



잘못된 테스트 사례 학습

Java/Spring 테스트를 추가하고 싶은 개발자들의 오답노트

왜 내가 하는 TDD는 실패하는가?

개요

테스트의 필요성에 대해 같이 공감해보고 테스트의 3분류에 대해 알아보시다.

제 1장

필요성

제 2장

테스트 3분류

1. 필요성

01. 테스트는 왜 필요한가

테스트가 왜 필요한지 살펴봅니다.

1. 필요성

테스트는 왜 필요한가? 레거시 코드

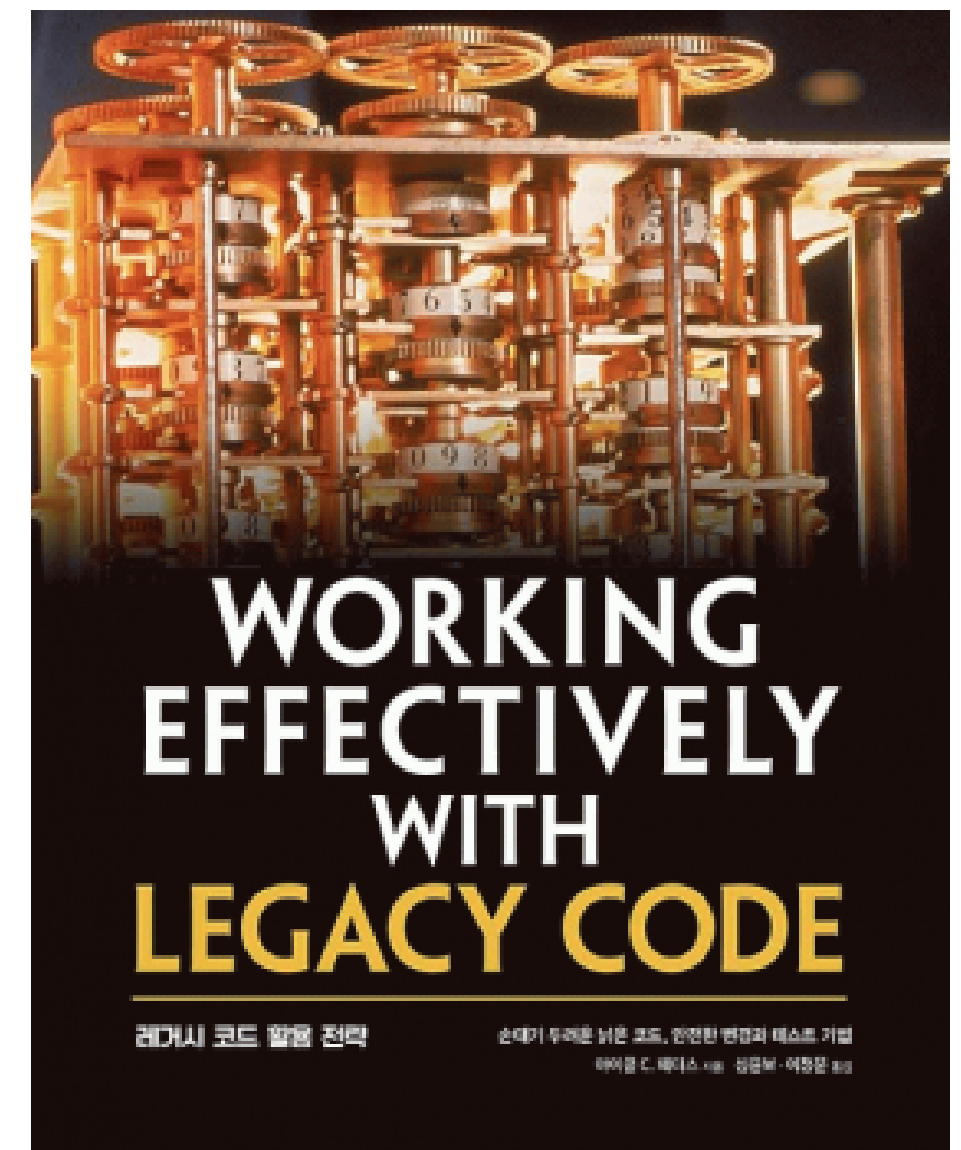


1. 필요성

테스트는 왜 필요한가? 레거시 코드

“ 내게 레거시 코드란, 단순히 테스트 루틴이 없는 코드다. 다만 이정의는 다소 불완전하다.

마이클 C. 페더스, 레거시 코드 활용 전략 손대기 두려운 낡은 코드, 안전한 변경과 테스트 기법, 심윤보 이정문 옮김, (에이콘, 2018-09-18), 12p



1. 필요성

테스트는 왜 필요한가? Regression

정상적으로 동작하던 서비스가
이번 배포로 동작을 안해요.

프로젝트에 합류한지 얼마 안되서 수정이 무서워요.
제 코드가 다른 코드에 영향을 주지 않을까요?



1. 필요성

테스트는 왜 필요한가? Regression

“ 11.1.1 구글 웹서버 이야기

타이터스 윈터스, 톰 맨쉬렉, 하이럼 라이트 큐레이션, 구글 엔지니어는 이렇게 일한다 구글러가 전하는 문화, 프로세스, 도구의 모든 것, 개앞맵시 역, (한빛미디어, 2022-05-10), 283p

2005년 구글 웹서버에 생산성이 급격히 떨어지는 문제 발생.

설상가상 이 시기에는 릴리즈 주기도 길었고 버그도 많아짐.

아무리 똑똑한 엔지니어를 투입해도 문제가 해결되지 않았고, 항상 **불안에 떨며 릴리즈**를 했음.

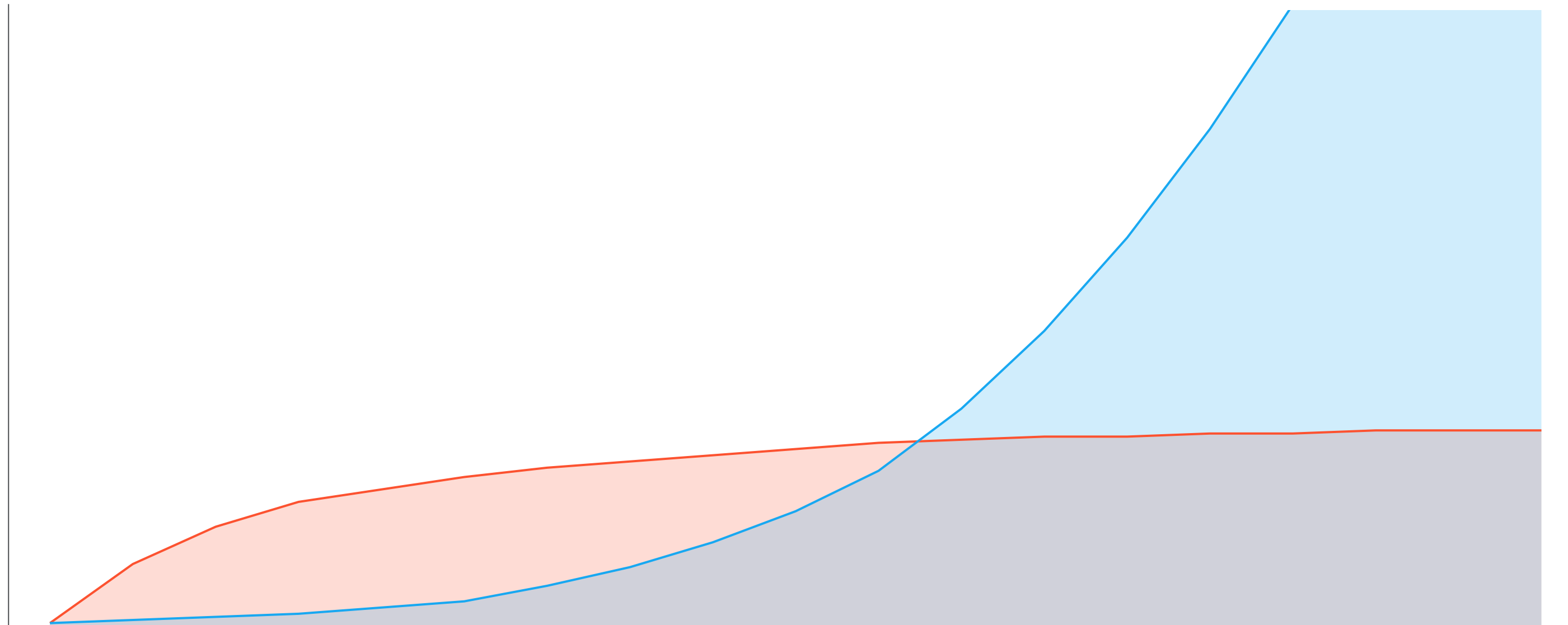
GWS 테크 리더는 **자동 테스트를 도입**하기로 결정함.

