

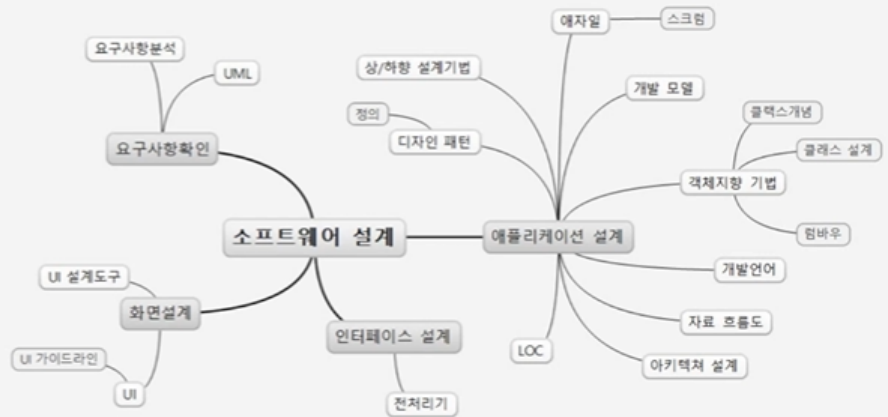
## 정처기 강의 요약

정보처리기사

두목 오빠

# 1과목 소프트웨어 설계

최대폭보 | 1과목



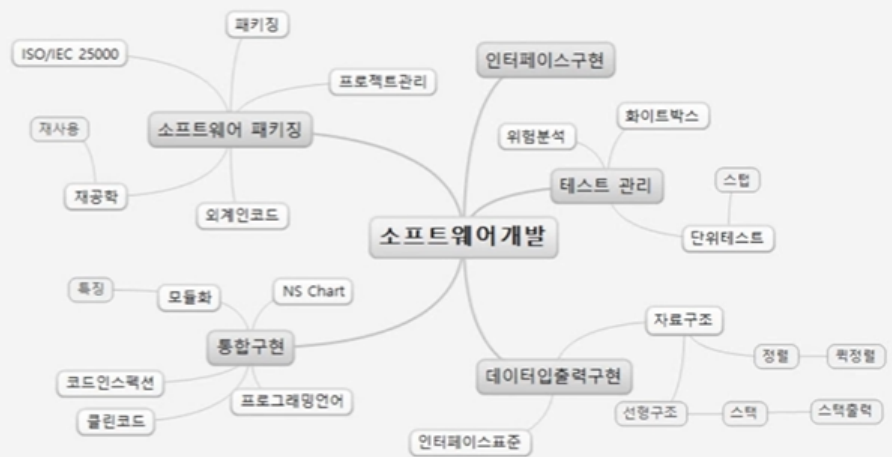
이거전

정보처리기사

두목 오빠

# 2과목 소프트웨어 개발

최대폭보 | 2과목

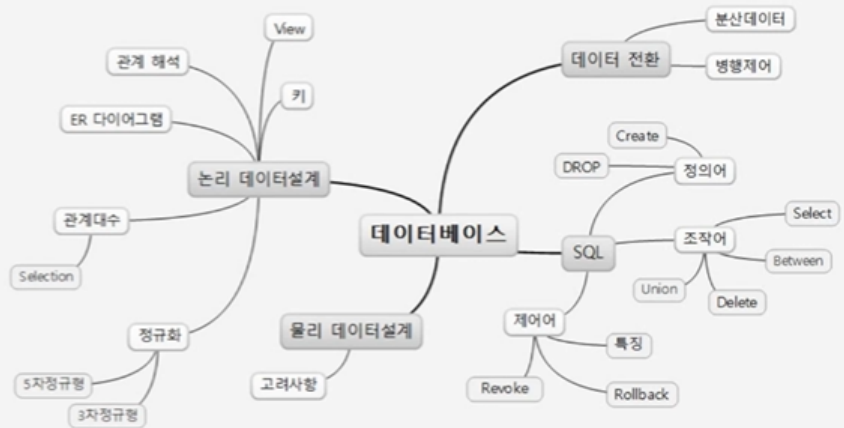


이거전

# 3과목

## 데이터베이스 개발


 절대꼭보 | 3과목



이거!

