Part 9

테스트 방법론

시스템 개발 전과정에 걸친 테스트 방법을 알아본다.
코드리뷰와 같은 정적 테스트에서 부터, 단위 테스트 부하 테스트와 같은 시스템 테스트 계획 및 수행 방법과, 필요한 도구를 살펴본다.

소프트웨어 테스트

서버 백엔드 에 대한 검증을 위해서, 어떤 테스트 방법론과, 도구를 사용해야 하는지를 알아본다.

테스팅

서비스가 요구 사항대로 작동함을 확인하고, 잠재적인 문제를 발견하기 위한 단계

- 예전에는 개발 완료된 후, 테스트 단계에서 수행
- 현대에는 CI 단계에서 빌드마다 자동으로 테스트 수행이 추가됨
- 테스트의 범위에는 완료된 시스템의 검증 이외에, 법률적 요건과 기타 규제에 대한 검증과정을 포함함.

정적 테스트와 동적 테스트

정적 테스트

코드를 실행하지않고, 문서와 코드를 검토

개발 초기 단계에 잠재적인 문제점을 찾아낼 수 있음.
팀원간 지식 공유가 가능함 (교육)

- **코드 리뷰(Code Review):** 개발자들끼리 코드 내용을 검토하며, 오류나 개선점을 찾는다.
- **워크스루(Walkthrough):** 개발자가 코드 흐름을 설명하고, 팀원들이 리뷰하면서 문제점을 찾는 방식이다.
- **정적 분석 도구 사용:** 자동화된 도구(예: SonarQube, ESLint)로 코드의 잠재적 오류와 결함을 찾는다. 코딩 표준, 보안, 성능 문제도 검출 가능하다.
-

동적 테스트

코드를 실행하면서 정상 작동 여부를 검증

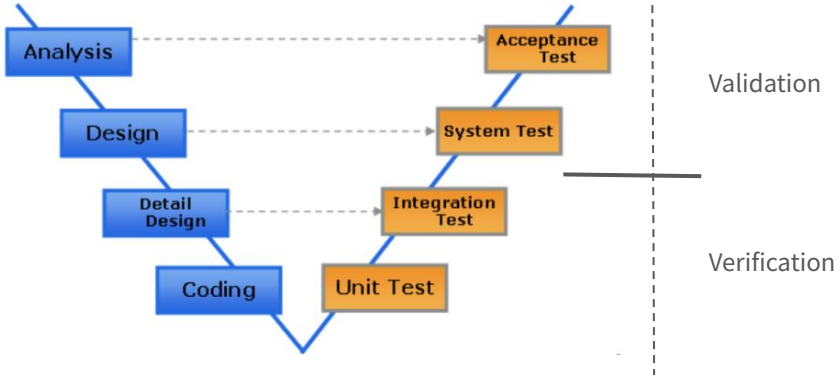
소프트웨어 실행중 발생할 수 있는 문제를 찾고, 설계된 형태로 제대로 작동하는지 검증. 기능/비기능 테스트로 분류

- **유닛 테스트(Unit Testing):** 개별 함수나 모듈의 기능을 검증하는 테스트.
- **통합 테스트(Integration Testing):** 여러 모듈이 상호작용할 때 예상대로 작동하는지 확인하는 테스트.
- **시스템 테스트(System Testing):** 전체 시스템의 기능이 요구사항에 맞게 작동하는지 테스트하는 방식.
- **회귀 테스트(Regression Testing):** 코드 수정 이후 기존 기능이 제대로 작동하는지 확인.

테스트 방법론 - V 모델

- 워터폴 모델의 확장편. ISTQB에서 소개
- 워터폴 모델 좌측의 프로세스에 맞춰서 각 단계별로 테스팅 단계를 추가

검증 (Verification)	확인 (Validation)
올바르게 만들고 있는가?	올바른것을 만들고 있는가?
주로 산출물을 검증	구현된 시스템을 확인 평가
정적 테스팅	동적 테스팅



동적 테스트 종류

동작 가능한 코드를 이용하여 테스트를 수행

단위 테스트	통합 테스트	시스템 테스트	인수 테스트
소프트웨어의 가장 작은 단위(함수, 메소드, 클래스 등)를 독립적으로 검증하는 테스트. 개별 모듈이 예상대로 작동하는지 확인	각 모듈을 결합하여 상호작용이 올바르게 이루어지는지를 확인하는 테스트. 모듈 간의 인터페이스, 데이터 흐름, 통신이 예상대로 작동하는지 확인.	전체 시스템이 요구사항에 맞게 작동하는지를 검증하는 테스트. 기능/비기능 테스트를 포함.	고객이나 최종 사용자가 소프트웨어가 비즈니스 요구사항을 충족하는지 확인하기 위해 수행하는 테스트. (계약 사항, 법률 사항등에 대한 검증 포함)