팀원들은 자신감있게 배포를 할 수 있게되었고 **그 결과 생산성이 올라감.**

1. 필요성

테스트는 왜 필요한가? 좋은 아키텍처를 유도

피쳐 하나를
개발하는데
드는 부담감



— 좋은 아키텍처 — 나쁜 아키텍처



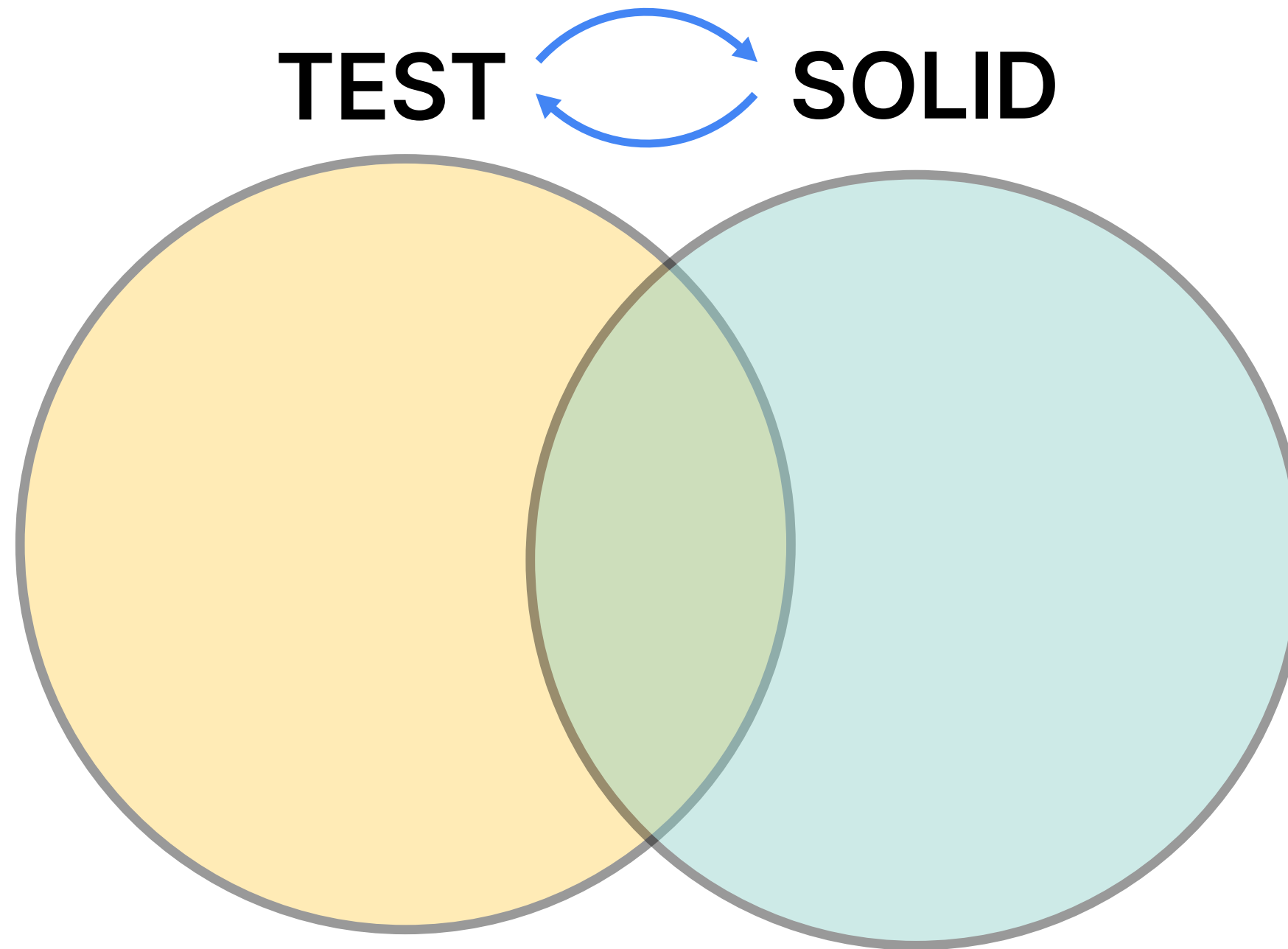
1. 필요성

번외) 좋은 아키텍처란?

S O L I D

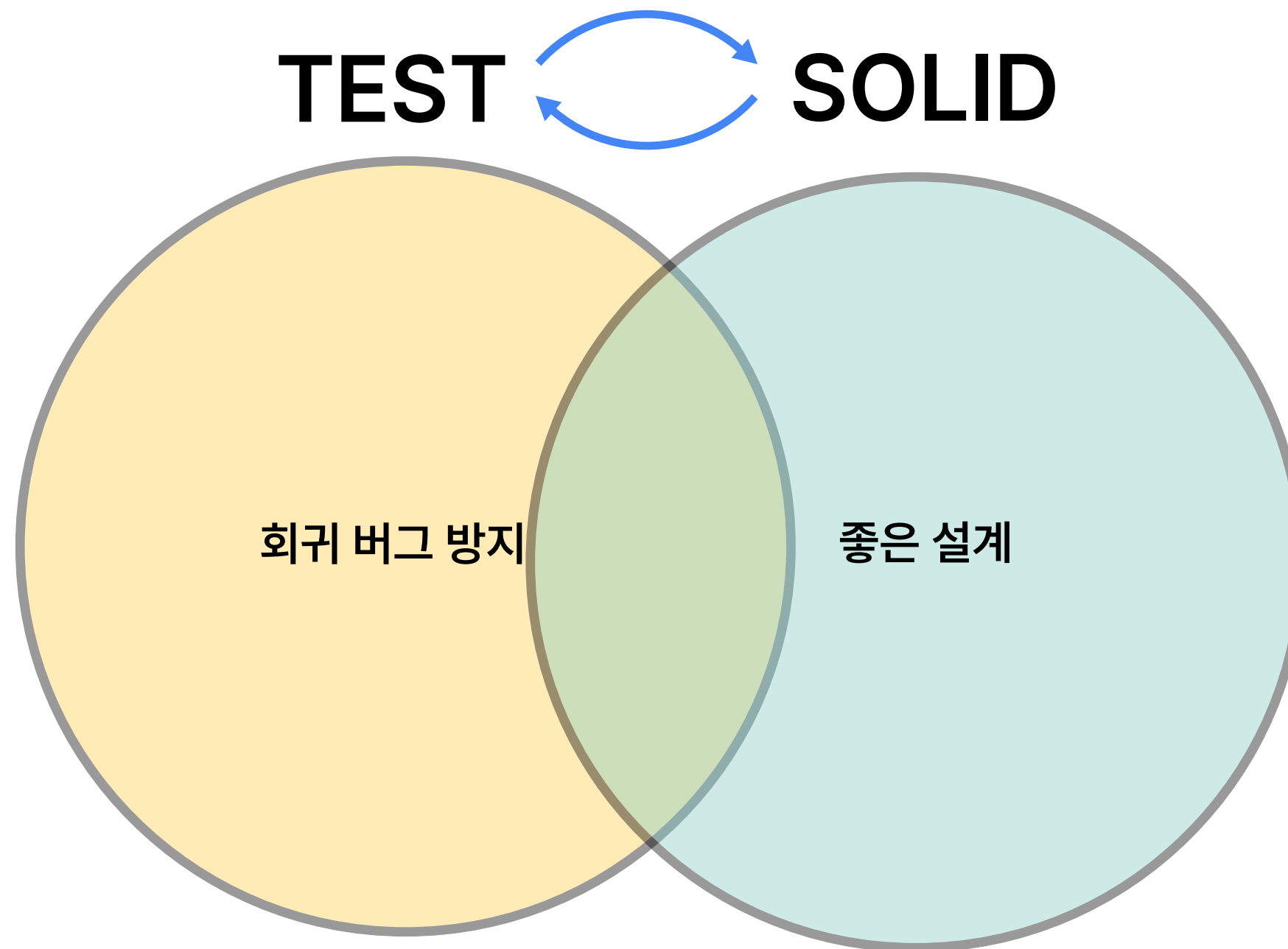
1. 필요성

번외) 좋은 아키텍처란?



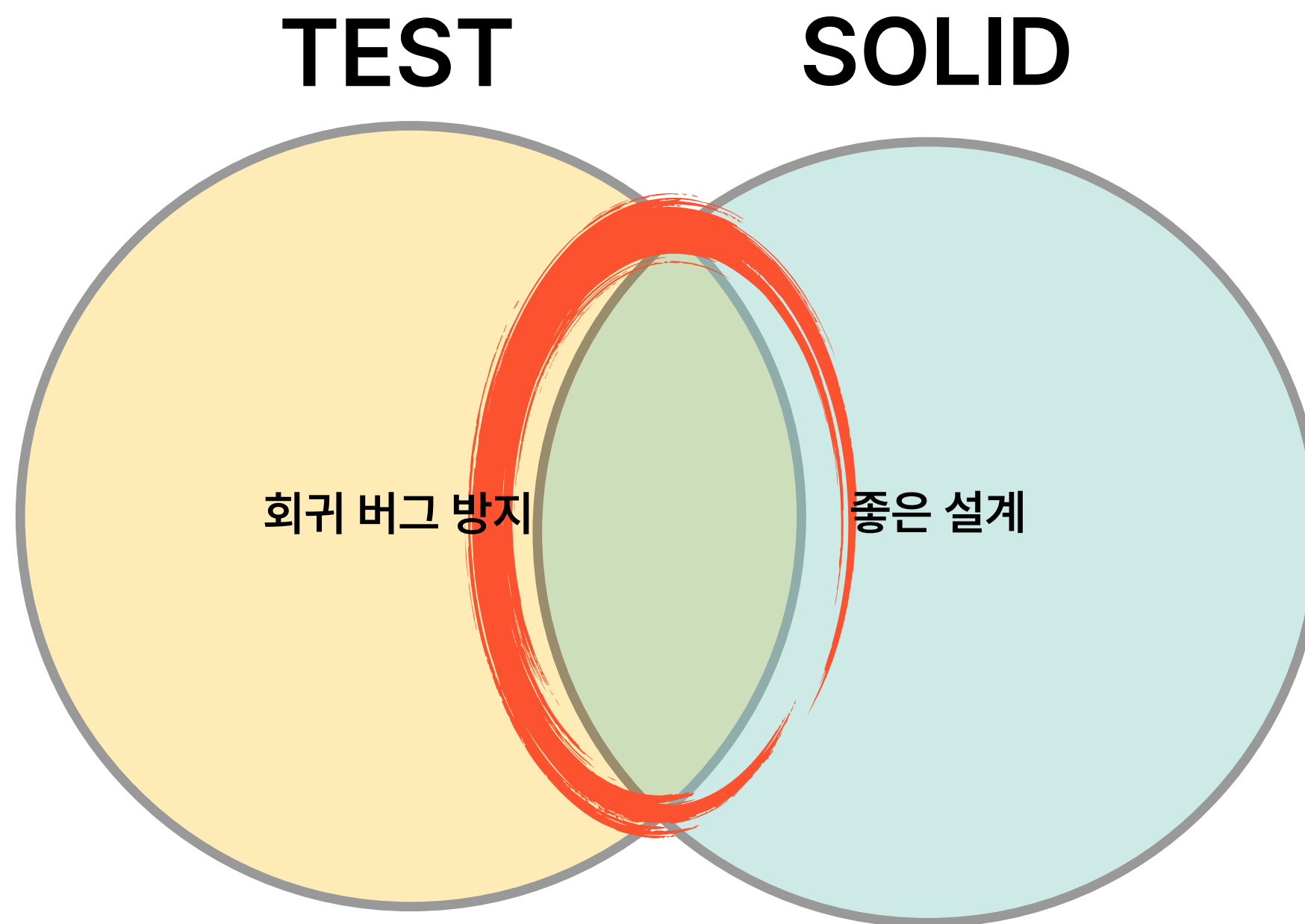
1. 필요성

번외) 좋은 아키텍처란?



1. 필요성

번외) 좋은 아키텍처란? 상호보완적



1. 필요성

번외) 좋은 아키텍처란?

SOLID



단일 책임 원칙


테스트는 명료하고 간단하게 작성해야하기 때문에, 단일 책임 원칙을 지키게 됨.

테스트가 너무 많아져서 이게 무슨 목적의 클래스인지 눈에 안들어오는 지점이 생김.
이 때가 클래스를 분할해야 하는 시점, 그러면서 책임이 자연스럽게 분배 됨.

1. 필요성

번외) 좋은 아키텍처란?

SOLID



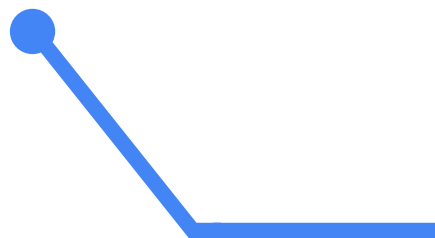
개방 폐쇄 원칙

테스트 컴포넌트와 프로덕션 컴포넌트를 나눠 작업하게 되고 필요에 따라 이 컴포넌트를
자유자재로 탈부착이 가능하게 개발하게 됨.

