강의 교재

https://drive.google.com/drive/folders/1zjTtqnGOAdbxMga7nWi8uLElBnnuUATa

강의 자료

https://mybox.naver.com/share/list?shareKey=qNiQyfILf0d0SVD5vQikmKl6T3DvuLwt53Ym0utoFjQD

- 김윤수 강사님

- 개발자

- 클라우드에 개발자

- 보안/품질 관리

- 책집필: ISMS, 모의 해킹/취약점, 보안 테스트

- 01086384839

- kchris000@gmail.com

! SW테스트와 품질\_ 김윤수

! 단위 테스트 - 인프라/QA 부서와의 협업 및 자동화

================1일차================

- 셀프 피드백

[TDD의 목적]

* 소프트웨어 개발(품질, 테스트)를 위해
* TDD, Test 자동화

[강의 수행]

* 파이썬 기반으로 테스트
* 파이Test
* 코드는 도구와 제품의 방향성에 따라 달라질 수 있다.
* 이미지 매칭 기반
* 스크립트를 통한 자동화

[TDD가 유용할 때]

* 요구사항이 급변할 때

[개발 고민 요소]

* 개발하고, 테스트한 그 코드가 서버에 어떻게 배포되었음을 보장하지?  
  🡺 기능단위를 도커로 묶어서 이미지 배포
* 클라우드 인프라와 운영서버 인프라가 같다는 것을 어떻게 보장할 거야?  
  🡺 테라폼 도입
* 로그 파일이 너무 많이 쌓여서 죽어버렸는데 분석도구는 어떻게 할 거야?  
  🡺 문제 생기면, 알려주는 체제 도입
* MSA, Cloud, Docker 이 상황에서 품질확보 및 언제든 서비스 할 수 있다는 환경을 어떻게 보장할 것인가?  
  🡺

[개발보안진단 도구-보안적인 이슈가 있는지 확인]

* sonar-scanner

! 전체적으로 요구사항을 어떻게 충족하고, 안정성을 어떻게 확보하냐

[테스트 종류]

* 단위 테스트
* 시스템 테스트: 스트레스 테스트(성능 테스트)
* 모듈과 모듈간 테스트

! 파이썬, 3.11버전부터 엔진이 바뀌어 성능이 좋아짐

! SikuliX – 이미지 매칭 테스트

! 지향점

* 요구사항 분석
* 결과에 대한 검증 방안에 대해 고민
* 상황과 환경이 바뀌었을 때 대응 🡺 컨테이너 단위로 배포 🡺 배포하기 전 테스트 수행

첫날 순서

1. 인프라 설정
2. 협업 요소
3. 소프트웨어 공학(요구사항, 개발 방향, 생산성-품질확보)

\* 환경 변수: 운영체제가 참조하는 글로벌 변수

! 환경변수 역할: 간접 참조

ex)

(win)

SET MYENV=HELLO

echo %MYENV%

(linux)

export, $MYENV

\* 환경변수 등록

- 윈도우버튼+R

- sysdm.cplr

! 설정한 다음 재부팅을 해줘야 함

JDK 환경 변수 잡아주기

‘JAVA\_HOME, PATH’

C언어 레벨에서 TDD 하는 법은 다름

\* 가상 머신 만들고 활용

- 개발하다 보면, 운영환경에 준한 가상머신(HOST)가 필요

- 테스트를 위해서 환경 구축

\* 윈도우에서 리눅스를 사용하는 방법

- 클라우드의 리눅스 접속

- VM 환경 구축

- 컨테이너 구축

- WSL2, 도커 컨테이너

- HyperVisor : 가상머신 만들어서 사용 가능

실습: 리눅스 환경(운영이라고 가정) 구성

* 공유기(NAT-ip 주소 변경)
* 포트포워딩

! 후배들한테 강조하는 것, 계정명이 한글이 들어갈 때, 몇몇은 인지를 못함.

\* 개발환경 구성시 꼭참고

- 한글 X(계정명)

- 파일 확장자 보기 (활성화)

- 파일과 경로는 띄어쓰기 및 한글을 적지 않는다.

