안 된다고 한 것이 그대로 구현되어 있는지 확인) 그 내용을 테스트하는 것.
3. 비 기능적 요구 사항 (요구 사항 문서에 언급되어 있을 수도 안 되어 있을 수도 있음)에 대한 테스트를 수행하는 것. (이것은 일반적으로 성능성, 가용성, 편의성 등이 있음)
4. 요구사항에 언급되지 않으나 유저나 고객, 업계나 표준에서 중요시 하는 특성에 대한 테스트 (업계의 경쟁 제품이 가지는 공통 요구사항, 표준에서 정의한 요구사항, 과거 제품이 가졌던 요구사항, 경험상 필요한 요구사항 등)

위의 내용은 테스트의 내용과 범위에 대한 고려 사항이다.

여기에는 또 두 가지 고려 사항이 존재하는데,

1. 위에서 언급한 테스트 범위에 대한 고려
2. 각 단위 범위 내에서의 조합에 대한 고려

즉, 한 가지 기능을 검증하기 위해서는 각 시나리오가 가질 수 있는 조합을 커버하는 것이 필요하다. 물론 모든 커버리지를 커버하는 것이 좋겠지만, 이것은 위에서 언급한 "완전한" 테스트 자체가 불가능하다는 전제에 의해서 매사에 모든 커버리지를 커버하는 것이 좋지 않을 수도 있다. (이 역시 현명한 판단이 요구된다.)

따라서, 단위 시나리오가 있다면 일반적인 경로에 대한 커버는 당연히 테스트 내용에 포함되어야 하고, 더불어 예외 경로 들에 대한 테스트 또한 포함시켜야 한다는 의미이다.

Principle 2 - Testing Work Is Creative and Difficult

테스팅에 대한 잘못된 믿음

- 테스팅은 쉽다.
- 누구라도 테스팅을 수행할 수 있다.
- 훈련이나 교육, 경험이 필요치 않다.

위의 "잘못된 믿음"이 진실이 되려면, 다음과 같은 조건이 충족되면 된다.

1. 시스템이 매우 단순하다.
2. 따라서, 시스템 전체를 이해하기가 매우 쉽다.
3. 개발팀에 해당 시스템을 여러 번 구현한 경험이 있으며 현재 시스템은 이전 구조에서 크게 변경되지 않았다.

정도의 차이는 있겠지만 현대의 IT 시스템들은 계속 어려워 지고, 복잡해 지고, 거대해 진다는 사실을 누구도 부정하지 못할 것이다. 또한, IT 개발 인력이라는 것이 이직이 심하고 지속적인 커리어 유지가 쉽지 않은 점을 생각할 때, 3번의 논제 또한 충족되기에 매우 어렵다.

따라서, 현재의 IT 시스템에서는 테스팅이 매우 어렵고, 창조적인 사고를 필요로 하며, 교육과 훈련, 경험이 필요한 것이다.

이 논제에 대해서 필자가 말하고 싶은 것은 다음의 두 가지 이다.

1. 사람에 따라서 뛰어난 센스를 가져 별도의 교육을 받지 않고도 감각에 의해 버그를 찾아내는 사람이 존재한다.

2. 하지만 센스에 의존하지 않고, 일반적인 시스템의 규칙이나 테스트 디자인 교육과 경험, 테스팅 절차를 숙지한 사람이 장기적으로 더 좋은 성과를 낸다.

테스팅 프로젝트를 진행해 본 사람들은 테스팅이 일회성의 테스트 수행으로만으로 끝나지 않으며, 반복적인 활동이라는 점을 잘 알고 있다. 즉, 산출물 빌딩 이후의 과정이 테스트 - 수정 - 재테스트의 단위 싸이클을 가지고 반복적으로 어느 수준에 도달할 때까지 수행되어야 한다는 것이다. 이것은 테스팅 센스가 아닌 어떤 형태의 경험과 교육을 받아야 하는 문제이다. 특히 필자는 개발 프로젝트의 시작과 끝 (요구사항 정의 - ... - 유지보수)을 모두 경험해 보는 것이 중요하다고 생각한다. 여러분들도 기회가 된다면 프로젝트의 시작 부터 함께 해 보아라. (아마 그럴 기회가 많지는 않을 것이다.)

Principle 3 - An Important Reason for Testing is to Prevent Deficiencies from Occurring

"테스팅은 하나의 액션이 아닌 반복적인 단계별 활동이 되어야 한다." 이 말에는 대단히 많은 의미가 함축되어 있다. 지금부터 이 의미를 따져보자.