1. 필요성

번외) 좋은 아키텍처란?

SOLID



리스코프 치환 원칙

이상적으로 테스트는 모든 케이스에 대해 커버하고 있으므로,
서브 클래스에 대한 치환 여부를 테스트가 알아서 판단해줌

1. 필요성

번외) 좋은 아키텍처란?

SOLID



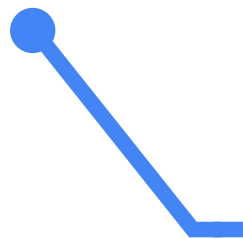
인터페이스 분리 원칙

테스트는 그 자체로 인터페이스를 직접 사용해볼 수 있는 환경.
불필요한 의존성을 실제로 확인할 수 있는 샌드박스.

1. 필요성

번외) 좋은 아키텍처란?

SOLID



의존성 역전 원칙

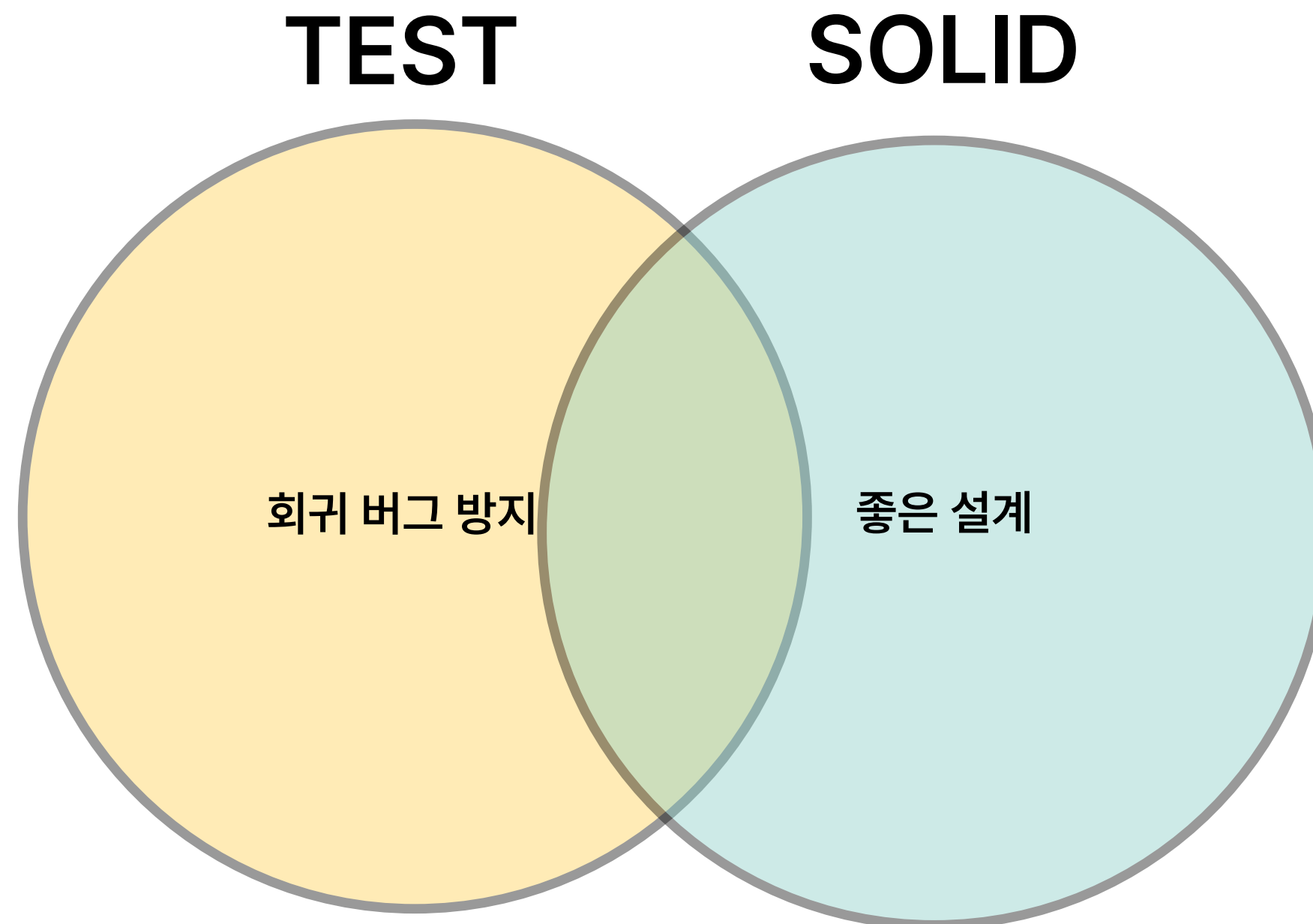
가짜 객체를 이용하여 테스트를 작성하려면, 의존성이 역전되어 있어야 하는 경우가 생김

1. 필요성

번외) 좋은 아키텍처란?

테스트가 SOLID 를 강제하는 것은 아니다.

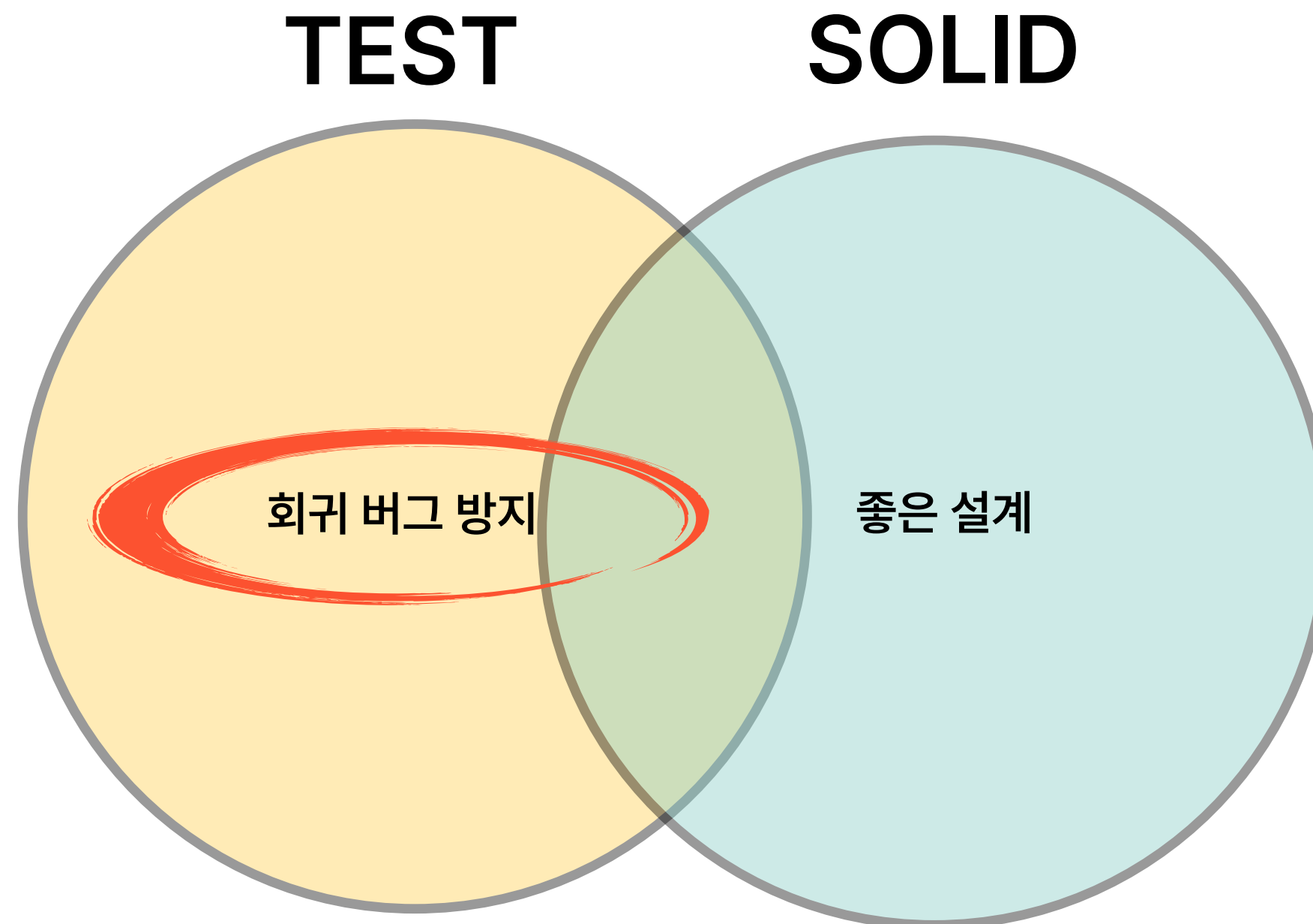
테스트를 넣으면서 이런 것도 함께 챙겨줘야 한다는 의미



1. 필요성

번외) 좋은 아키텍처란?