! PC에 Network 카드 두개 이상 꽂을 수 있음 (총4개까지 꽂을 수 있음)

id: root/pw: team1234

[centos 네트워크 설정]

* 고정IP 할당

[ssh접속]

* ssh 계정@ip {필요시 -i 옵션 넣고 파일 지정}

[리눅스 간단히 설명]

* 배포판:   
  + 레드햇(RedHat, CentOS, Rocky…)  
  => yum install httpd  
  + 데비안(Ubuntu, Kali Linux, RPI OS)  
  => apt(apt-get) install apache2

[서비스 실행]

# systemctl start/enable/status/stop/disable httpd

[방화벽]

# firewall-cmd --permanent --add-port=80/tcp

# firewall-cmd --reload

요구사항 정의(what이 중요)

- 챗지피티

- 코파일럿

현재 수행중인 프로세시

* netstat -antp
* curl , telnet  
  \* curl: 웹서버의 API를 테스트 할 수 있다. ‘-X’ 옵션으로 HTTP Method를 지정할 수 있다.  
  \* telnet
* GPT – Python이 처음 학습한 언어임

\* 고객이 원하는 것을 표현하는 것은 어렵다.

\* 무엇을 원하는 것을 정하는 것이 어렵다.

[요구사항]

* 기능적 요구사항  
  ex) ID와 PW로 로그인
* 품질적 제약사항  
  ex) 500명이 동시에 로그인을 시도해도 0.5초 이내 응답
* 보안적 요구사항  
  ex) ID를 고정하고 PW 반복 입력하면 문제가 될 수 있으니, 방어해라.

\* 요구사항 점검 기준  
- 완전성: 모든 요구사항이 빠짐없이 도출되어야 한다.

- 명확성

- 구체적: 무엇을 구현해야 하는지 구체적으로 명시해야 한다.

! 오류 부재의 궤변

! 요구사항에 따른 시스템 스펙을 정한다

! ip bandwidth

! 아키텍쳐: 어떠한 제약사항과 원칙에 따라 배치할지 구조적으로 해결한다.  
  
! 확장성 / 가용성 / 재사용성

! 소프트웨어 아키텍쳐 설계 원칙

- 모듈화 => 재사용성

리펙토링 관점: 재사용적인 측면에서 괜찮은 코드인가?

! 모듈: 재사용 단위.

* 인터페이스(약속)을 잘 지켜주는 것.

1. 결합도: 2개 이상의 모듈간에는 느슨한 결합이여야 한다.  
   🡺 직접적으로 제어하면 안된다.  
   🡺 최소한의 파라미터를 이용하여 함수를 호출하도록 한다.
2. 응집도: 하나의 모듈 내에는 필요한 기능만 모여 있어야 한다.

* 기능에 심플한 모듈로 만들어야 하며 응집도가 낮아야한다.

! monolithic VS microservice

\* API GateWay

\* Monitoring

\* 조직의 문화와 환경에 따라 MSA 구조가 나뉜다.

! 소프트웨어는 체계적으로 관리해주지 않으면 문제가 발생할 수 있다.

[테스트]

* Grade , Quality
* 소프트웨어의 잠재적 결함을 사전에 발견하여 대응한다.

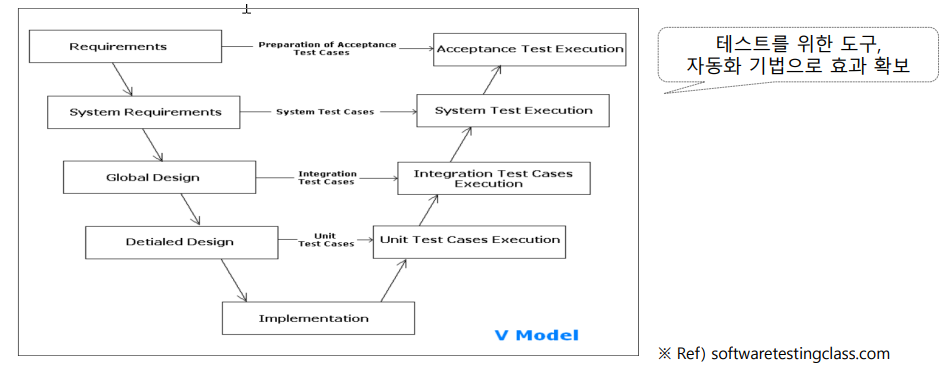
! 요구사항 오류: 고객 요구사항의 불완전한 반영, 잘못된 정의 및 의사소통, 고의 누락, 고객 승인 없는 기능 추가