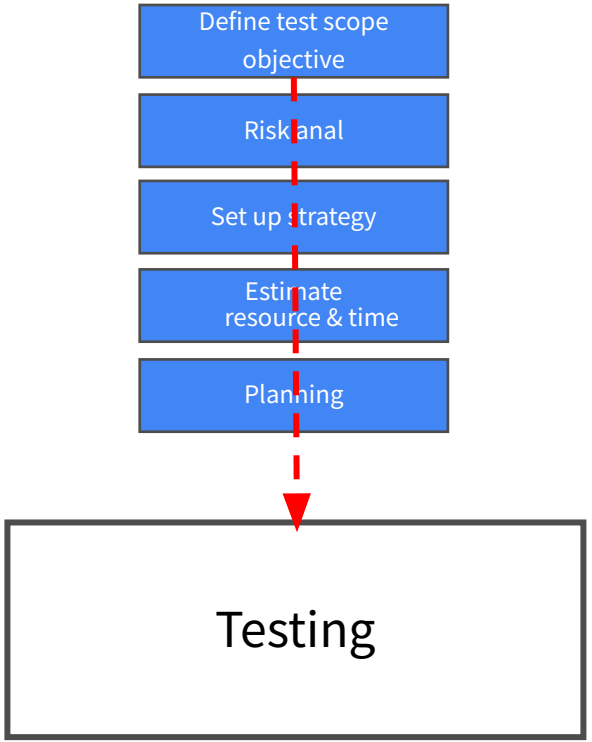
정적 테스트 종류

코드를 수행하지 않고, 코드 또는 기타 산출물을 리뷰하여 검증

01	코드리뷰, 웍스루	• 다른 개발자, 팀과 작성된 코드를 검토하여, 잠재적 문제를 파악하고 수정
02	정적 코드 분석	• 작성된 코드를 자동화된 도구를 통해서 잠재적인 문제를 파악
03	산출물 리뷰	• 각 개발단계별로 작성된 산출물 (아키텍쳐 명세서, 기능 명세서, API SPEC등)을 리

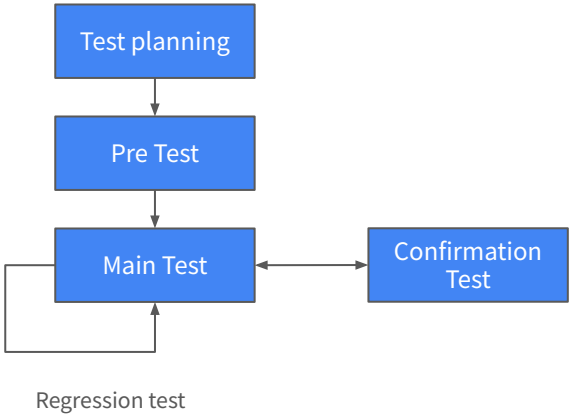
테스트 계획

- 테스트의 수준,범위 정의
- 위험도와 발생 가능성에 따른 테스트 우선 순위 지정
- 테스트 일정 (주요 마일 스톤), 리소스(테스트 환경, 비용, QA 엔지니어) 정의, 테스트 도구 선정
- 테스트 시나리오 작성,일정 수립
- 테스트 환경 구성



테스트 프로세스

- **테스트 계획** : 테스트 시나리오 작성 및 계획, 테스트 데이터 작성
- **예비(계통) 테스트** : 테스트가 가능한지 시나리오에 따라 기능이 제대로 작동하는지 검증
- **메인 테스트** : 메인 테스트
- **확인 테스트** : 문제가 발생하였을 경우, 문제를 해결한후, 제대로 해결 되었는지 확인
- **회귀 테스트** : 다른 기능 테스트시에는 이전에 테스트 했던 기능들을 모두 포함해서 테스트 함 (사이드 이펙트 확인)

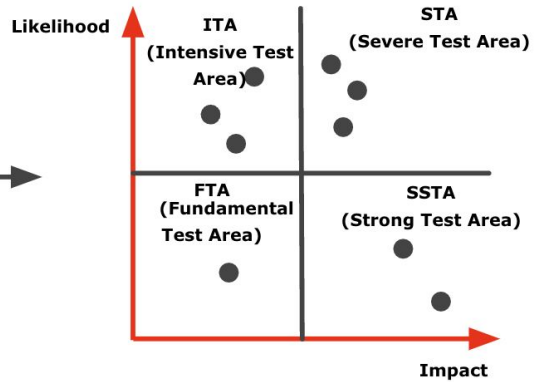


위험도 분석

- 위험도(Risk) = 발생 가능성 (Likelihood) * 파급 효과 (Impact)
- 36개 정도의 주요 분야로 구성
- 발생 가능성(Likelihood) : 코드 복잡도, 기술적 구현 난이도, 시스템의 크기, 개발팀의 역량
- 파급효과(Impact) : 비즈니스 손해

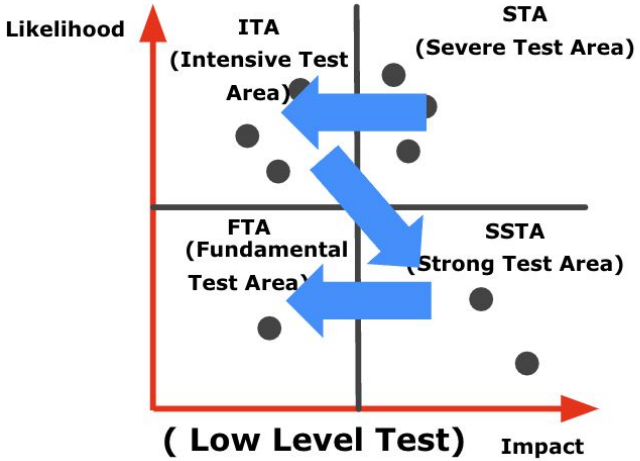
	Likelihood		Impact	
	factor	factor	factor	factor
Risk item				
Risk item				