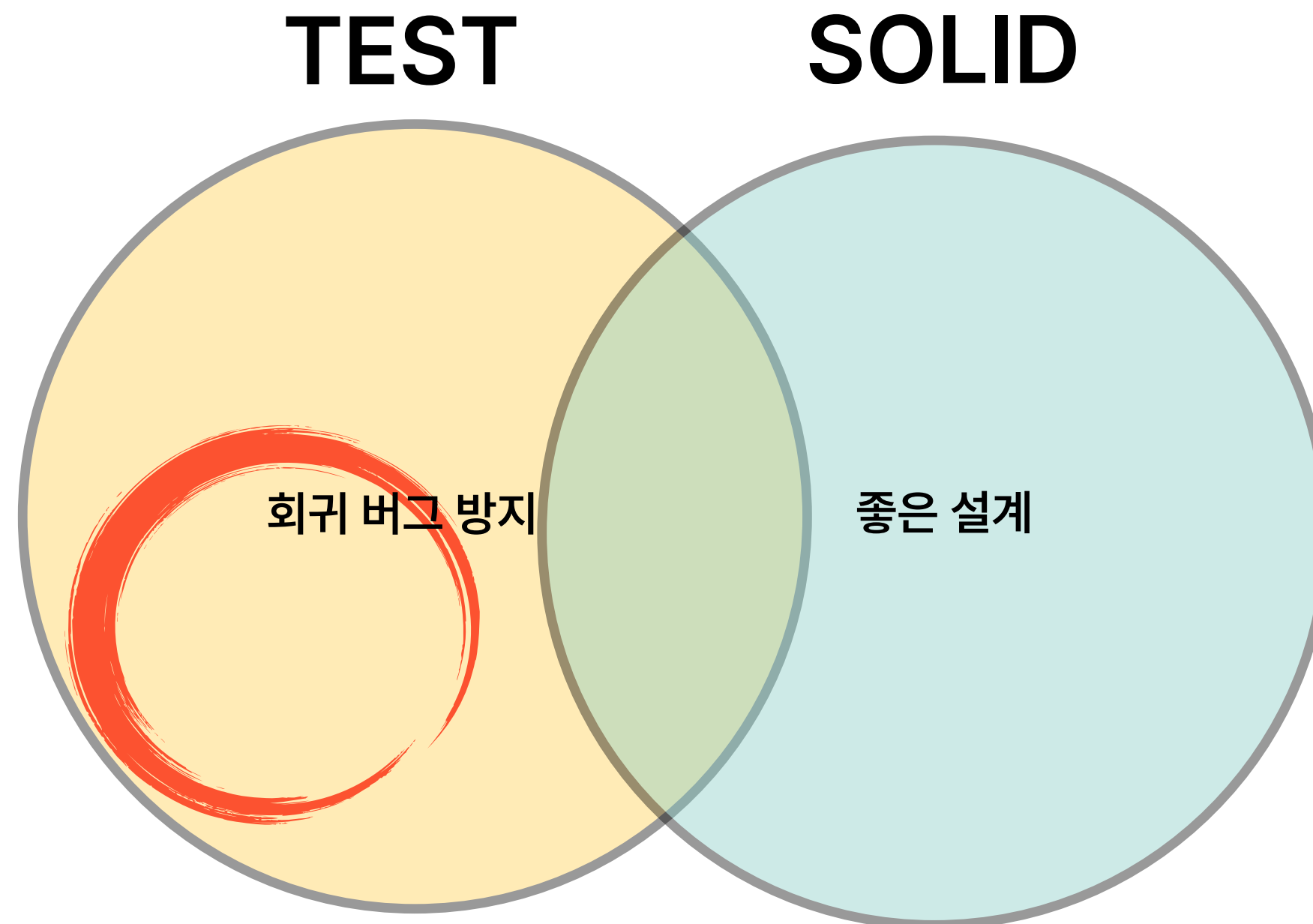
CASE 1. 회귀 버그도 잡고 좋은 설계는 약간 신경 쓴 경우



1. 필요성

번외) 좋은 아키텍처란?

CASE 2. 회귀 버그만 신경 쓰는 경우

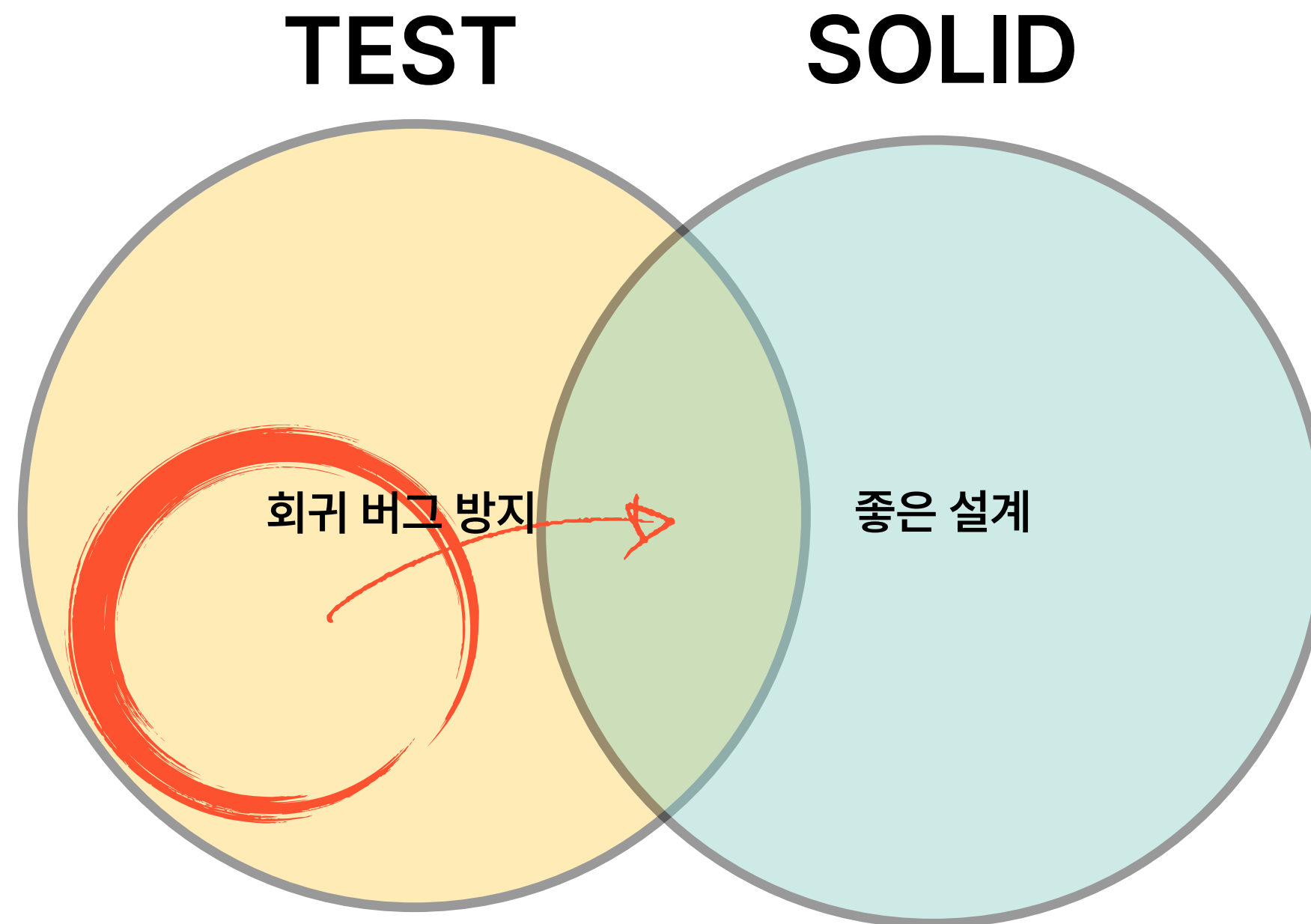


1. 필요성

번외) 좋은 아키텍처란?

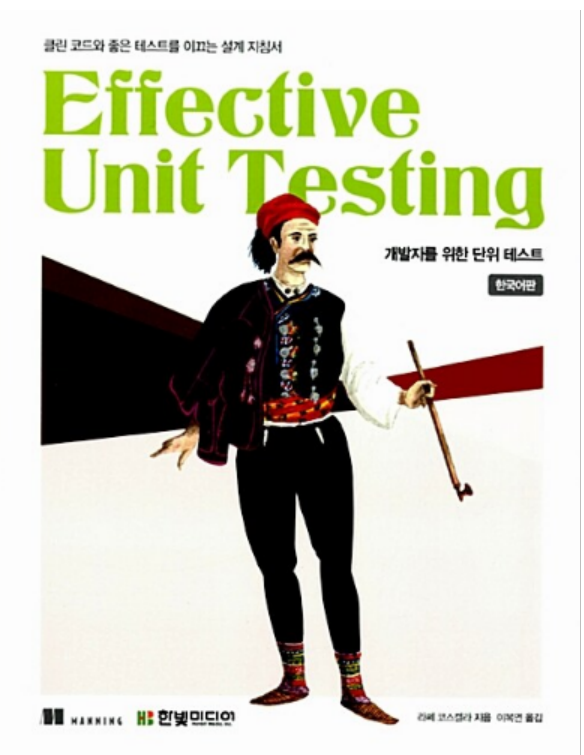
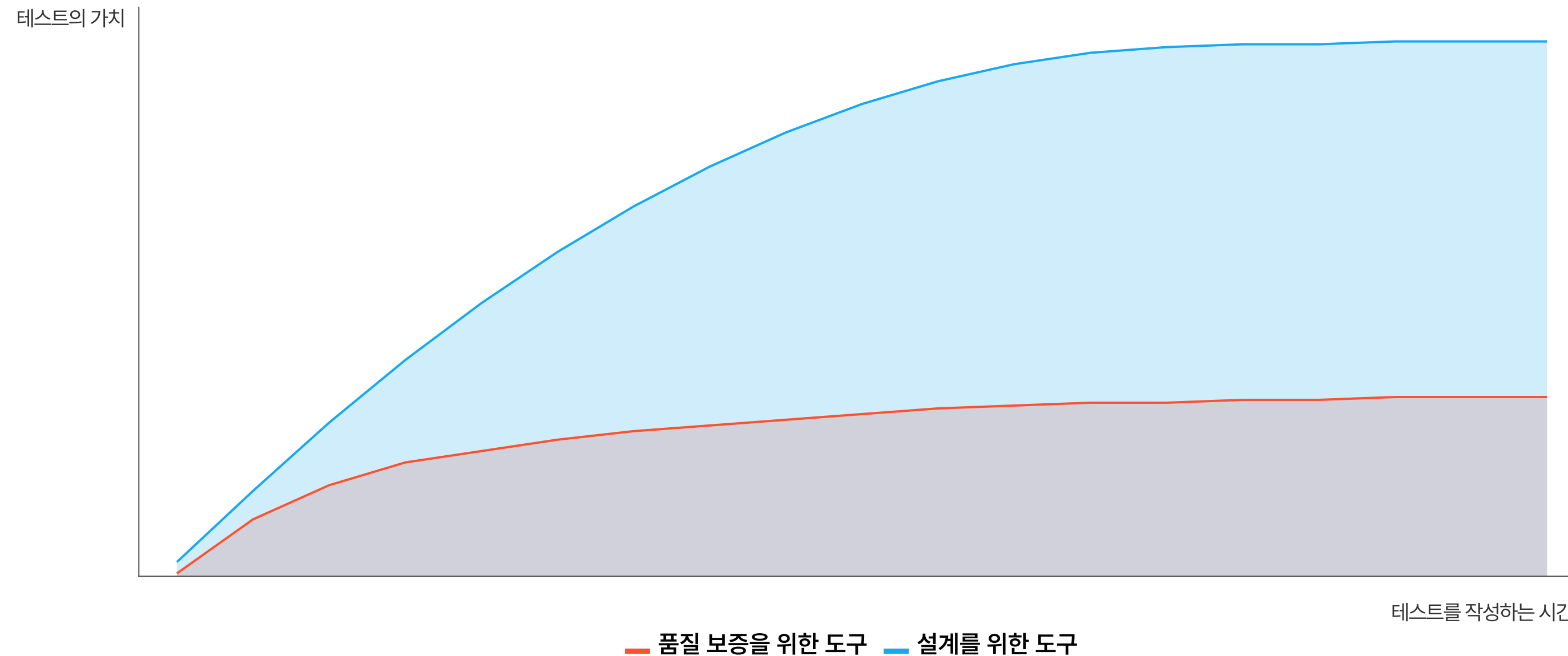
처음 테스트를 넣으려고 했을 때, 이유가 무엇이었는지 잘 생각해보시길