# 4과목

## 프로그래밍 언어 활용

 절대꼭보 | 4과목

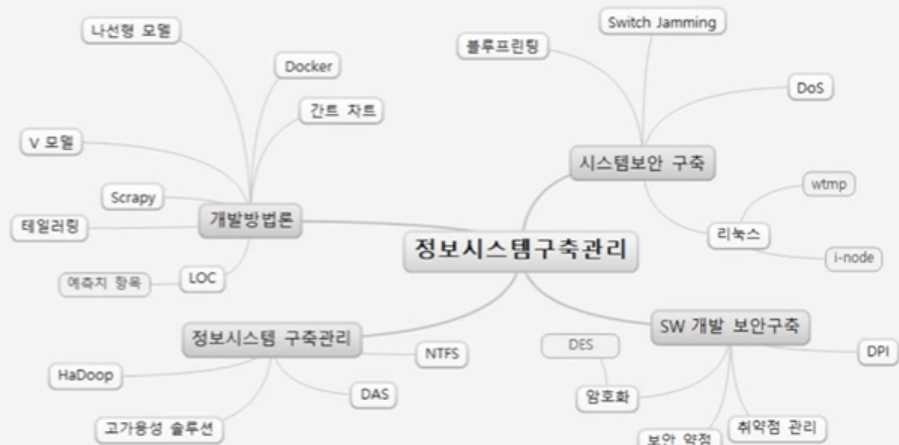


이거!

# 5과목

## 정보시스템 구축관리

절대족보 | 5과목



이거점

1,2,3 과목은 기출에서 많이 나옴

4, 5과목은 아는 것만 하고 안되면 버리기

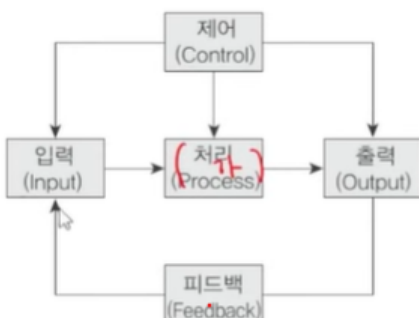
## 소프트웨어 설계 - 1

### 1. 소프트웨어 공학의 개념

#### 01 소프트웨어

소프트웨어 개념

- 소프트웨어 특징
  - 상품성 : 돈 벌어야지
  - 복잡성 : 공수가 많이들고 다양한 단계가 있다
  - 변경 가능성 : 업데이트 가능
  - 복제성 : 배포, 유통
- 시스템의 개요와 기본요소
- **기본요소**
  - 입력
  - 출력
  - 제어
  - 피드백



- 소프트웨어 위기
  - 개발비용 증가 (많은 인력이 들기 때문에)

- 개발기간 지연
- 개발인력 부족 및 인건비 상승
- 성능 및 신뢰성 부족
- 유지보수 <- 애가 제일 돈이 많이 듦 (윈도우 업데이트도 돈 다 들어감)

## 02 소프트웨어 공학

소프트웨어 공학 : 적은 돈으로 최고의 효율 뽑아내기

- 소프트웨어 공학의 기본원칙
  - 신뢰성(2 + 2 = 4 가 반드시 되어야함)
  - 현대적인 프로그래밍 기술
  - 편리성 유지보수성
  - 지속적인 검증(유지보수 느낌인듯)

## 2. 재공학

옛날에 만들었던걸로 다시 쓰기

- 장점
  - 개발시간 비용감소,
  - 품질향상
  - 생산성향상
  - 신뢰성 향상
  - 등등
- 목표
  - 유지보수성 향상이 최우선 목표
  - 골수까지 빨아먹자 같은 느낌
- 과정
  - 분석 -> 구성 -> 역공학(애가 어떻게 작동하는지 거꾸로 들어가는거) -> 이식

## 02 역공학

역공학의 핵심은 **재문서화**

## 03 CASE(Computer Aided Software Eng)

- CASE(소프트웨어 개발에 도움을 주는 도구) <- 소프트웨어 분야의 CAD같은거
- CASE가 제공하는 기능
  - 개발을 신속하게 하고 오류 수정 쉽게
  - ....
- CASE 사용의 장점
  - 개발 기간 단축, 비용절약, 생산성 향상
- CASE의 분류
  - 상위(Upper CASE) : 요구분석, 설계 지원
  - 하위(Lower CASE) : 실제 구현에 도움을 줌
  - 통합(Integrate CASE) : 개발 주기 전체과정을 지원함
- CASE 도구 예시
  - SADT(Struc) : SoftTech사, 블록다이어그램 키워드면 애임
  - 애는 나오면 강 틀리자