Erik van neneendaal, Risk Based Testing, STAREAST,2006

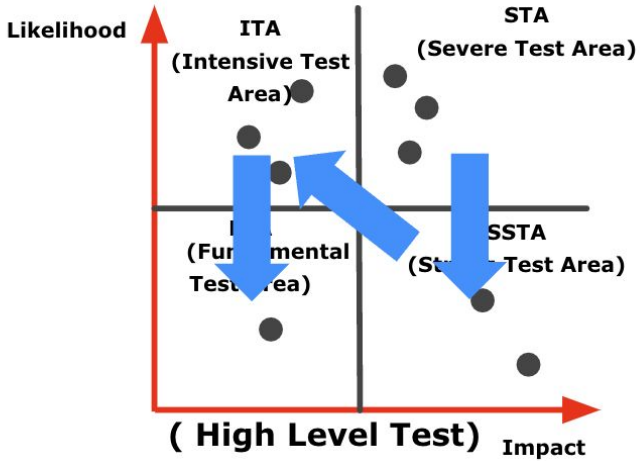


우선 순위 설정

ISTQB Risk based testing strategy (위험도 기반 테스트 전략)



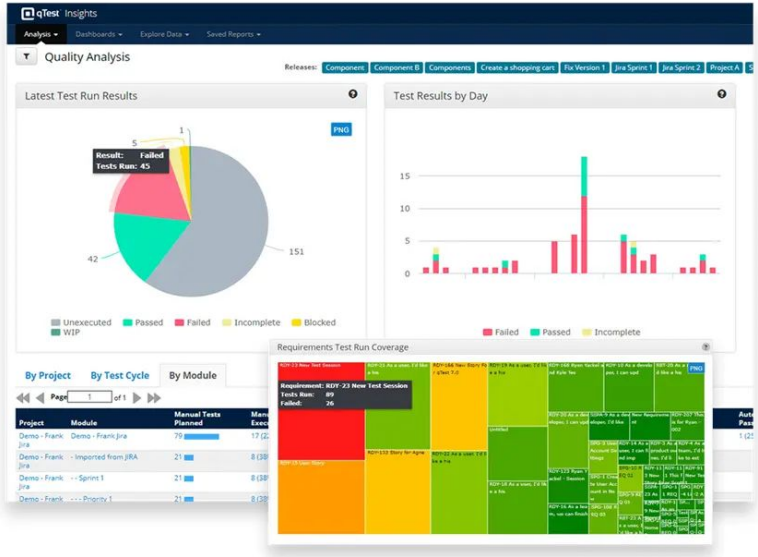
단위 테스트, 통합 테스트
자주 발생할 가능성이 높은 케이스 우선으로 테스트. (시스템이 잘 작동함을 확인하기 위함)



시스템 테스트, 인수 테스트
비즈니스 임팩트가 큰것을 우선으로 테스트. (심각한 비즈니스 임팩트를 사전에 방지하기 위함)

테스트 케이스 관리 도구

- 테스트 케이스 관리 도구 : 테스트 케이스를 효율적으로 생성,저장,실행, 추적 관리하기 위한 소프트웨어.
- 이슈 트래킹 시스템 연동, 커버리지 리포트 제공
- vs 액셀 : 수동 관리시 중복 및 누락, 불일치, 팀원간의 테스트 케이스 공유가 어렵고, 진행/커버리지 관리가 어려움
- 도구 : qTest, Test Rails,Browser Stack (테스트 자동화 가능)



qTest : 출처
https://www.tricentis.com/products/unified-test-management-qtest

테스트 수행

- **테스트 수행**
  - 테스트 케이스 작성, 테스트 데이터 생성
  - 정해진 시나리오와 할당된 도구를 이용하여 테스트 수행
- **리포팅**
  - 테스트 결과 리포팅
  - 테스팅 시간, 결과(성공/실패), 테스트 로그 수집
- **결함 추적**
  - 실패케이스의 경우 테스트 케이스와 연결하여 bug ticket 생성
  - 수정후 회귀 테스트를 통한 재 검증

테스트 방법론 - 탐색적 테스팅

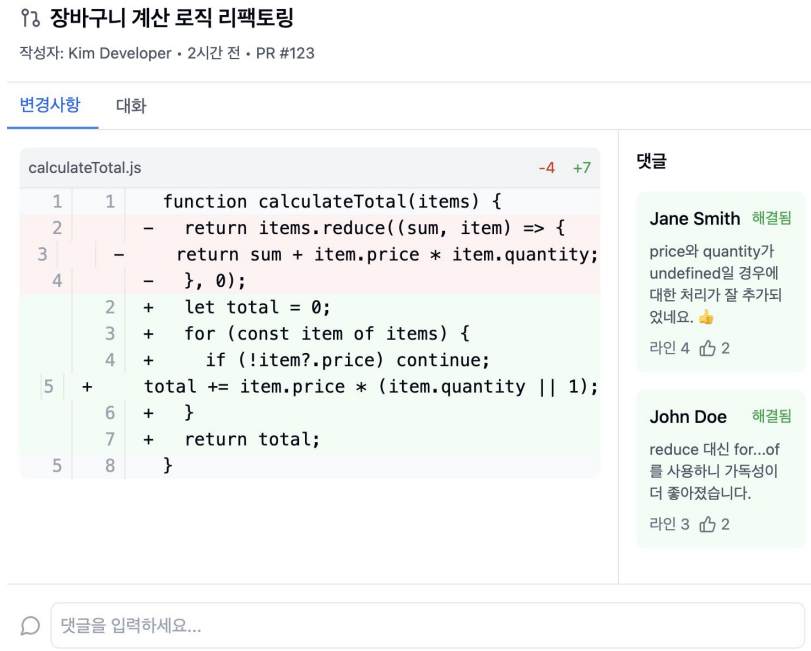
- 탐색적 테스팅 : 사전에 정의된 테스트 시나리오 없이, 테스터가 기능과 동작을 탐색하면서 수행하는 테스트 기법 (테스터의 역량에 의존함)
- 언제
  - 초기 단계에서 제품에 대한 이해가 필요한 경우
  - 명확한 요구 사항 문서가 없을때
  - 테스트 시나리오 이외의 부분을 테스팅 할때
- 문서화
  - 테스트 계획이 없기 때문에, 수행한 절차와 결함에 대한 문서화 누락이 될 수 있음 (결함 보고서, 스크린샷등이 필수)
- 세션 기반 테스팅 (Session based testing)
  - 탐색적 테스팅의 보완 방법
  - 시간 제한을 두고 테스팅 (60~120분)
  - 세션에서 수행할 테스트의 목표를 정의 (“로그인 기능에서 다양한 사용자의 역할 테스트")