1. 회귀 버그 방지 / 2. 좋은 설계



1. 필요성

테스트 가능 설계 ref. Effective Unit Testing



2. 테스트 3분류

전통적인 테스트 3분류

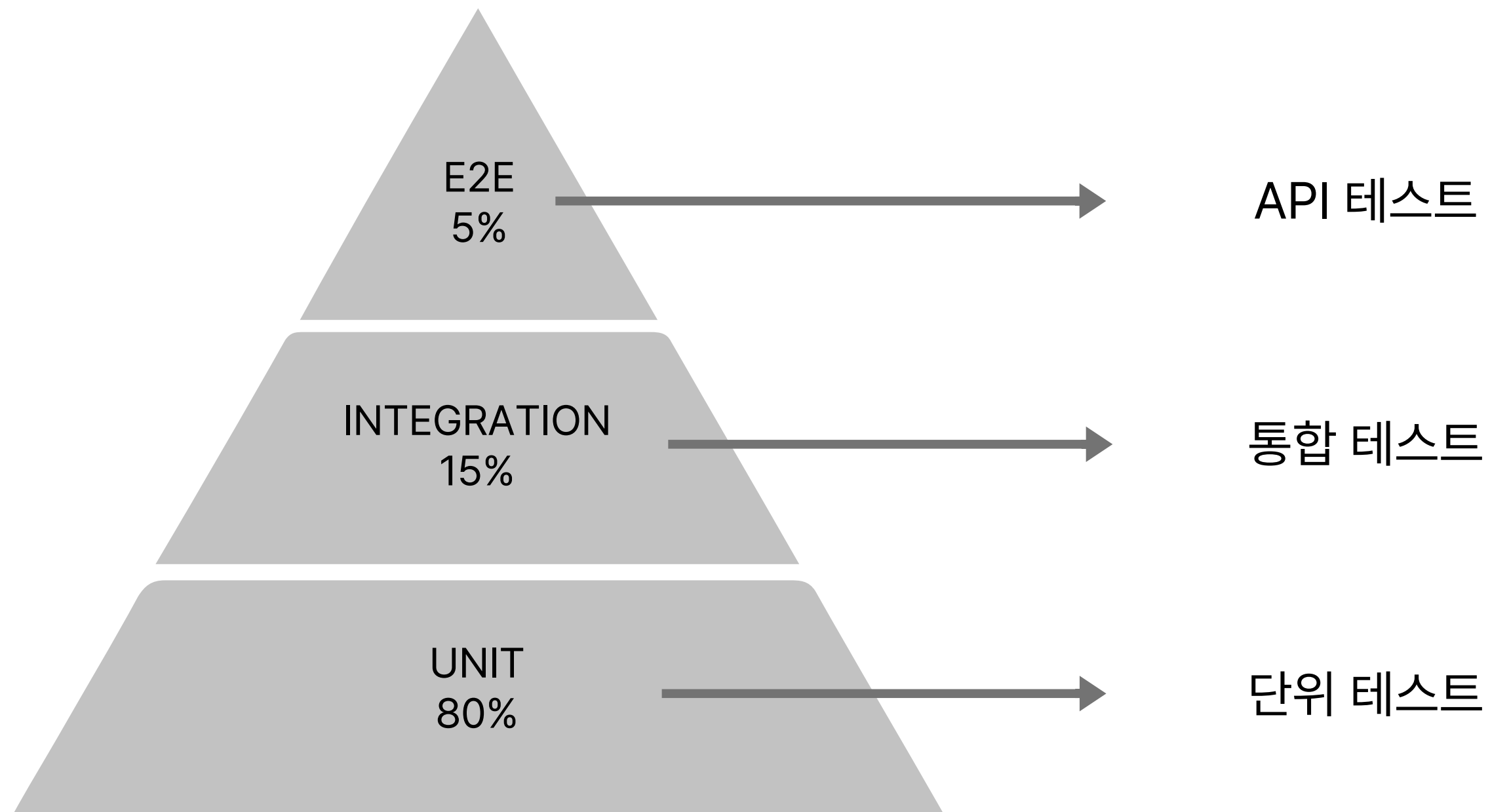
테스트의 전통적인 3분류 피라미드를 알아봅니다

구글의 테스트 3분류

구글에서 이야기하는 테스트 3분류를 알아봅니다

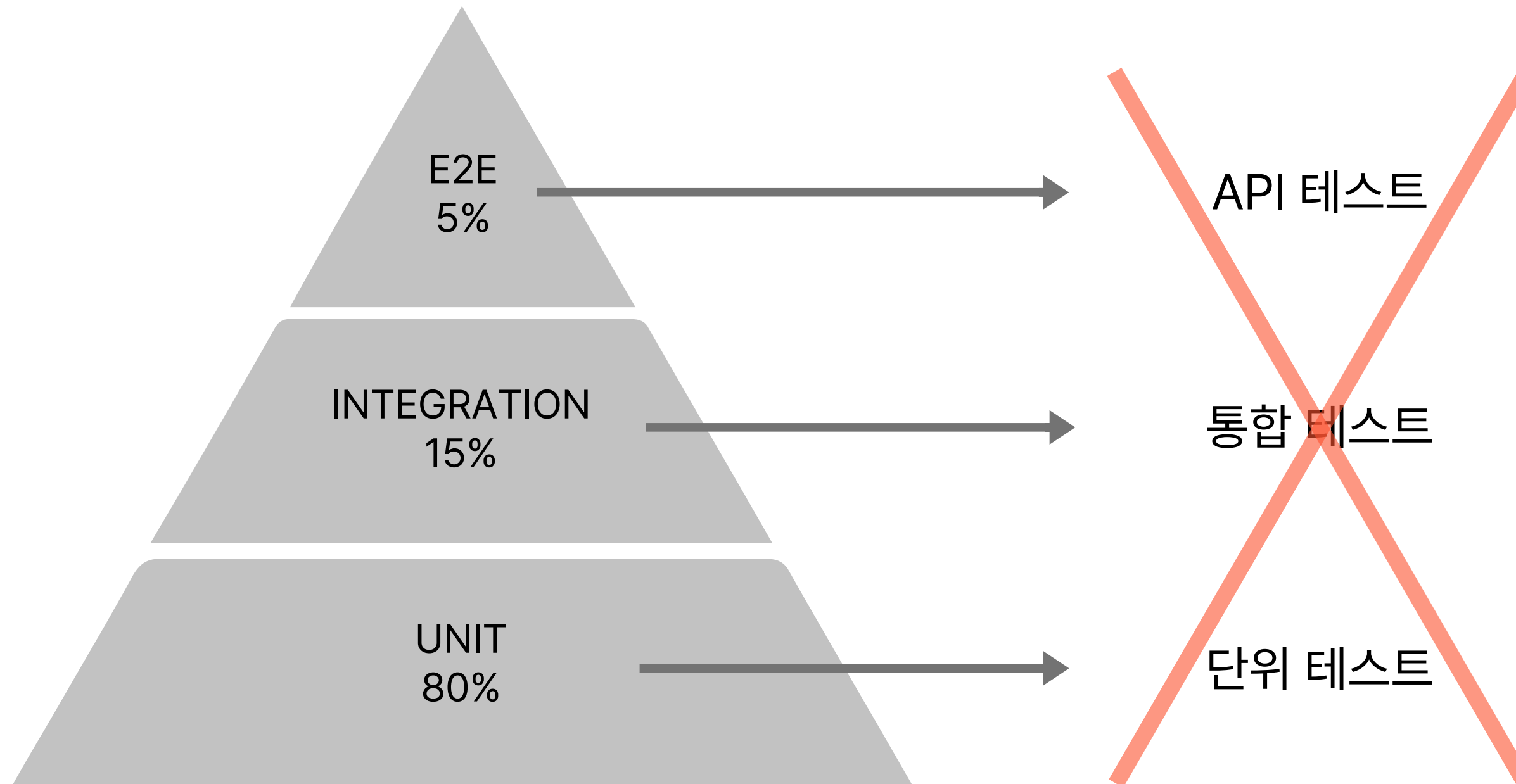
2. 테스트 3분류

전통적인 테스트 3분류



2. 테스트 3분류

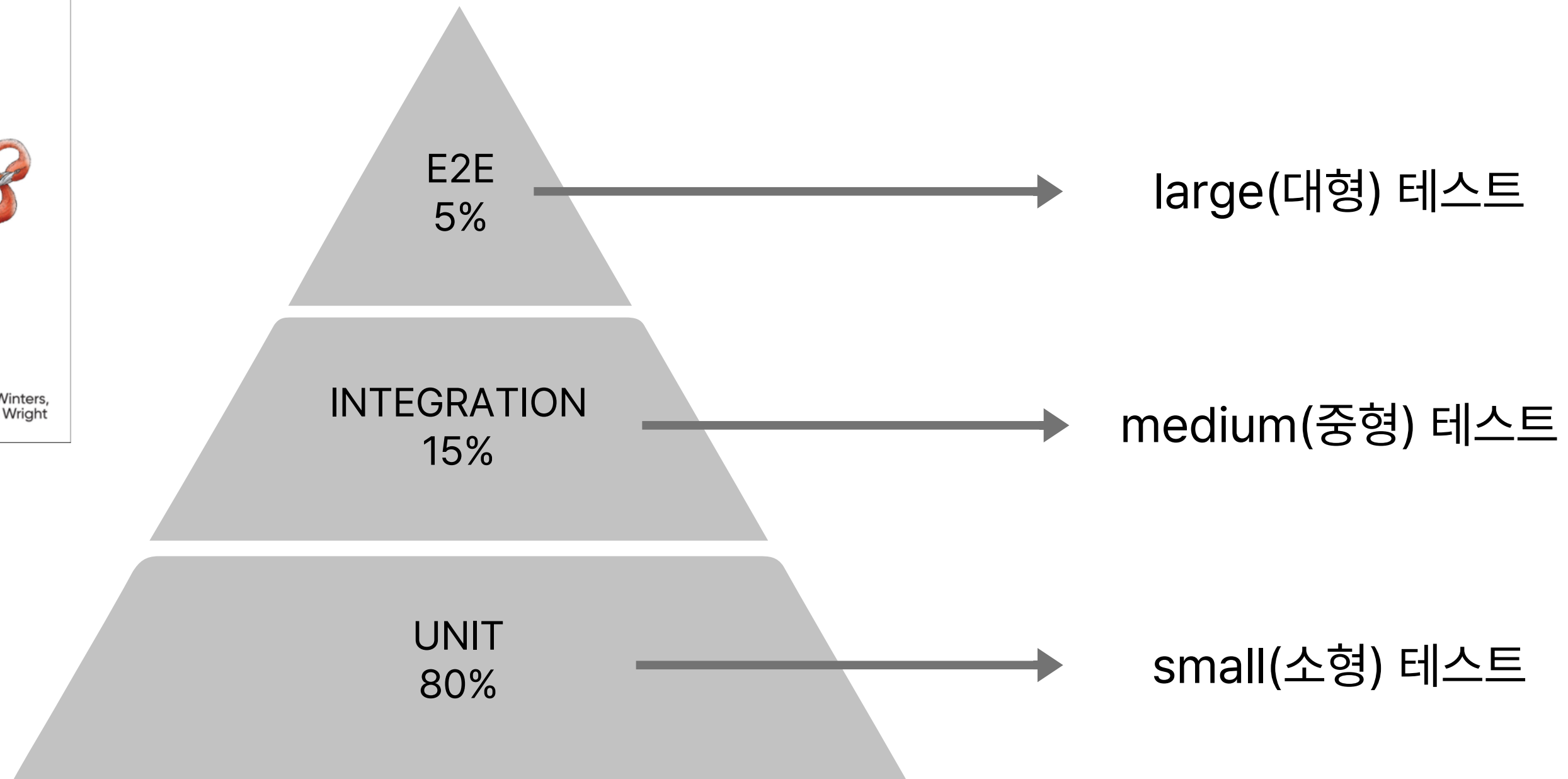
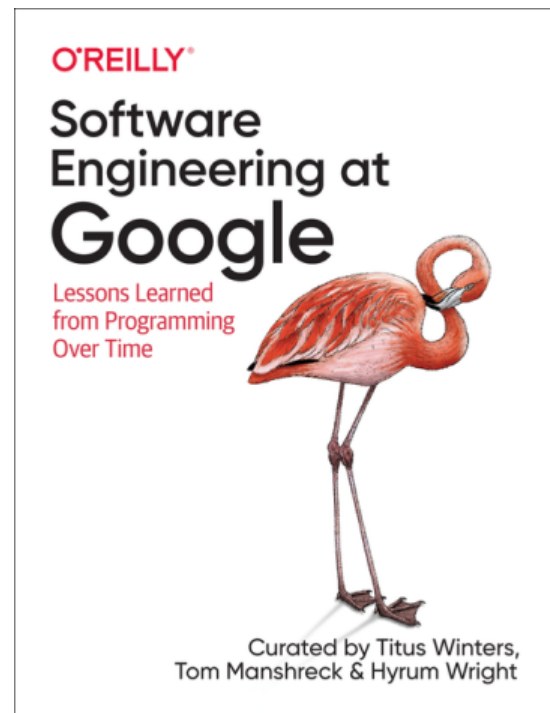
전통적인 테스트 3분류) 개인적으로 정의가 모호해서 안 좋아합니다.