! Defect Clustering

[문제가 많이 생기는 경우]

* 연계가 많은 소프트웨어
* 기술적 이해도가 낮은 상태에서의 개발(신입, 신기술 도입)

\* V-Model : 테스트 레벨

- 

! [테스트 기법]

ex) 학사관리 프로그램 제작

* 어떻게 테스트하지?  
  ! 어썰션 확인
* 무효한 형식을 입력한다. **– Negative Test** (‘A’,’Hello’) 무효한 구간

🡺 등가 집합.(동등 분할 테스트) – **Positive Test** : 유효한 형식, 구간 에 유효한 데이터를 넣어둠  
  
- 변경되는 구간에 대한 테스트

🡺 **경계값 분석 테스트**

🡺 **Pair Wise Test** (조합 테스트): 상태의 경우의 수에 따라 테스트

|  |
| --- |
| 명세 기반 테스트 |
| ▶ 등가분할(Equivalence Partitioning)  ▶ 분류 트리 기법(Classification Tree Method)  ▶ 경계 값 분석(Boundary Value Analysis)  ▶ 상태 전이 테스팅(State Transition Testing)  ▶ 결정 테이블 테스팅(Decision Table Testing)  ▶ 원인-결과 분석(Cause-Effect Graphing)  ▶ 조합 테스트 기법(Combinatorial Test Techniques)  ▶ 시나리오 테스팅(Scenario Testing)  ▶ 오류 추정(Error Guessing) |

|  |
| --- |
| 경험기반 테스트 |
|  |

**오늘의 핵심: 테스트는 요구사항의 일치여부를 확인하는 과정**

**구현을 테스트하는 것이 아닌, 요구사항을 테스트하는 것이다.**

***\**** Test Test harness

[결정테스트]

* 조건(if)에 참과 거짓이 한번씩은 테스트가 되어야 해., 개별 조건, 전체 조건
* 결정 커버리지, 조건 커버리지

! 검색해보기 MCDC 레벨 테스트

에자일은 선언. – 기민하게 변화

XP – 실행지침

* 메타포
* 코딩스텐다드
* 페어프로그래밍
* 40시간 초과 근무
* 리펙토링

! 용기, 고객의 요구사항을 수용할 수 있는 용기, 평소에 TDD를 문화로 가진 습관

\* 내가 만든 테스트를 통과하게 하는, 심플한 코드

jest, supertest

junit

pytest

|  |
| --- |
| 자격증:   * 면허증의 속성을 갖는 자격증 1) 정보시스템 감리원: IT 7년 이상(1주일 교육, 시험) 증 (~75살까지) 🡺 1달 풀타임 4-500만원 🡺 제안서 쓰면, 500정도 받음 2) 감리사, 기술사: 감리 + 감리 총괄 3) 개발보안진단원  감리원이면서 개발보안진단원이면 우대받음.   개발 + 무엇 = 시너지 |

\* Daily Meeting, 칸반 🡺 배포계획은 단계적으로 고객에게 어떤 가치를 제공할 것인가?

\* 지속적인 통합과 지속적인 배포

Subversion: 중앙관리, 인터넷끊어지면 관리 어려움, 구조 심플

- Python 설치 3.11.2 (3.11x버전부터, 엔진이 C++로 개발되어, 성능개선이 됨)

- virtualenv 설치 + 환경 설정

- VS Code 설치

- 플러그인 설치 : **Python, Python Extension**

- 터미널 설정 (디폴트 터미널 command 창으로)

- 가상환경 만들기 + activate/deactivate

**\*** [**위키독스 (wikidocs.net)**](https://wikidocs.net/)

**(** [**https://wikidocs.net/**](https://wikidocs.net/) **) 🡺 Jump To Python**

**\*** [**https://hey.technonia.com/**](https://hey.technonia.com/)

\* shutdown -h now

python 자료형

* Tuple : Read Only List