8장. 실험주의 과학적 사고와 실천

실험

- 가설을 지지, 반박 또는 검증하기 위해 수행되는 절차
- 실험은 특정 요인을 조작했을 때 어떤 결과가 발생하는지 보여줌으로써 원인과 결과에 대한 통찰력을 제공

예시

- 우리는 심지어 리처드 파인먼처럼 가장 중요하거나 카리스마가 있거나 유명한 사람의 말에 따라 의사결정을 내리는 것에서 벗어나 증거에 기반한 의사결정과 선택을 해야만 합니다.

실험주의자가 되는 4가지 접근 방식

- 피드백
  - 우리는 피드백을 진지하게 받아들여야 하며, 명확한 신호를 제공할 결과를 수집해 우리가 생각하고있는 지점에 이 결과를 효율적으로 전달하는 방법을 이해할 필요가 있다
- 가설
  - 평가하고자 하는 아이디어를 염두에 둘 필요가 있다.
- 측정
  - 가설에서 테스트하려는 예측을 평가하는 방법에 대한 명확한 아이디어가 필요하다. 이 컨텍스트에서 '성공' 또는 '실패'는 무엇을 의미할까?
- 변수 통제
  - 실험이 우리에게 보내는 신호를 이해할 수 있도록 가능한 한 많은 변수를 제거할 필요가 있다

피드백

질주 본능

나는 복잡한 금융 거래 소프트웨어를 제작하는 회사에서 일한 적이 있다. 개발자들은 매우 뛰어났고 회사는 성공했지만 사람들은 자신이 더 잘할 수 있다는 사실을 알고 있었고, 내가 맡은 일은 소프트웨어 개발 관례를 개선하는 것이었다.

내가 합류했을 당시에는 자동화된 테스트에 대해 상당히 효과적인 접근 방식을 채택하고 있었다. 테스트가 상당히 많았다. 밤새 빌드 서버를 운영했고, 제공한 서비스는 모든 테스트 실행을 포함하여 완료하는 데 9시간 30분이 걸리는 대규모 C++ 빌드로 구성되어 있었다. 그래서 매일 밤 빌드를 실행했다. 한 개발자는 이런 방식으로 작업해 온 지난 3년 동안 모든 테스트가 통과된 경우는 단 세 번뿐이었다고 내게 말했다.

그래서 매일 아침 테스트에 실패한 모듈은 보류하면서 모든 테스트가 통과된 모듈만 골라서 릴리스했다. 통과한 모듈 중 하나가 실패한 모듈의 변경에 의존하지 않는 이상 괜찮았지만, 때로는 문제가 생기기도 했다.

변경하고 싶은 부분이 많았지만 우리는 첫 단계로 다른 변경사항 없이 피드백의 효율성을 개선하는 작업을 진행했다. 많은 실험과 힘든 작업 끝에 빠른 단계의 커밋 빌드를 12분 만에 실행하

고 나머지 테스트는 40분 만에 실행하는 데 간신히 성공했다. 9시간 30분이 걸리던 빌드를 더 빠르게 수행했다! 빌드 속도를 높이고 개발자에게 더 효율적으로 결과를 제공한다는 장점 이외에 조직, 프로세스 또는 도구에 다른 변화는 없었다.

이 변경사항이 릴리스된 후 처음 2주 동안은 모든 테스트를 통과한 빌드가 두 번 있었다. 그 후 2주 동안, 그리고 내가 그곳에서 일하는 동안 모든 테스트를 통과하고 모든 코드를 릴리스할 수 있는 빌드가 최소한 하루에 한 번 이상 있었다.

피드백 속도를 개선하는 것 외에는 다른 변화를 주지 않은 결과, 팀은 근본적인 불안정성을 수정하는 데 필요한 도구를 확보할 수 있었다.

이 사례에서는 개발자에게 제공하는 피드백의 효율성과 품질을 개선하기 위해 실험을 진행했다. 이 작업을
진행하는 동안 빌드 성능에 대한 더 나은 측정 기준을 수립하고, 향상된 버전 관리와 IaCinfrastructure as Code로 변수를 제어 했으며, 여러 가지 기술 해법과 빌드 시스템을 A/B 테스트로 진행했다

가설

“이어지는 프로세스에 따라 새로운 법칙을 찾는데, 먼저 이를 추측한다!”

- 추측이나 가설은 출발점이다. 다른 덜 효과적인 접근 방식과 비교해 과학과 공학의 차이점은 출발점에서 멈추지 않는다는 사실이다. 과학주의자가 되려면, 일단가설의 형태로 추측을 하고 나서 몇 가지 예측을 하기 시작하고 그런 다음에 이런예측을 확인할 방법을 찾으려 노력할 수 있다

- 가설을 검증하기 위한 일련의 실험을 진행하기 위해 우리의 생각과 작업을 구조화하는 것은 작업의 질을 향상하는 중요한 요소다.

측정

[사례 1]

- 내 고객 사이트 중 한 곳에 서는 테스트 커버리지 수준을 높임으로써 = 품질을 개선할 수 있으리라 판단했다. 그래서 측정 프로젝트를 시작해 데이터를 수집하고 개선된 테스트 커버리지를 장려하는 정책을 채택하기 시작했다. 여기서 ‘테스트 커버리지 80%’라는 목표를 설정했다. 그런 다음 이 측정 결과를 사용해 개발 팀에 장려책을 제공하고, 테스트 커버리지 목표 달성에 따라 보너스를 연계했다 ?