2. 테스트 3분류

구글의 테스트 3분류

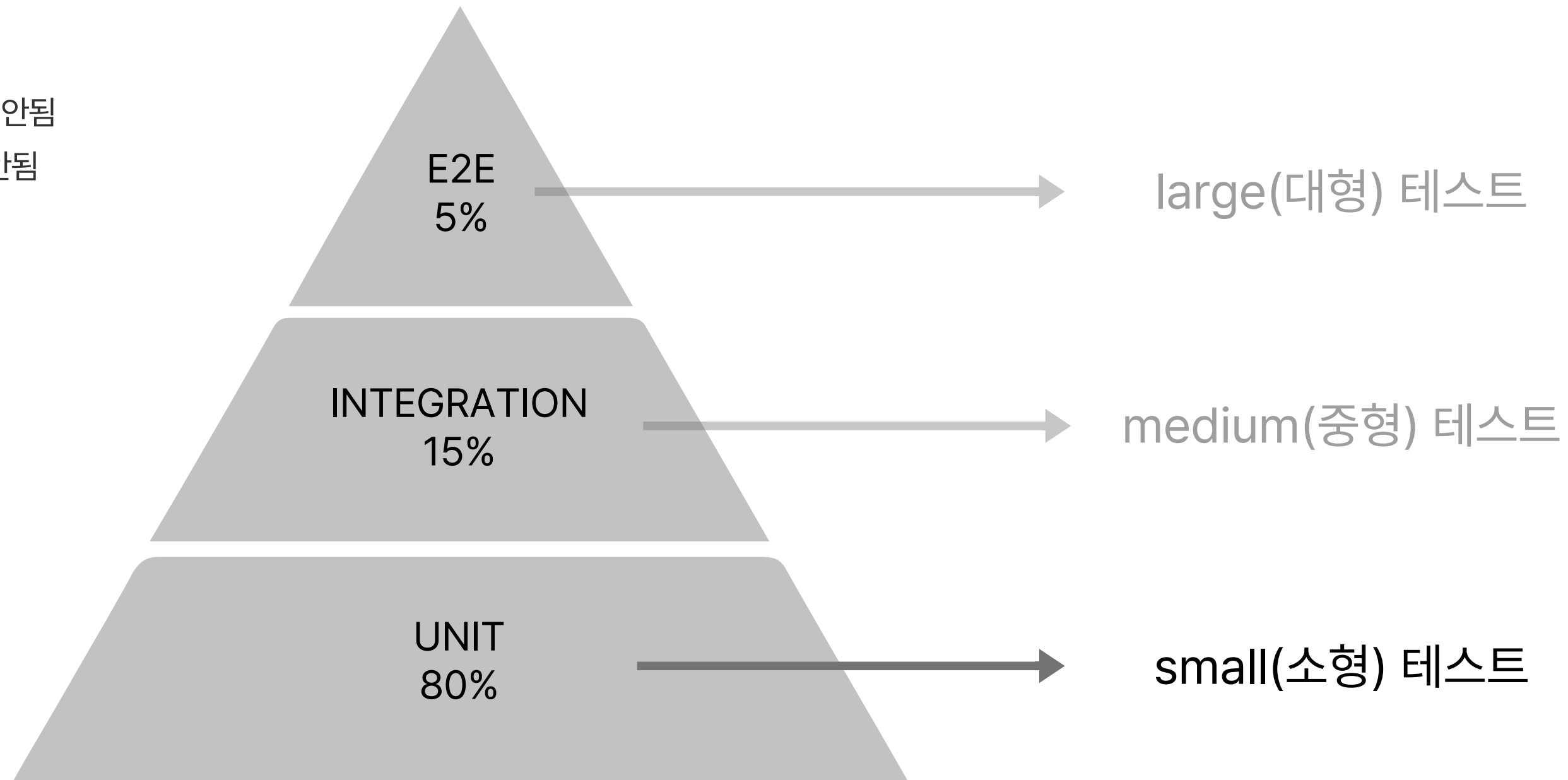
“ Curated by Titus Winters, Tom manshreck & Hyrum Wright, Software Engineering at Google: Lessons Learned from Programming Over Time, (O'Reilly Media, 2020-09-04)



2. 테스트 3분류

구글의 테스트 3분류

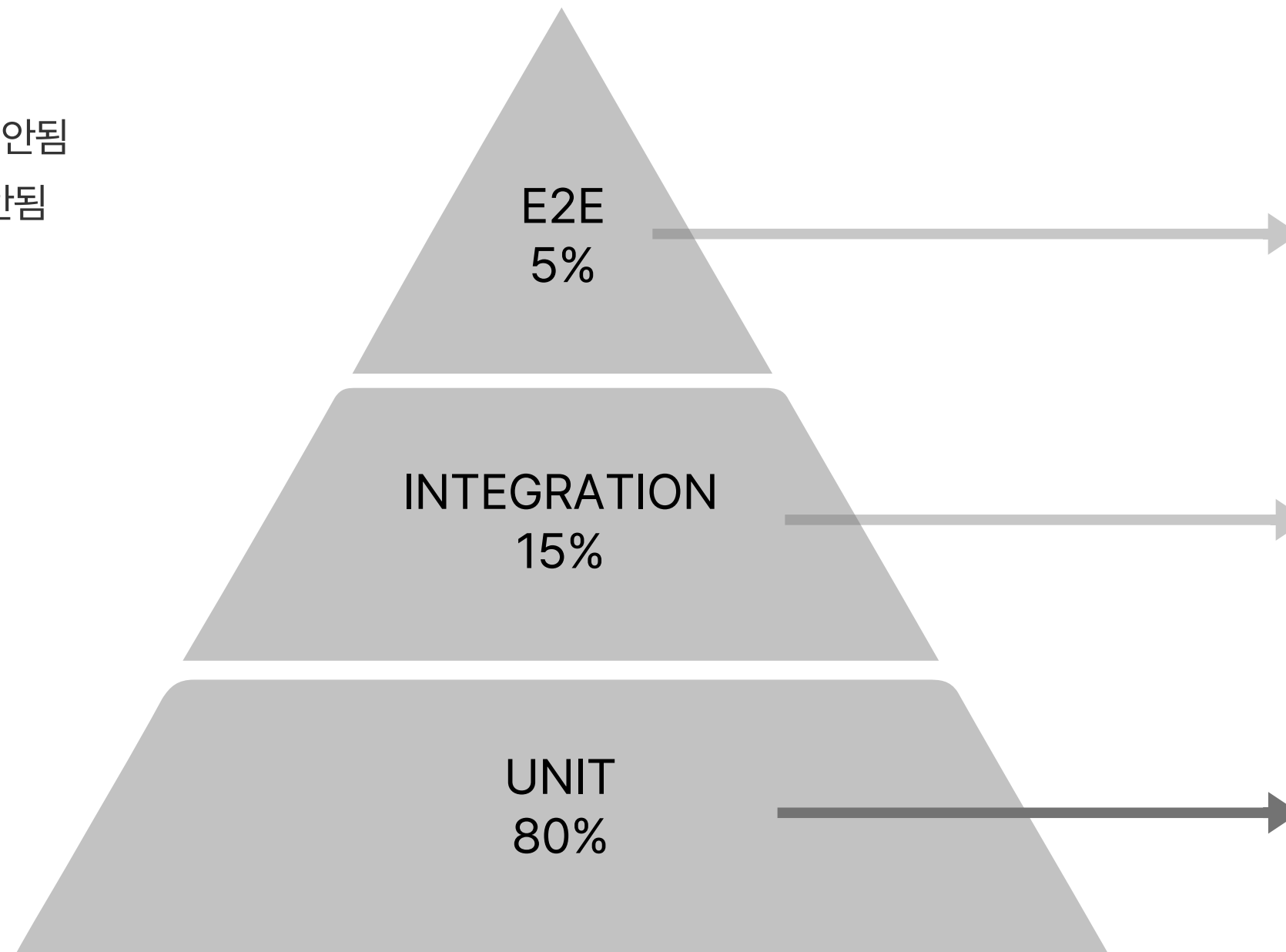
- 단일 서버
- 단일 프로세스
- 단일 스레드
- 디스크 I/O 사용해선 안됨
- Blocking call 허용 안됨