정적 테스팅

코드 리뷰

코드리뷰는 팀원들이 서로 코드를 검토하고 개선하는 활동으로, 코드에 대한 품질 향상과 잠재적인 오류를 없앨 수 있으며, 팀원에 대한 교육 목적으로 사용할 수 있다.

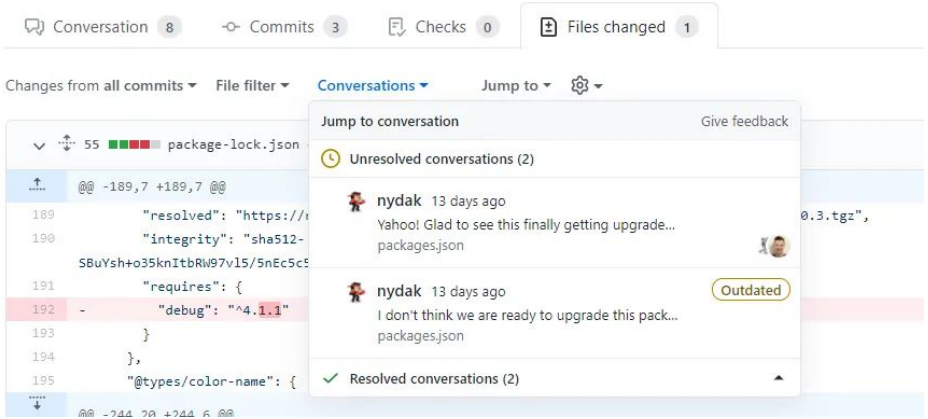
1. **코드 제출**: 개발자는 변경된 코드를 저장소에 Push하고 Pull Request(PR) 또는 Merge Request(MR)를 생성하여 리뷰를 요청한다.
2. **리뷰어 지정**: 코드 작성자가 아닌 다른 팀원이 리뷰어로 지정된다.
3. **코드 검토**: 리뷰어는 코드의 로직, 성능, 스타일을 확인하며 필요한 피드백을 남긴다.
4. **피드백 반영**: 작성자는 리뷰어의 피드백을 반영하여 코드를 수정하고, 필요 시 추가 리뷰를 요청한다.
5. **병합**: 리뷰어가 최종 검토 후 승인을 하면, PR/MR을 병합하여 코드베이스에 반영한다.



코드 리뷰 도구

코드리뷰 도구는 형상관리 도구, CI/CD 도구와 통합되어가는 추세

- GitHub Pull Request
- GitLab Merge Request
- BitBucket
- Rhode Code
- JetBrain Space
- AWS CodeCommit
- Azure Devops



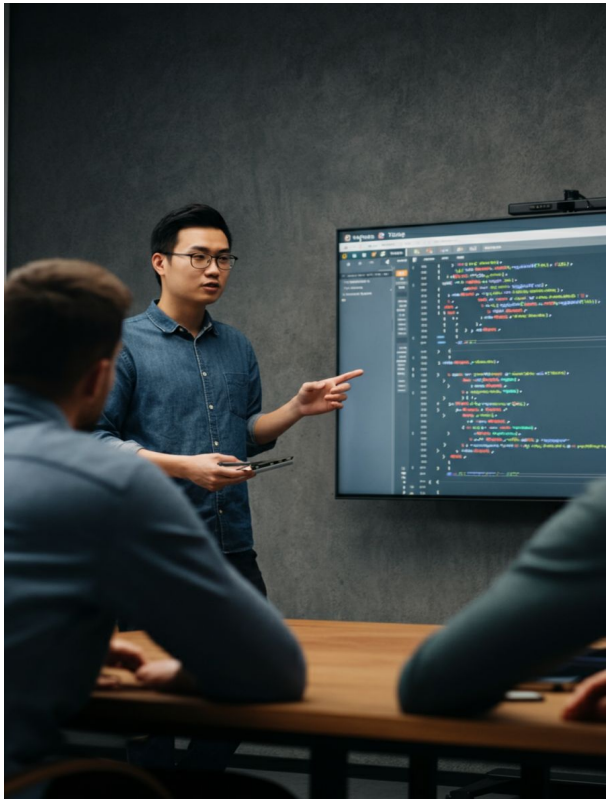
출처 :
https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/about-pull-request-reviews

코드 워크쓰루 (Walkthrough)

코드리뷰가 도구 기반, 비동기 1:1 이라면 워크쓰루는 대면미팅으로 1:N 팀 단위로 이루어진다.

- 발표자가 자신의 코드를 흐름,목적,로직을 팀원들에게 설명한다.
- 팀원들은 개선 방안을 제안한다. 이 과정에서 코드를 변경하지 않고 의견을 나누는데 집중한다.
- 피드백을 중심으로 작성자는 코드를 수정한다.