## 3. 소프트웨어 개발 방법론

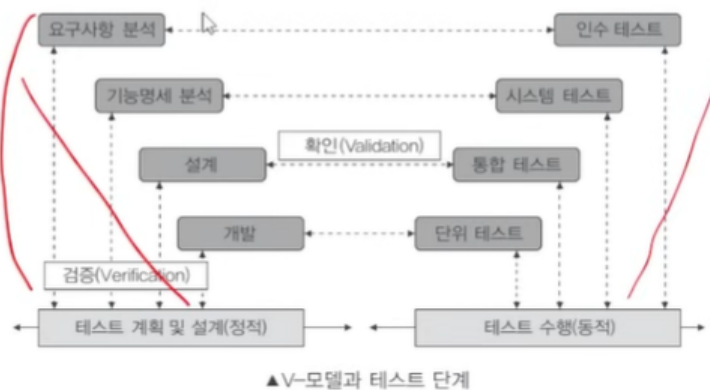
### 01 소프트웨어 설계 방법론

- 소프트웨어 생명주기
- 타당성 검토 -> 개발 계획 -> 요구 사항 분석 -> 설계 -> 구현 -> 테스트 -> 운용 -> 유지보수

- 폭포수 모형(Waterfall Model)
  - 순차적으로 꼭 하는거, 이전 단계로 돌아가지 않음
  - 중간에 바뀌어도 돌아갈 수 없는 모델이라는 단점이 있음
- 나선형 모델(Spiral Model)
  - 반복적으로 작업
  - 위험 분석이라는 단계가 있음(이벤트, 변동사항을 위한 완충지역)



- 위 과정 4개 기억하기
- 하향식 설계(Top down) : 폭포수랑은 다르게 애는 기능적으로 위아래임, 네트워크 layer 생각하기
- 상향식 설계(Bottom up)
- 프로토타입 모형 : 결과물의 예상 견본
- HIPO(Hierarch Input Process Output)
  - 계층적으로 되어있는 문서 도구
  - 가시적 도표, 총체적 다이어그램, 세부적 다이어그램
  - 보기쉽고 이해하기 쉽다(가시적 도표니까)
  - 하향식 소프트웨어 개발을 위한 문서화 도구
  - 폭포수형이랑 비슷함
- V - 모델
  - HIPO + 테스트



- 정적 테스트 : 코드분석
- 동적 테스트 : 실제로 실행해봄
- 뒷 단원에 또 나옴, 그때 자세히

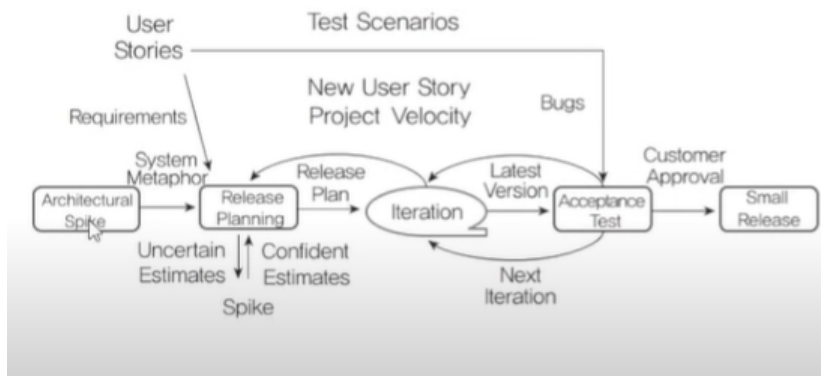
## 02 애자일(Agile) 개발 방법론

- 빨리, 고객이 필요한것만
- 절차, 도구 다 상관없음
- 과녁에 날아가는 화살같은 이미지
- 제대로 된걸 만들어야함(빠르게 하더라도 대충한다는 의미는 아님)
- 종류
  - 익스트림프로그래밍(XP)

- 스크럼(SCRUM)
- 린(Lean)
- DSDM(Dynamic System Development Method)
- FDD(Feature Driven Development)
- 요정도 알아두기

### 03 XP(eXtremProgramming)

- XP 핵심 가치
  - 소통(communication)
  - 단순성(Simplicity) : 부가적인건 다 빼자
  - Feedback
  - 용기(Courage) : 고객의 요구사항에 능동적으로 대처
  - 존중(Respect) : 싸우지말자