2. 테스트 3분류

구글의 테스트 3분류

- 단일 서버
- 단일 프로세스
- 단일 스레드
- 디스크 I/O 사용해선 안됨
- Blocking call 허용 안됨



```
class MathTest {  
  
    @Test  
    void Math_는_정수를_더해_정수를_얻을_수_있다() {  
        // given  
        int a = 10;  
        int b = 20;  
  
        // when  
        int result = Math.add(a, b);  
  
        // then  
        assertThat(result).isEqualTo(30);  
    }  
}
```

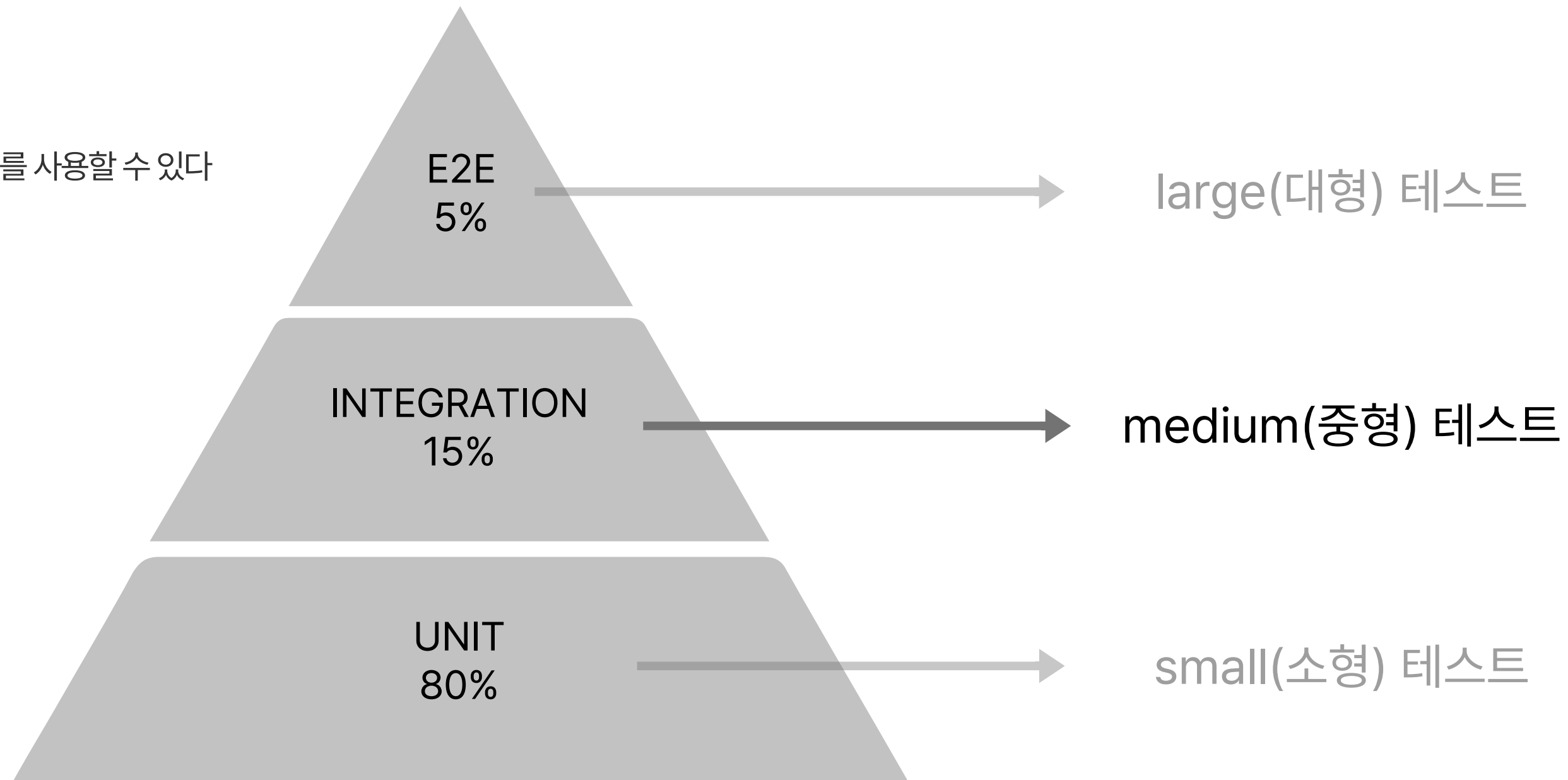
```
class Math {  
  
    public static int add(int a, int b) {  
        return a + b;  
    }  
}
```