표준화와, 팀의 지식 공유에 효과적임



정적 코드 분석

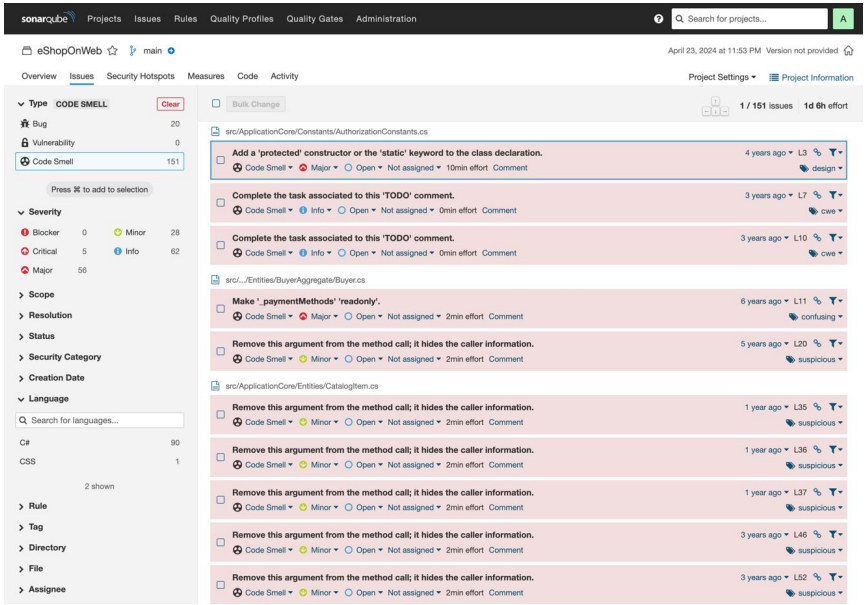
자동화된 도구로, 코드를 검사한다.

- 코드의 표준 준수 여부
- 보안상 위험 요소
- 오픈소스 라이선스 위반 여부

주요 도구

- SonarQube
- Fortify Static Code Analyzer (보안 취약점)

CI 단계에 통합하는 경우가 많음



출처 :
https://www.sonarsource.com/learn/static-code-analysis-using-sonarqube/

동적 테스팅

단위 테스트

소프트웨어의 가장 작은 단위(함수, 메소드, 클래스 등)를 독립적으로 검증

- 주로 개발자가 수행하며, 테스트 자동화 도구(JUnit, NUnit 등)를 사용해 개별적인 기능의 정확성을 확인.
- 테스트 커버리지 검증.
- 외부 의존성(데이터베이스, API 등)을 Mock이나 Stub으로 대체해 테스트 환경을 독립적으로 유지.

테스트 커버리지 리포트

통합 테스트

소프트웨어 모듈 뿐만 아니라 미들 웨어(DB) 등, 모듈을 결합하여 상호작용이 올바르게 이루어지는지를 확인하는 테스트

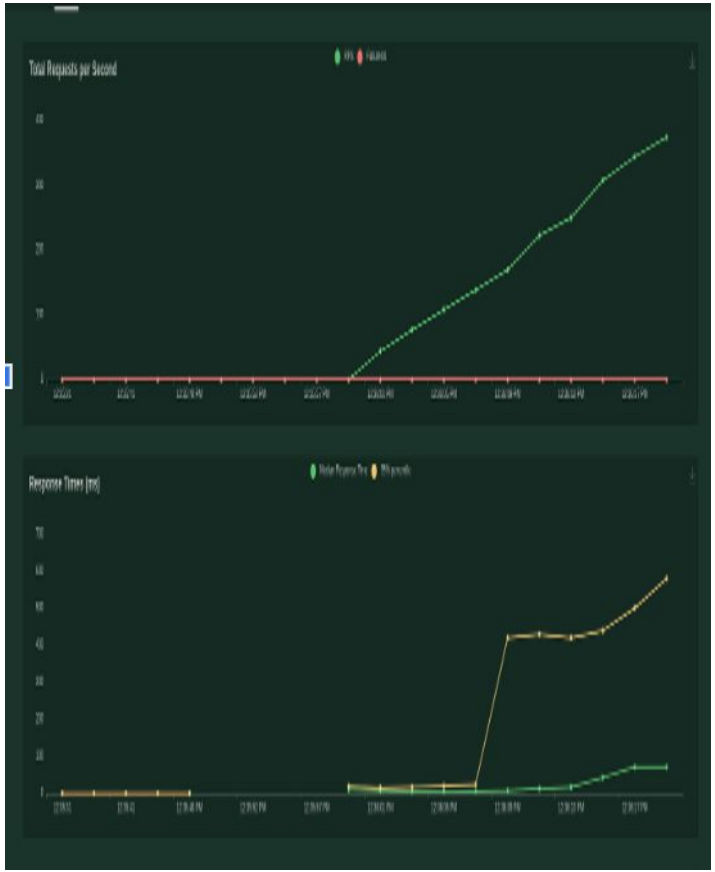
- 예전 개발 모델에서는 Big Bang 방식을 사용하였으나, 현대 개발에서는 CI로 인하여 필요성이 많이 줄어듬
- 스테이징 환경 구축시 마이크로 서비스 컴포넌트간의 상호 작용을 검증하기 위해서 여전히 필요함
- 시스템 테스트전 **개통 단계**로 많이 사용됨

시스템 테스트

기능/비기능 테스트를 포함한다.