- Spike : 요구사항을 확인하기 위한 간단한 프로그램

### XP의 12가지 실정사항(Practice) : 큰 그림으로 이해하기

- Fine Scale Feedback
  - Pair Programming(짝 프로그래밍) : 한명은 개발, 한명은 테스트
  - Planning Game : 게임처럼 규칙, 목표를 두고 하기
  - Test Driven Development : 실제 코드 작성 전에 단위 테스트를 하기
  - Whole team : 사용자가 팀에 속해야함(모두 한팀이야)
- Continous Process
  - Continous Integration : 하나를 만들면 항상 배포할 수 있는 상태로 유지
  - Design Improvement : 잘 하자
  - Small Release : 짧은 주기로 배포를해서 고객에게 피드백받기
- Shared Understanding
  - Coding Standards : 표준에 맞게 코딩하자
  - Collective Code Ownership : 모든코드의 주인은 팀원 전체다
  - Simple Design : simple is best
  - System Metaphor : 최종적으로 개발될 시스템의 구조를 기술한다
  - Sustainable : 근로기준법을 지키자
- 효과적인 프로젝트 관리를 위한 3대요소(3P라고도 물어볼 수도)
  - People : 사람
  - Problem : 문제인식
  - Process : 작업계획

### 4. SCRUM

- 한번 보고 넘어가자, 쓰기 너무 길드

### 5. 현행 시스템 분석(회사가 어떻게 굴러가나, 뭘 쓰나 한번 보자)

- 현행 시스템 파악 절차
  - 1단계 (시스템 구성 파악(어떤 부서가 있나) - 시스템 기능 파악 - 시스템 인터페이스 현황 파악)
  - 2단계(아키텍처 파악 - 소프트웨어 구성 파악)
  - 3단계(시스템 하드웨어 현황 파악 - 네트워크 구성 파악)
  - 
  - 시스템이 어케 되었나 -> 내부 아키텍처, 소프트웨어를 보고 -> 하드웨어 보는 순서
- 시스템 아키텍처 (컴퓨터 시스템이 아니라 조직이라는 의미의 시스템)
  - 말 그대로 설계도
  - 복잡한 시스템의 경우에는 핵심적인걸 기준으로
  - 시스템 전체의 **구조**, **행위**, 작동원리 설명 같은 거
- 플랫폼 성능 특성 분석
  - 특성 분석 항목 : 응답시간(Response Time), 가용성(Availability), 사용률(Utilization)
  - 성능 분석 방법
    - 기능 테스트
    - 사용자 인터뷰
    - 문서 점검

TCO(Total Cost of Ownership) : 총 비용이랑 비슷한 개념

Apach 2.0 : ex) HADOOP (싼 컴퓨터를 하나처럼 묶어서 빅데이터를 처리하는 기술 )

## 6. 요구사항 개발

서로 의사소통을 원활히 하기위해 소단위 명세서 등을 사용해서 잘 하자

요구사항 도출 기법 :

고객의 발표

문서조사

설문

업무 절차 및 양식 조사

브레인 스토밍

워크숍

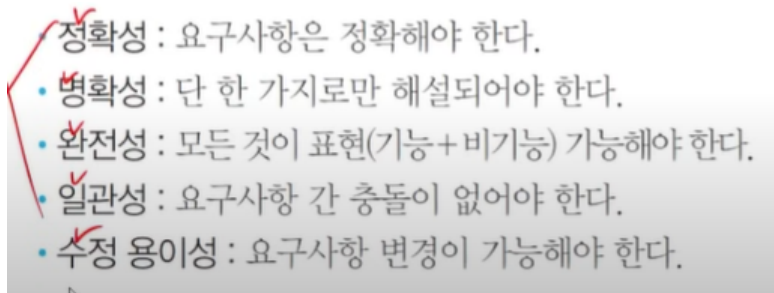
인터뷰

Use Case

BPR(업무 재설계)