2. 테스트 3분류

구글의 테스트 3분류

- 단일 서버
- 멀티 프로세스
- 멀티 스레드

→ h2같은 테스트 DB를 사용할 수 있다

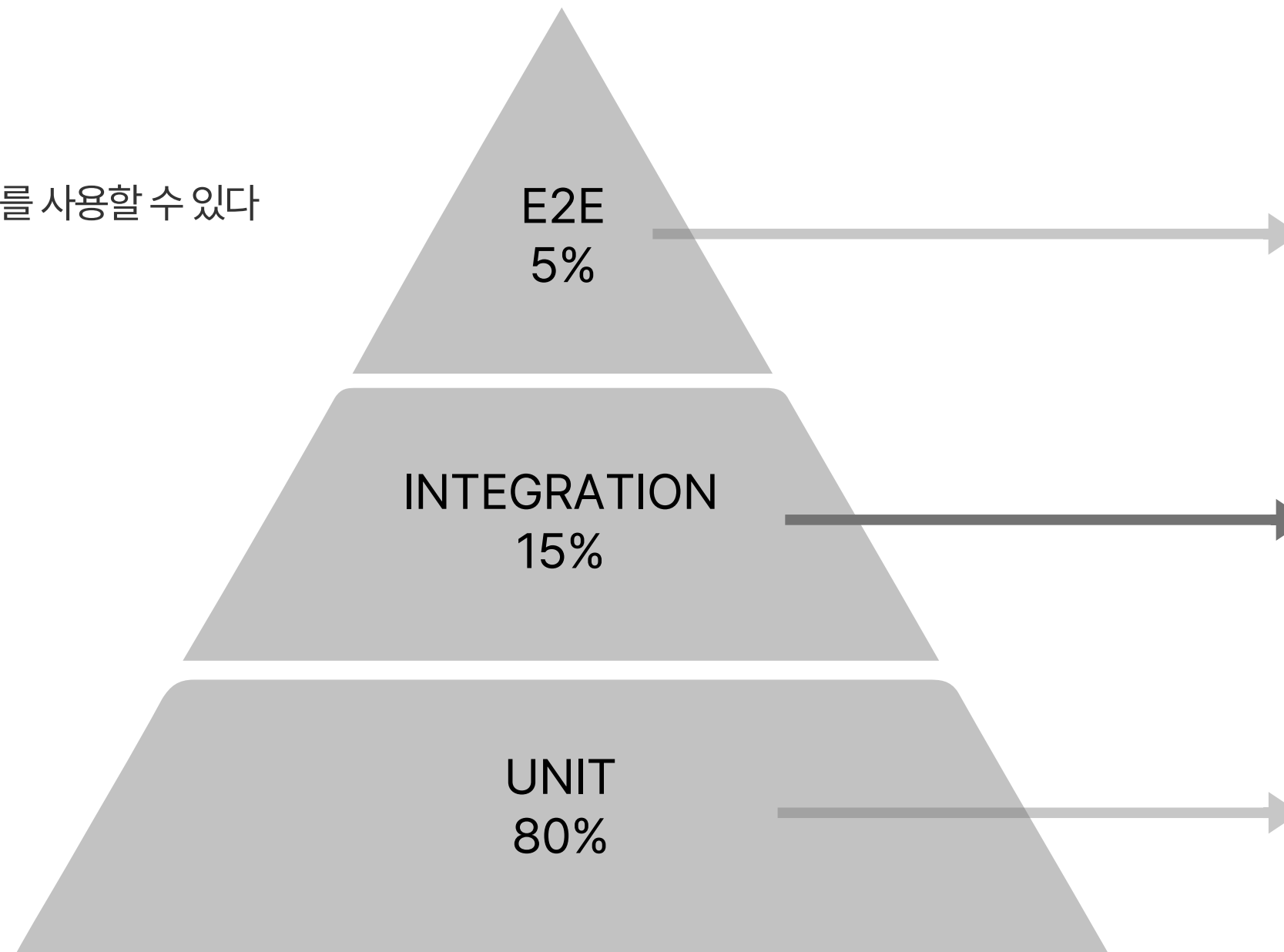


2. 테스트 3분류

구글의 테스트 3분류

- 단일 서버
- 멀티 프로세스
- 멀티 스레드

→ h2같은 테스트 DB를 사용할 수 있다



```
@ExtendWith(SpringExtension.class)
@DataJpaTest
@TestPropertySource("classpath:test-application.properties")
class UserRepositoryTest {

    @Autowired
    private UserRepository userRepository;

    @Test
    void 사용자_정보를_save로_저장_할_수_있다() {
        // given
        UserEntity userEntity = new UserEntity();
        userEntity.setEmail("kok202@naver.com");
        userEntity.setStatus(UserStatus.PENDING);
        userEntity.setCertificationCode("hash1234567890");
        userEntity.setAddress("Seoul");
        userEntity.setLastLoginAt(Clock.systemUTC().millis());

        // when
        userEntity = userRepository.save(userEntity);

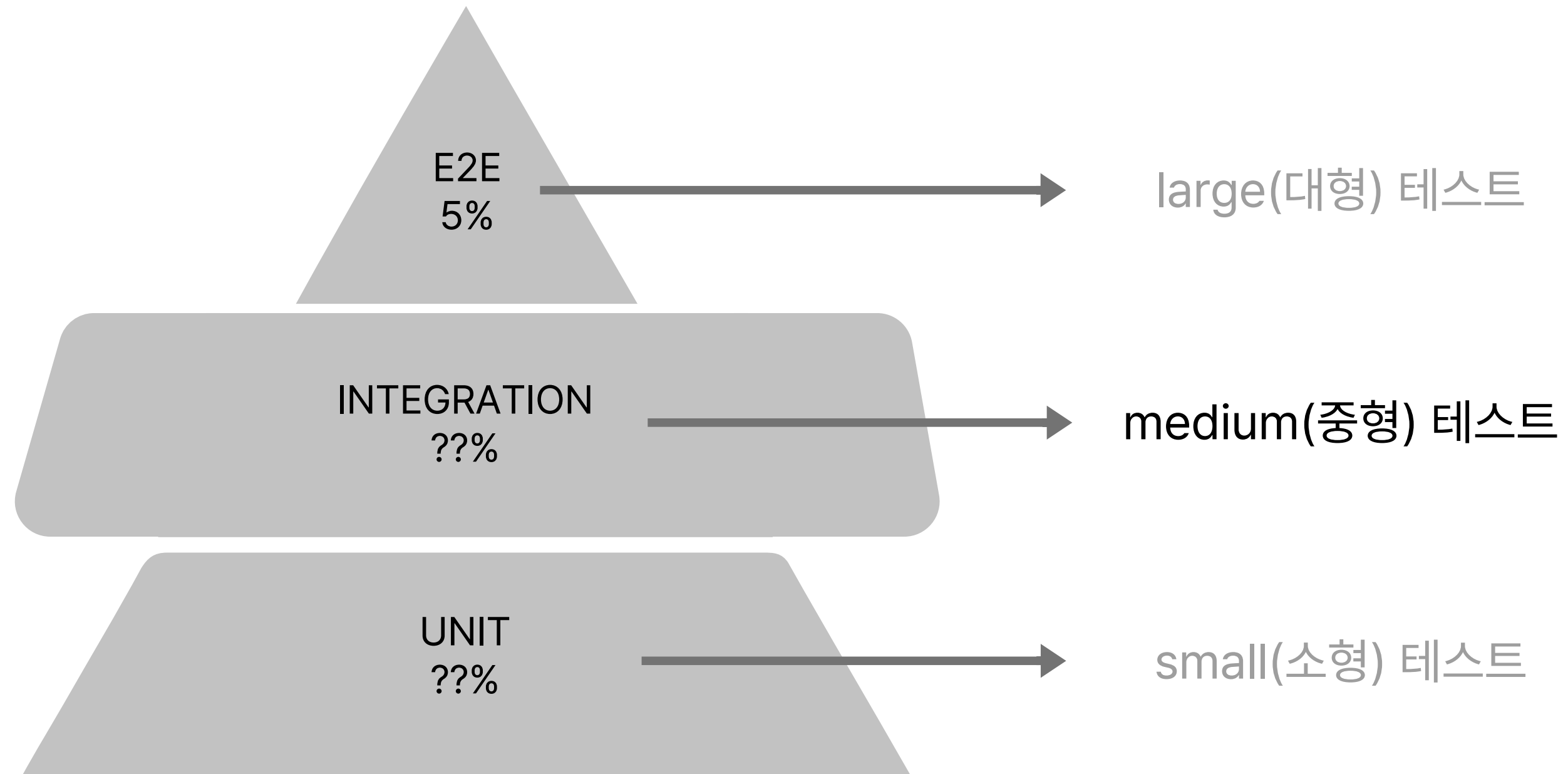
        // then
        assertThat(userEntity.getId()).isNotNull();
    }
}
```

```
public interface UserRepository extends JpaRepository<UserEntity, Long> {
}
```

2. 테스트 3분류

구글의 테스트 3분류

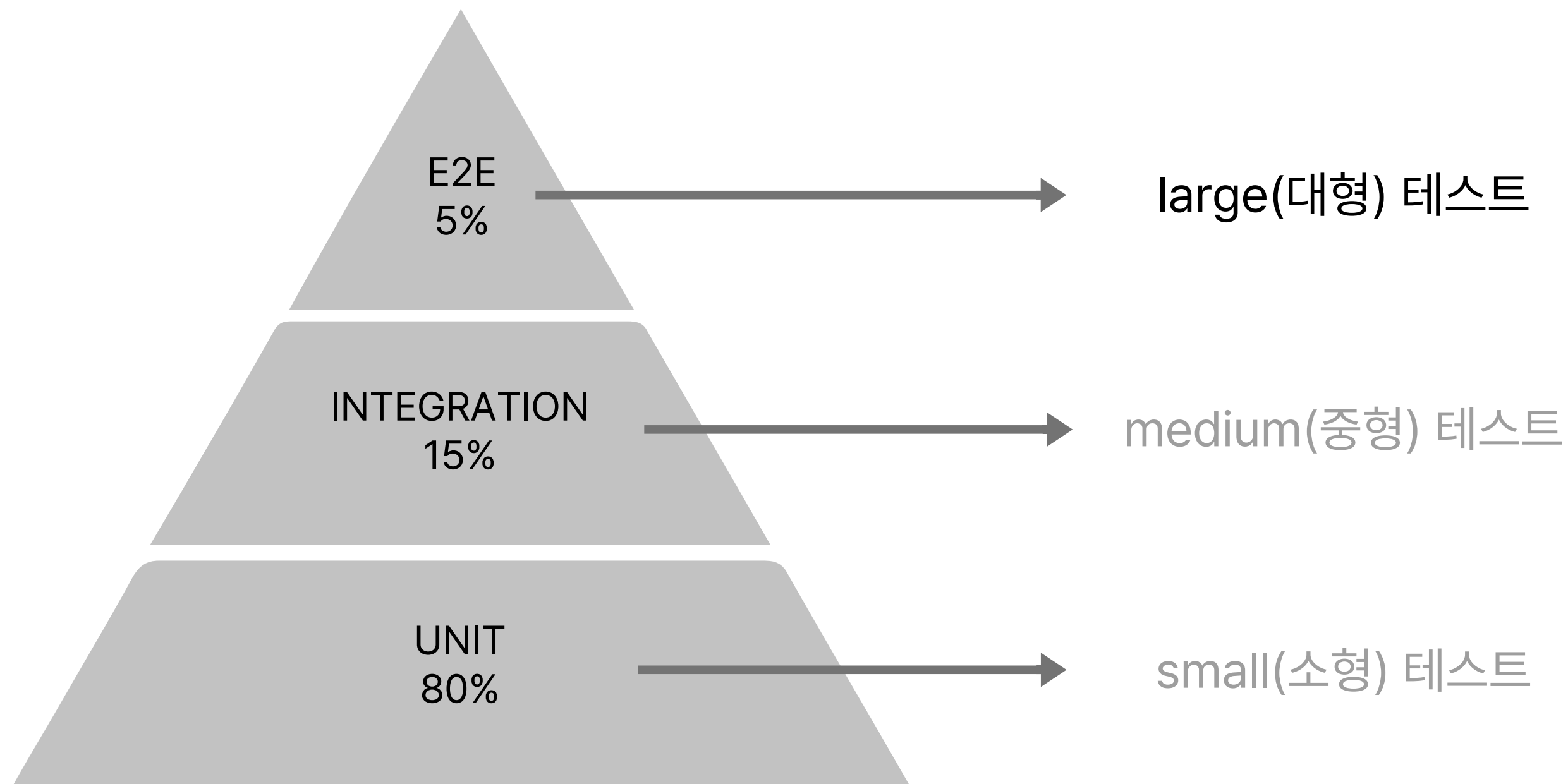
h2 같은 테스트용 DB를 사용하는 테스트 코드가 많은 경우



2. 테스트 3분류

구글의 테스트 3분류

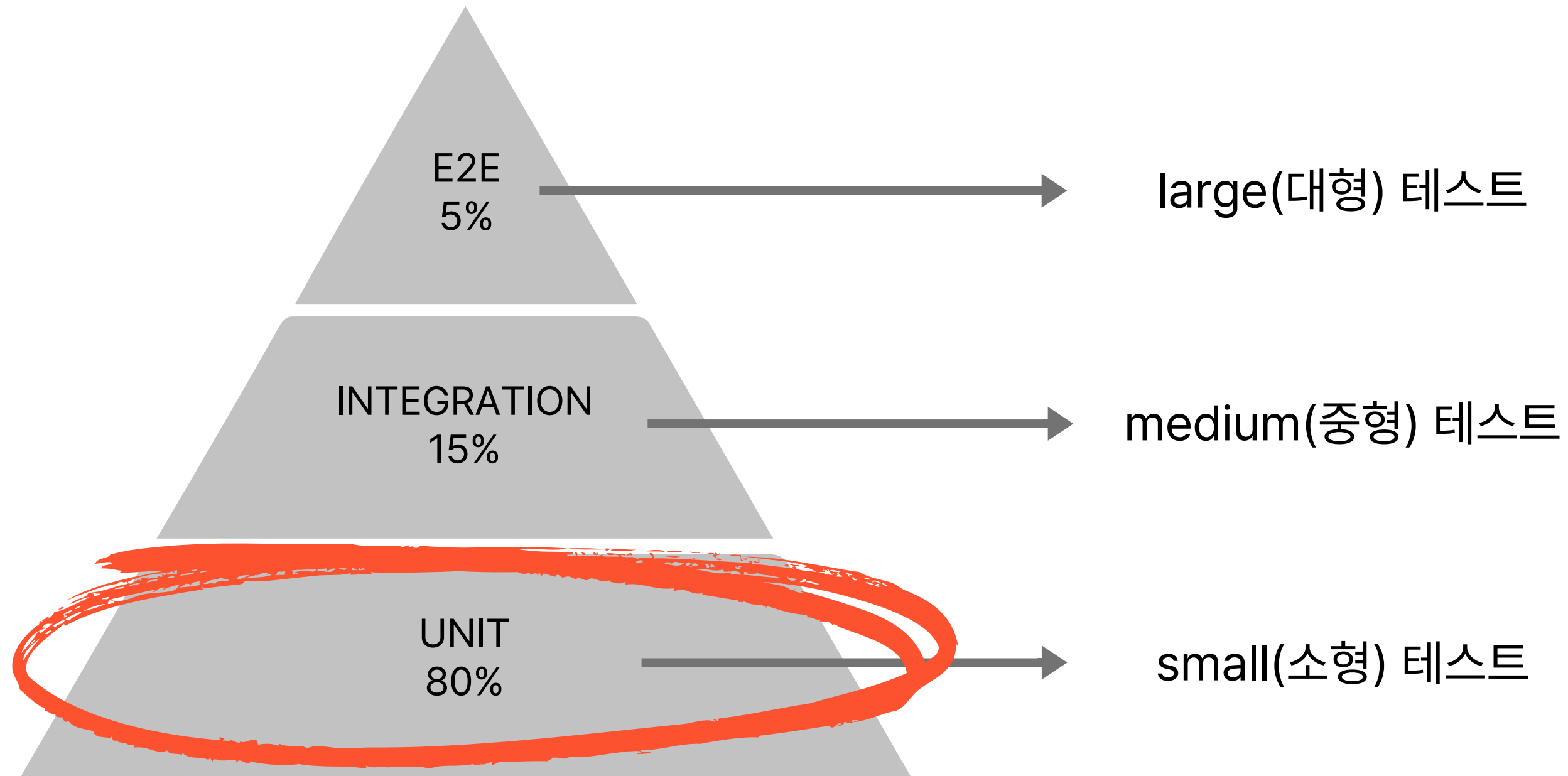
- 멀티 서버
- End to end 테스트



2. 테스트 3분류

구글의 테스트 3분류

당연하게도 우리가 집중해야 하는 것은 제일 많은 portion을 차지하는 소형 테스트입니다.



정리

- ☐ 테스트의 필요성에 대해 같이 살펴보았습니다
- ☐ 테스트와 좋은 아키텍처의 상관관계를 같이 살펴보았습니다
- ☐ 테스트 3분류에 대해 같이 살펴보았습니다



RETURN;