많은 사람들이 테스팅에 대해서 지엽적으로 생각하고 있는 부분중 하나는 테스팅이 단순히 버그를 찾아내는 일이라는 점이다. QA와 테스팅을 엄밀히 분류하자면 진실이다. 하지만, 테스팅만으로는 전체적인 품질 수준과 품질 있는 산출물을 개발해 내는 본질적인 능력 자체를 끌어올리는 데는 한계가 있다. 이는 테스팅이 반복적인 활동이라는 점과 관련이 있는데, 버그 하나를 찾아 수정하고 수정 확인을 하는 것은 개발 능력 자체가 현저하게 끌어올려진 것은 아니며, 단순히 산출물의 품질만 끌어 올린 것이다. 당연히 품질 있는 산출물을 만들어 낼 수 있는 개발 능력이 뒷받침되지 않으면, 버그를 수정해 품질을 일시적으로 높인다 하더라도 이후에 또 다시 동일 수준의 품질 수준이 될 가능성이 농후하다. 하지만, 개발 활동이 정규화, 절차화된다면 그것은 품질 있는 산출물을 산출할 수 있는 개발 능력 자체가 끌어올려진 것이다. 따라서, 상황상 QA와 테스트가 엄밀히 구별되어 있지 않은 우리의 환경에서는 테스팅 역할을 하는 부서의 비젼이 QA의 본질적인 역할, 즉 Detection이 아닌 Prevention을 위한 모든 활동으로 점진적으로 맞춰져야 한다고 생각한다. 이를 위해서는 두 가지의 활동이 필요한데,

1. 테스팅 자체의 품질을 높일 수 있는 활동 (테스트 디자인, 기법, 절차, 계획, 도구 사용 등)
2. 개발 능력 자체의 품질을 높일 수 있는 활동 (개발 프로세스, 협업 가속화 방안, 요구사항 형성 체계화, 의사 결정 과정의 정형화, 기술적 도구의 사용, 개발 앞단의 리뷰 강화 등)

이다. 즉, QA/테스팅의 수준을 다음과 같다고 보았을 때 뒷 부분의 개선 영역도 고려해야 한다는 것이다.

개개의 테스터가 버그를 잘 발견하는 것 < 다수의 테스터가 효과적으로 버그를 잘 발견하는 것

< QA의 시각을 테스팅 뿐만 아니라 개발 영역으로 확대하는 것 < 개발 과정의 품질이나 개발 능력을 높일 수 있는 이슈를 제기하는 것

Principle 4 - Testing is Risk-Based

앞의 전제로 모든 경우의 수, 완전한 테스팅은 불가능하다는 점을 언급했다. 따라서, 제한된 인력과 시간을 가지는 경우 우선 순위를 정해 테스팅을 수행하는 이른바 "위험을 안고 가는" 테스팅 전략이 필수적이다. 이때 필요한 것은

1. 우리의 산출물에서의 위험이라는 것을 어떻게 정의할 수 있나? (프로젝트 목표와 위험과의 관계는?)
2. 누구에게 위험이라는 것인가? 관련 이해 당사자들이 위험이라고 동의하는가?
3. 그 위험의 우선순위를 정할 수 있나?
4. 남아 있는 리소스를 가늠해 보아 우선 순위가 높은 위험을 커버할 수 있는 테스팅을 계획한다.

라고 볼 수 있다. 조금 더 리소스가 제한적이라면 어떨까? 이때는 다음 사항도 해봄직하다.

1. 표준에 맞춰야 한다면 표준만을 커버하도록 테스팅을 계획한다.
2. 경쟁 제품이 있다면 업계 표준적인 기능만을 커버하도록 테스팅을 계획한다.
3. 단순히 요구사항 문서의 내용만을 검증하려는 의도로 테스팅을 계획한다.
4. 과거에도 유사한 구조로 개발된 이력이 있다면, 예전의 테스트를 참고하여 테스팅을 계획한다. (예전에 크게 문제가 되었던 경우만을 확인)

이때 매우 중요한 것은 이러한 의도의 테스트가 계획되어 있음을 개발 부서와 협의하여 서로 간에 이해하는 일이다. 심한 경우에는 이러한 Risk taking의 책임을 지지 않으려고 신경전을 벌이는 경우가 있을 수 있는데, 이것은 글 몇 자로 설명하거나 해결할 수 있는 문제가 아니다. 어쨌든, 중요한 것은 서로 간에 협의한 내용에 대해서 수행한다는 점이고, 테스팅 부서는 본인들이 약속한 내용에 대해서 완수하려는 노력을 보여야 한다는 점이다.