비기능 테스트

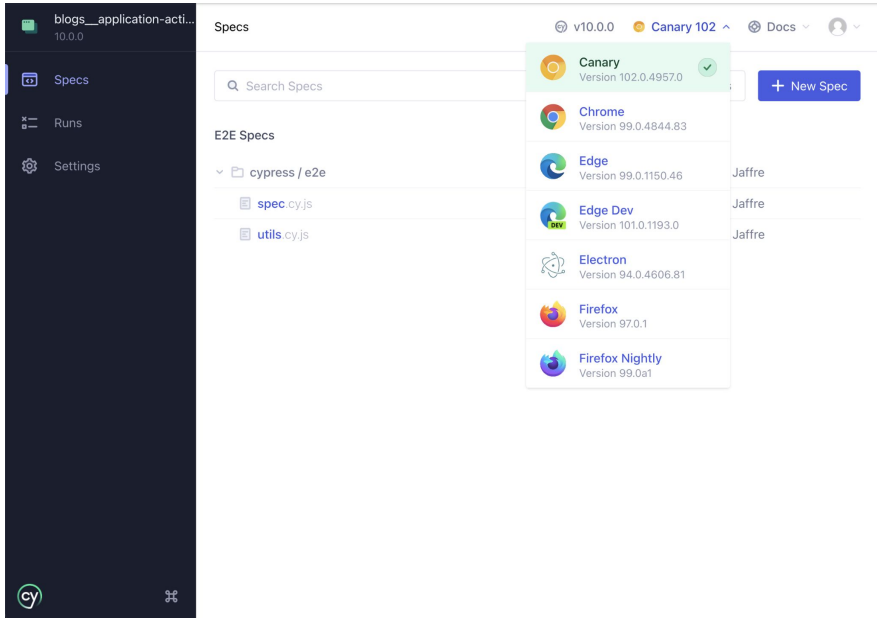
- 성능 테스트 : 시스템의 성능 및 한계 성능 측정
- 장애 테스트 : 장애시 대응 능력 측정
- 안정성 테스트 : 오랜 시간 가동해도 문제가 없는지 측정



시스템 테스트 - 웹 기능 테스트

웹 테스트

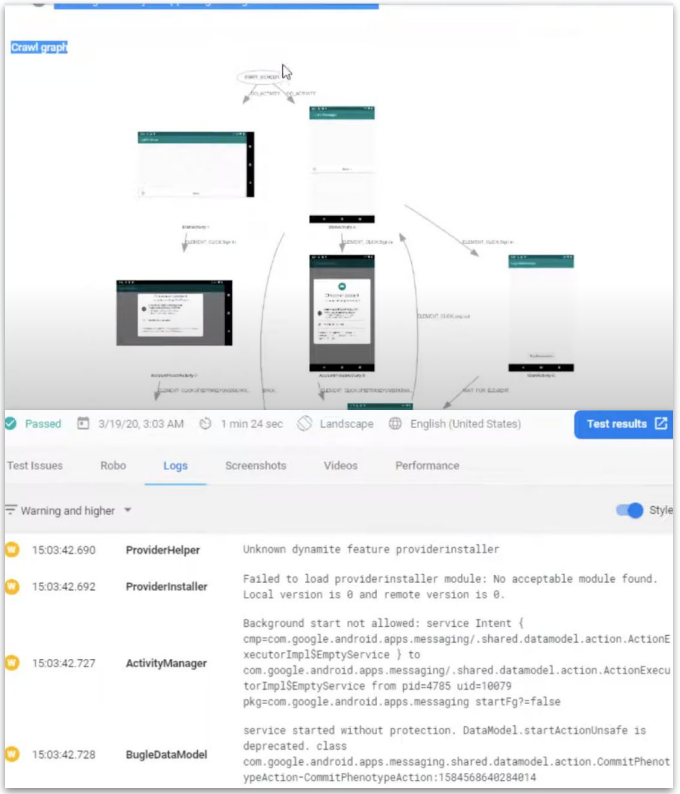
- 실제 웹 브라우저를 자동으로 수행해서 테스트를 실행
- Selenium : OSS, (Java,Python,Javascript, Ruby 지원)
- Cypress : 통합 테스팅 플랫폼



시스템 테스트 - 모바일 기능 테스트

모바일 애플리케이션에 대한 자동화 테스트

- 로보트를 이용해서 테스트
- 실제 디바이스 환경에서 테스트
- 테스트 스크립트를 수행하고, 수행 과정을 비디오로 녹화, 애플리케이션 프로파일, 성능 자료 및 발열등을 측정
- Firebase TestLabs



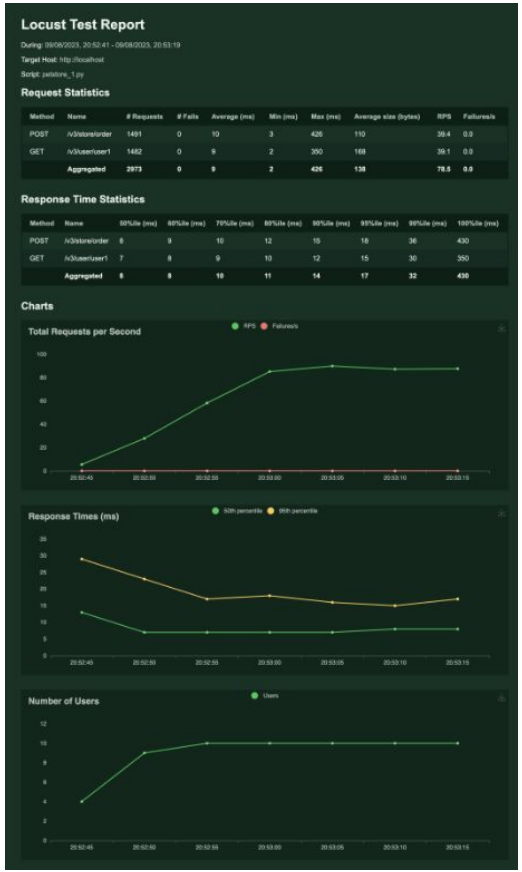
시스템 테스트 - 성능 테스트

성능 테스트

- 부하 상태에서 시스템의 성능과 한계 용량을 측정
- 애플리케이션 병목을 찾기 좋음
- 모니터링 상태에서 테스트
- 단일 테스트 : 특정 기능(API)만 테스트
- 복합 테스트 : 여러 기능들을 실제 호출