- 그들은 확보하고 있었던 테스트를 분석한 결과 테스트의 25% 이상에 단정문assert이 전혀 없다는 사실을 발견했다
- 즉, 이 조직이 원한게 코드 품질이였다면 직접 측정하는 편이 더 효과적이였을 것이다.
    - 측정의 방법도 매우 중요하다는 의미
-

[사례 2]

- 처음 시작할 때는 대기 시간과 처리량을 측정하는 데 매우 집중했기 때문에 '시스템이 초당 10만 개의 메시지를 2ms 이하의 지연 시간으로 처리할 수 있어야 한다'는 목표를 설정하고 측정값을 잡아내기 위해 열심히 노력했다. 첫번째 시도는 평균을 기반으로 했지만 나중에 의미가 없다는 사실을 알게 되었다.
    - 후속 거래 주기에서 최대 부하 초당 10만 개의 메시지와 맞먹는 수준을 초과해 초당 수백만 개의 메시지에 맞먹는 수치가 최고점을 찍을 때가 있었기 때문

- 결국 측정방법은 학습이 전부다.

변수 통제

- 피드백을 수집하고 유용한 측정을 하기 위해서는 현실적으로 최대한 변수를 통제 할 필요가 있다
- 버전 관리는 상용 환경에 릴리스하는 변경사항을 훨씬 더 정확하게 제어하도록 만든다.

---

1. 실험주의의 핵심 원칙

실험주의자가 되기 위한 접근 방식은 네 가지 주요 원칙으로 정의됩니다:

1. **피드백 (Feedback)**: 피드백을 진지하게 받아들여야 하며, 명확한 신호를 제공하는 결과를 수집하고 이 결과를 효율적으로 전달하여 사고 지점에 정보를 제공하는 방법을 이해해야 합니다. 이 루프를 닫는 것이 중요합니다.

2. **가설 (Hypothesis)**: 평가하고자 하는 아이디어를 염두에 둘 필요가 있습니다. 과학은 추측에 기반을 두고 있으며, 이 추측을 **'가설'**이라고 부릅니다. 가설은 출발점이며, 실험을 통해 이를 증명하거나 반증하는 방법을 찾아내야 합니다. 파인먼은 "여러분의 추측이 실험과 일치하지 않는다면 (여러분의 추측이) 틀린 것이다"라고 말했습니다.

3. **측정 (Measurement)**: 가설에서 테스트하려는 예측을 평가하는 방법에 대한 명확한 아이디어가 필요하며, 이 컨텍스트에서 '성공' 또는 '실패'가 무엇을 의미하는지 정의해야 합니다. 측정값은 신뢰할 수 있어야 하며, 잘못된 것을 측정하는 것은 해로울 수 있습니다 (예: 테스트 커버리지 대신 품질을 측정해야 함).

4. **변수 통제 (Variable Control)**: 실험이 보내는 신호를 이해할 수 있도록 가능한 한 많은 변수를 제거할 필요가 있습니다. 변수 통제는 실험을 더 안정적이고 반복 가능하게 만듭니다. 소프트웨어에서는 IaC(Infrastructure as Code) 같은 지속적인 배포 기술을 통해 변수를 통제할 수 있습니다.

2. 과학적 사고와 소프트웨어 개발

- **컴퓨터의 이점:** 소프트웨어 분야는 컴퓨터라는 환상적인 실험 플랫폼을 제공한다는 점에서 다른 분야에 비해 엄청난 이점을 가집니다. 마음만 먹으면 매초 수백만 개의 실험을 실행할 수 있습니다.
- **TDD와 실험주의:** 소프트웨어에서 가장 유연한 형태의 실험은 자동화된 테스트이며, 특히 **테스트 주도 개발(TDD)**은 코드를 작성하기 전에 예상 동작에 대한 예측(가설)을 포착하기 위해 테스트를 작성하도록 권장합니다.

◦ TDD를 연습하는 것은 문제 특성 파악, 가설 수립, 예측, 실험 수행의 단계를 따르는 것입니다.

◦ TDD는 설계와 구현에서 아이디어를 평가하고 개선할 수 있는 훌륭하고 확장 가능한 실험적인 플랫폼을 제공하여, **더 높은 품질**의 결과물을 더 빨리 만들 수 있게 합니다. TDD를 사용하는 고성과 팀은 유용한 작업에 44% 더 많은 시간을 소비하는 것으로 나타났습니다.

- **지식 체계로서의 테스트:** 우리가 작성하는 모든 테스트는 통제된 우주(시스템)의 동작 방식에 대한 이해를 단언하는 실험 모음을 포함하여, 시스템에 대해 우리가 알고 있는 **지식의 체계**가 됩니다.
- **경험주의와의 차이 (자기기만 방지):** 이 장은 단순히 현실을 관찰하여 결론을 내리는 경험주의(7장)의 함정을 지적합니다. 인간은 성급하게 결론을 내리도록 진화했기 때문에, 잘못된 결론이 '명백해' 보일지라도, 증거를 바탕으로 의사결정을 내리는 **과학적 합리주의**를 적용해야 합니다. 과학은 자신을 속이지 않는 것이 첫 번째 원칙입니다.

8장 실험주의는 소프트웨어 개발을 단순한 코딩이 아닌, 과학적인 추론 방식을 적용하여 안전하게 진행하고 더 나은 해답을 찾을 수 있도록 **규율이 잘 잡힌 접근 방식**을 취하는 도구임을 강조합니다.