Principle 5 - Testing Must Be Planned

계획이라는 말은 일반론에서는 "하면 좋다"의 의미로 받아들여지기 쉽다고 생각한다. 테스팅에서는 단순히 일정 계획이나 인원 계획을 세우고 진행한다는 의미라기 보다는 좀 더 광의로 생각해 볼 수 있다. 즉, 전략적이고 계획적이어야 한다는 의미인데, 이것은 다음의 의미를 가지고 있다.

1. 계획을 세웠을 때야 계획 대비 달성율이 어느 정도이며 이것은 의도된 것인지 의도되지 않은 것인지를 판단할 수 있다. (물론 실행의 의도가 전혀 없는 보고용 계획을 말하는 것이 아니

다.)
2. 만약 미리 세운 계획에 미달하는 경우 재조정을 통해 다음 계획의 정확성이 높아질 수 있다.
3. 리소스 확보의 관점에서 계획을 여러 차례 수행한 전력이 있다면 (계획이 맞던 안 맞던), 추가적인 리소스 확보에 용이한 근거가 된다.
4. 테스팅을 수행하는데 있어 임의적인 테스트가 아닌, Scripted 테스팅 (즉 문서화 작업을 하고 난 이후에 검토를 마치고 그대로 수행하는 작업) 또한 계획의 의미에 포함될 수 있다.
5. 계획의 의미는 검토와 리뷰, 제출의 의미를 가지고 있기 때문에 문서화가 당연히 전제된다.
6. 문서화의 장점으로도 볼 수 있지만, 계획을 문서화 하는 경우, 여러 단계를 반복하는 경우 계속적으로 나아지고 완전해 질 수 있다. 당연한 얘기지만, 이런 계획이 나아진다면 그에 따른 테스팅 활동 또한 나아질 것이다.

Principle 6 - Testing Requires Independence

필자의 예전 경험에 QA팀이 뿔뿔이 흩어져서 개발팀에 자리를 위치한 적이 있었다. 이때의 장점은 다음과 같다.

1. 개발자들과 개인적인 친분을 쌓을 기회가 충분히 많다.
2. 개발 과정을 옆에서 같이 지켜보며 어려운 점, 일하는 방식 등을 배울 수 있다.
3. 그들의 용어를 자연스럽게 익히게 되어 그들의 언어로 말하는 법을 배울 수 있다.

세상의 모든 일이 그렇듯이, 장점이 있으면 단점도 있는 법 단점은 다음과 같다.

1. 개인적인 친분 때문에 친한 개발자의 버그를 제출하는 일에 머뭇거리게 된다. (필자는 실제로 친한 개발자에게 해당 버그를 먼저 얘기하고 리포트는 일부러 하지 않은 실수의 경험이 있다.)
2. 개발의 편의라는 명목에 의해 문서화나 부서간 공식적인 산출물 제출 등의 명제가 희생된다. 잘 알고 같이 있기 때문에 절차라는 부분에 대해서 생략한 경우도 포함된다.

일단 이 주제에 대한 필자의 생각은 다음과 같다.

1. 개발 과정에 대해서 테스터는 반드시 참여하여 배워야 한다.
2. 이를 위해서는 개발팀 테스팅의 이슈를 도와주며 이를 배울 수 있는 개발팀 지원 테스터의 역할도 생각해 볼 수 있다. (쥬니어의 경우)
3. 조직도상 개발팀과 완전히 별개의 부서에 QA팀이 위치하며, 적어도 개발팀의 하부 조직이어서는 곤란하다.
4. QA/테스팅/품질 이슈에 대한 마인드가 있는 부서장이 반드시 필요하다. (이때 직급 부분

도 고려 대상이다.)

5. QA/테스팅 조직의 독립성과 고립은 별개의 문제이다. 독립성은 유지하되 그 끈을 계속 유지할 수 있는 가능한 모든 방법의 동원이 요구된다.

필자가 두서 없는 생각을 유명한 책의 힘을 빌어 서술해 보았다. 테스팅에 고군분투하시는 여러분들에게 도움이 되었으면 하는 바램이다. 차후에 또 다른 기회가 있다면 또 다른 테스팅의 원리를 가지고 필자의 생각을 풀어보도록 하겠다. May the force be with you!