1. User log in : 5%
2. List contents : 30%
3. Select contents : 20%
4. View contents : 20%
5. Add comment : 25%

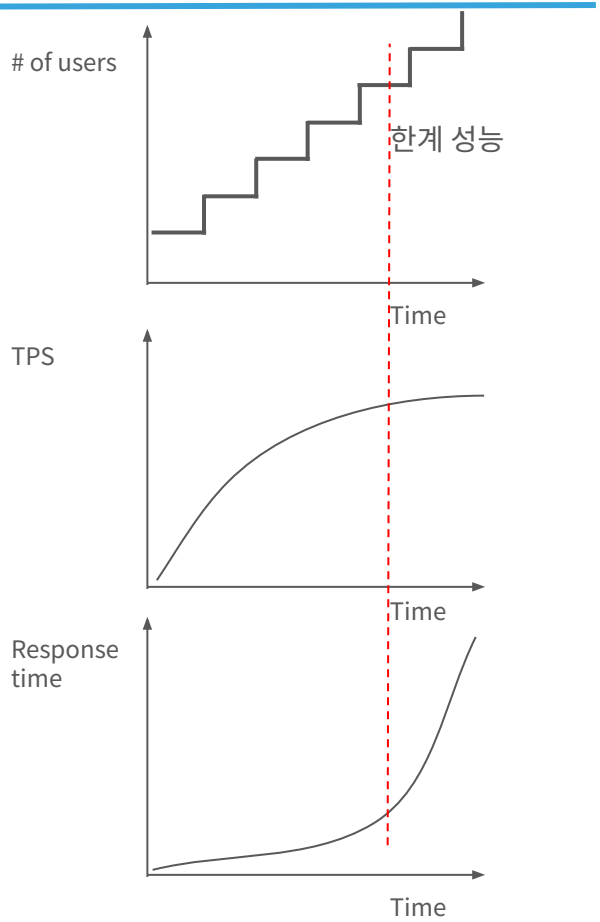
테스트 도구 : Apache Jmeter, Locust.io



시스템 테스트 - AB 테스트

성능 테스트

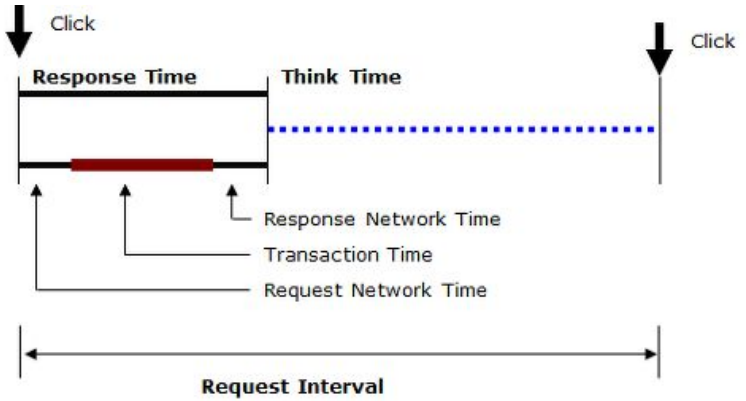
- 사용자를 계단식으로 시간에 따라 증가 시킴
- 한계 성능
  - 초당 응답량 (Throughput per second)가 완만하게 증가하는 구간
  - 응답 시간이 급격하게 증가 하는 구간
- 서버의 CPU 사용량, 특히 DB의 성능을 잘 모니터링 해야함.



TPS를 이용한 서비스 용량 산정

호출 모델

- Response Time : 사용자가 서버에 요청을 시작해서 최종 응답을 받은 시간
- Network Time : 서버에 요청을 보내고 받을때 소요되는 네트워크 시간
- Transaction Time : 실제 서버에서 소요된 시간
- Think Time : 사용자가 응답을 받은 후, 응답을 이해하는 시간 (웹페이지를 읽는 시간)