등등

- 요구사항 명세 속성

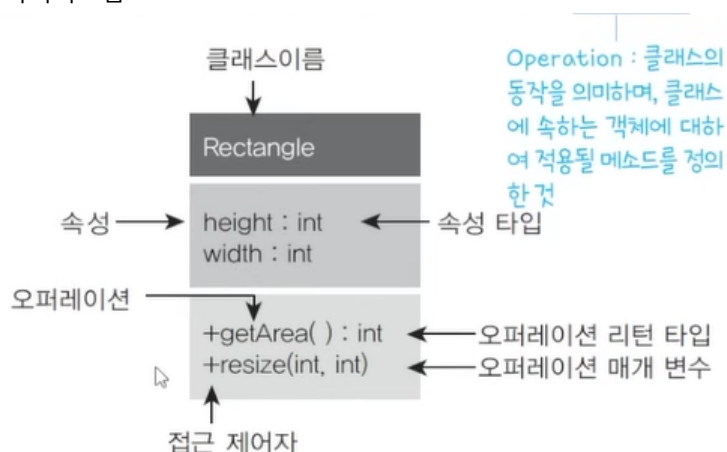


- 추적성 : RFP, 제안서를 통해 추적이 가능해야한다.
- 요구사항 관리 도구의 필요성 : 요구사항 변경으로 인한 비용 편익 분석, 변경 추적, 영향 평가

## 7. UML

- 종류 : Use case diagram, data flow model, state model, Goal-based model, User interactions, object model, data model
- UML(Unified Modeling Language)
- 럼바우(Rumbaugh) 객체 지향 분석 기법
  - 객체 모델링 : **객체 다이어그램**으로 표시
  - 동적 모델링 : 객체의 상호작용 등을 모델링을 **상태 다이어그램**으로 도출됨

- 기능 모델링 : 프로세스 간의 **자료 흐름** 이 도출됨
- UML 소프트웨어에 대한 관점
  - 기능적 관점 : 사용자 측면, 사례 모델링 - Use case diagram
  - 정적 관점 : 구조적 관계, 연관관계 - Class diagram
  - 동적 관점 : 내부의 실제 동작 - Sequence Diagram, State Diagram, Activity Diagram
- UML 기본 구성
  - 사물
  - 관계
  - 다이어그램
- 스테레오 타입 - 확장모델에서 스테레오 타입 객체를 표현할 때 **<< >>** (**길러멧**) 기호를 사용함 여기 안에 확장요소를 넣음
- UML 접근 제어자
  - Public(+)
  - Private(-)
  - Protected(#)
  - Package(~)
- 나중에 실기에서 다시 봐야함
- 구조적 다이어그램(정적인 애들)
  - 클래스 : 정적 구조 표현하는 다이어그램
  - 객체 : 객체 정보
  - 복합체 구조
  - 배치 : Deployment임 Batch (x)
  - 컴포넌트
- 행위 다이어그램
  - Use case : 사용자관점에서 시스템의 행위
  - Activity : 수행과정
  - State machine
  - Collaboration
  - 상호작용 다이어그램
    - 순차
    - 상호작용
    - 통신
    - 타이밍
  - **외우지 말고 동작인 느낌이면 행위 다이어그램. 정적인 느낌이면 구조적**
- 클래스 다이어그램



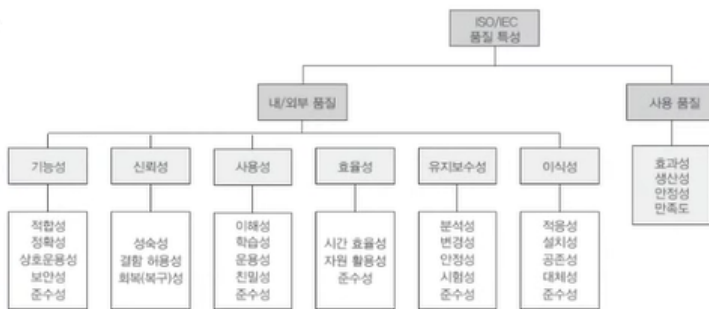
- UML 관계 표현
  - 검색해서 뭐있는지 보기
- UML 연관 관계





여러 관계를 다이어그램으로 어떻게 표현하는지 보고 가기

## 8. 소프트웨어 아키텍처



- 내/외부 품질, 사용품질 구분해서 파악하기

## UI 표준 및 지침

- UI 설계 원칙
  - 직관성
  - 유효성 : 너무 작게 만들지 말기
  - 학습성
  - 유연성
- UI 설계 지침
  - 사용자 중심
  - 일관성
  - 단순성
  - 가시성
  - 표준화
  - 접근성
  - 결과 예측 가능
  - 명확성
  - 오류 발생 해결

## 10. UI 설계

- UI 상세 설계 단계
  1. UI 메뉴 구조 설계
  2. 내/외부 화면과 폼 설계
  3. 검토