TPS를 이용한 서비스 용량 산정

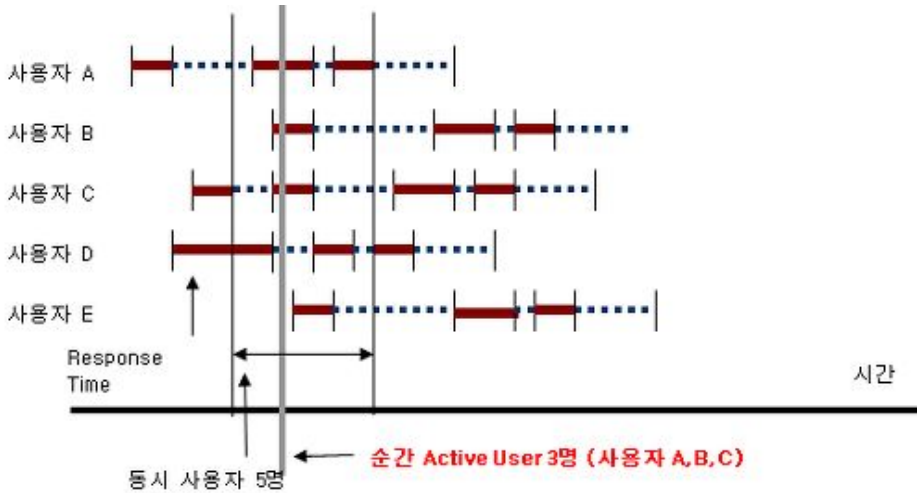
동시 사용자 : 현재 접속되어 있는 사용자

- 동시 사용자(Concurrent User) : 현재 접속되어 있는 사용자
- 활성 사용자 (Active User):현재 트랜잭션을 실행중인 사용자
- TPS = (Active User)/(Average Response Time)
- TPS = (Concurrent User)/(Request Interval)

즉 TPS를 성능테스트에서 구할 수 있으면, Request Interval을 가정하여, Concurrent User (시스템의 최대 동접자)를 산정할 수 있음

예) TPS = 5000 이고, Request Interval = 5초

동접자는 TPS * Request Interval = 25000명



시스템 테스트 - 장애 테스트

시스템이 장애에 대한 복구성이 있는지 테스트
예시

- API 서버 다운
- DB 다운시 페일오버

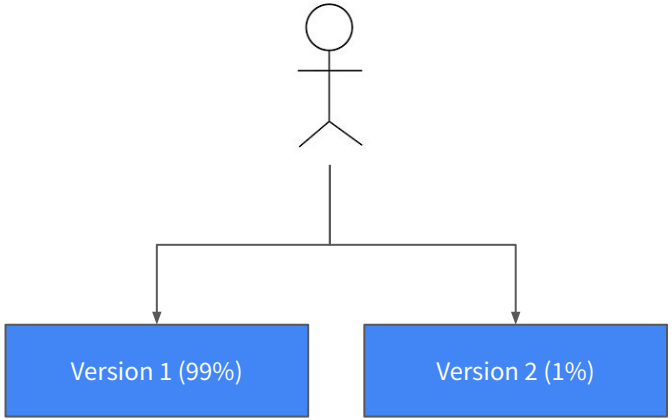
근래에는 시스템이 커지고 클라우드 매니지드 서비스를 사용하기 때문에,
점차 하지 않는 추세

카나리 테스트

새로운 기능을 테스트 할때 실
운영환경에서, 실 사용자중 일부에게
새기능을 오픈하여 검증하는 방식

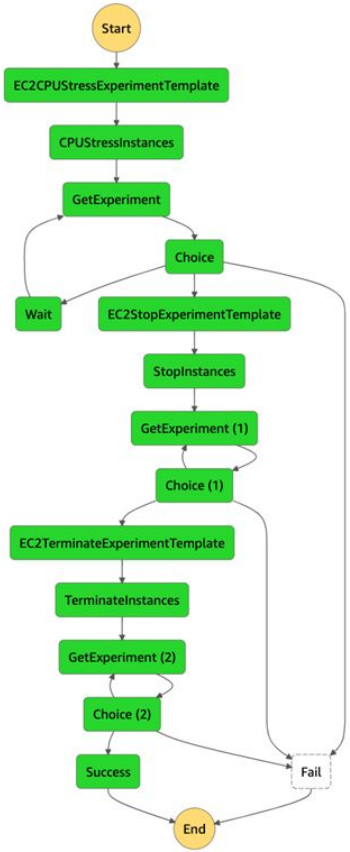
- API 트래픽 분할
- 클라이언트를 일부만 배포하는 방식

※ 복잡한 시스템에서 테스트가 어려운 경우
효과적으로 사용 가능



카오스 엔지니어링

- 대안으로 카오스 엔지니어링 사용 (넷플릭스 카오스 몽키, 그렘린,AWS FIS)
- 시스템 회복성 (Resiliency) 증대
- AWS FIS
  - AWS 서비스에 대한 장애를 생성
  - 스크립트에 정의된 시나리오에 따라 수행
  - CPU 과부하, EC2 정지, EKS Worker 정지, 네트워크 트래픽 생성,DB failover 생성등



출처 : aws.amazon.com

보안 테스트

01	취약성 테스트 (Vulnerability scanning)	• 보안 취약점 스캐닝 (OS 패치 버전이나, 애플리케이션 라이브러리, 미들웨어등의 알려진 보안 취약점 스캔) • 쿠버네티스 컨테이너 스캐닝 • Soos.io , Aws inspector
02	침입 테스트 (Pen testing)	• 외부로 부터의 사이버 어택에 대한 시뮬레이션 • SQL Injection, Cross site scripting • Cobalt, HackerOne (솔루션 보다는 업체를 통한 테스트 추천)
03	인증 인가 (Authentication, authorization testing)	• 패스워드 로그인 및 권한에 대한 해킹 가능 여부 테스트 • Hashcat (OSS GPU 사용 가속),
04	컴플라이언트 (Compliance Test)	• 법률 규정에 맞는지를 테스트 • 한국 ISMS 인증, 미국 SOC 2, GDPR, PCI DSS 등

인수 테스트

시스템 납품 (출시)전, 사용자의 요구 사항에 만족하는지 확인하는단계

- 알파 테스트 : 내부 사용자나 고객 일부를 대상으로 테스트
- 베타 테스트 : 제품이 거의 완성된 상태에서 실제 사용자를 통해 검증하는 테스트로 피드백을 수집

감사합니다.