

## 1 장. 요구사항 확인

### 1. 소프트웨어 생명주기 : 소프트웨어를 개발하기 위한 과정을 단계별로 나눈 것

#### ① 폭포수모형

- 고전적 생명주기, 각 단계가 끝난 후 다음 단계 수행, 단계별 결과물 명확

#### ② 프로토타입모형

- 견본품을 만들어 최종 결과물 확인

#### ③ 나선형모형

- 여러 번의 개발과정을 거쳐 점진적으로 개발하는 모형
- 보험이 제안, 유지보수 과정 불필요
- 진행 방법 : 계획수립-위험분석-개발및검증-고객평가 (계위개고)

#### ④ 애자일모형

- 요구사항에 유연하게 대응 가능
- 일정한 주기를반복하며 개발
- 폭포수 모형과 대조
- 핵심가치  
: 상호작용 중요, SW 중요, 변화에 반응 중요
- 대표적 개발모형

##### ■ 스크럼

- 팀이 중심이 되어 개발의 효율을 높이는 기법
- 스크럼 팀 구성  
: PO(제품책임자, 백로그 작성 주체), SM(스크럼마스터, 가이드역할), DT(개발팀)
- 스크럼 과정 순서 : 백로깅 - 스프린트 계획회의- 스프린트(개발)- 검토 - 회고

##### ■ XP(eXtreme Programming)

- 고객의 참여와 개발 과정의 반복을 극대화 하여 생산성을 확대
- 핵심가치 : 의사소통, 단순성, 용기, 존중, 피드백 (피존의 단순한 용기)
- 개발 프로세스 : 릴리즈 계획수립-이터레이션(개발)-승인검사-릴리즈
- 주요 실천 방법
  - 짝 프로그래밍 (Pair) : 함께 개발->책임 공동
  - 공통코드소유(Collective Ownership) : 코드에 대한 권한과 책임을 공동 소유
  - 테스트 주도 개발 : 개발전 테스트 케이스 먼저 작성
  - 전체 팀 : 개발 참여 모든 구성원들에게 책임
  - 지속적인 통합 : 개발된 코드들은 작업이 마무리 될 때마다 통합됨
  - 리팩토링 : 프로그램 기능 변경없이 시스템을 재구성-> 쉽게 이해하고, 수정 용이
  - 소규모 릴리즈 : 잦은 릴리즈로 고객 요구 변화에 신속 대응

##### ■ 칸반

##### ■ Lean

##### ■ 기능중심개발 (FDD)

#### ⑤ 소프트웨어 공학 : 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문

### 2. 운영체제

- 컴퓨터 시스템의 자원을 효율적으로 관리
- 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어
- 운영체제 고려사항(5) : 가용성, 성능, 기술지원, 주변기기, 구축비용

### 3. DBMS (데이터베이스 관리 시스템)

- 사용자와 데이터베이스 사이에서 정보 생성 및 데이터 베이스 관리 소프트웨어

- DBMS 고려사항(5): 가용성, 성능, 기술지원, 상호호환성, 구축비용

#### 4. WAS (웹어플리케이션 서버)

- 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어
- 서버 사이에서 인터페이스 역할을 수행
- 데이터접근, 세션관리, 트랜잭션 관리 등을 위한 라이브러리 제공
- 데이터베이스 서버와 연동하여 사용
- WAS 고려사항(4) : 가용성, 성능, 기술지원, 구축비용

#### 5. 오픈소스

- 공개된 소프트웨어
- 오픈소스 고려사항 (3): 라이선스의 종류, 사용자수, 기술의 지속가능성

#### 6. 요구사항

: 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 운영되는데 필요한 제약조건

##### ① 기능 요구사항

- 기능, 수행에 관련된 요구사항
- I/O에 대한 사항, 데이터 저장 연산 수행사항, 기능수행사항, 사용자가 제공받기를 원하는 기능

##### ② 비기능 요구사항

- 품질과 제약사항에 관련된 요구사항
- 장비 구성요구사항, 성능 요구사항, 인터페이스 요구사항, 구축에 필요한 요구사항
- 테스트 요구사항, 보안 요구사항
- 품질 요구사항 : 가용성, 정합성, 상호호환성, 대응성, 이식성, 확장성, 보안성
- 프로젝트 관리, 자원 요구사항

##### ③ 사용자 요구사항

- 사용자 관점에서 본 시스템이 제공해야할 요구사항

##### ④ 시스템요구사항 (소프트웨어 요구사항)



- 개발자 관점에서 본 시스템 전체가 제공해야 할 요구사항

#### 7. 요구사항 개발 프로세스 : 도출- 분석- 명세- 확인 (도요명확)

##### ① 요구사항 도출

- 요구사항을 식별하고 이해하는 과정
- 소프트웨어 개발 생명 주기 동안 지속적으로 반복
- 요구사항 도출 기법
  - 청취, 인터뷰, 설문, 워크샵, 프로토타이핑, 유스케이스(요구사항을 기능 단위로 표현)

##### ② 요구사항 분석

- 요구사항의 타당성을 조사하고 비용과 일정에 대한 제약을 설정
- 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화 하는 활동
- 구조적 분석 기법
  - 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법
  - 하향식 방법 사용, 중복을 배제
  - 구조적 분석 기법 도구
    - I. 자료 흐름도 (DFD) = 버블 차트 , 자료 흐름 그래프
      - 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술 하는 방법
      - PDDT
        - Process, Data Flow, Data Store, Terminator
        - 프로세스, 자료흐름, 자료 저장소, 단말
        - O, ->,  , 

II. 자료사전 (DD) = 메타 데이터

- 자료 흐름도에 있는 자료를 정의하고 기록한 것
- = (정의), +(자료연결), ()(자료생략), [](자료 선택, OR), {} (자료 반복), \*(주석)

III. 소단위 명세 (Mini-Spec)

IV. 개체관계도 (ERD)

V. 상태전이도 (STD)

VI. 제어 명세서

③ 요구사항 명세

- 요구사항을 바탕으로 모델을 작성하고 문서화 하는 것
- 기능 요구사항 필수, 비기능 요구사항 선택
- Mini-Spec 사용
- 요구사항 명세 기법

	정형 명세 기법	비정형 명세 기법
기법	수학적 원리기반, 모델기반	상태/기능/객체중심
특징	요구사항 간결 표현 일관성이 있어 검증 용이 표현법 어려움	자연어로 표현 일관성이 떨어지고 해석이 달라짐 의사소통 용이
종류	VDM, Z, Petri-net, CSP	ER 모델링, Decision Table, State Chart, FSM

④ 요구사항 확인

- 요구사항 명세서가 정확하고 완전하게 작성되었는지 검토
- 문서들에 대한 형상 관리 수행

8. 요구공학

- 요구사항을 정의하고 분석 및 관리하는 프로세스를 연구하는 학문

9. 요구사항 CASE(자동화 도구)

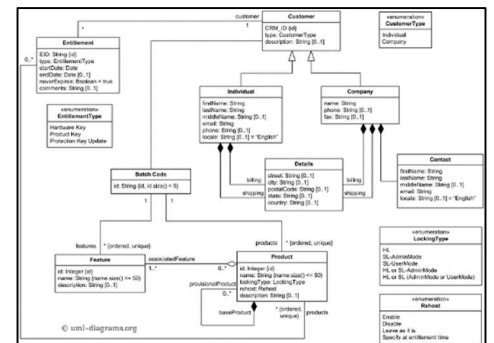
- 요구사항을 자동으로 분석, 요구사항 분석 명세서를 기술하도록 개발된 도구
- 분석용 CASE
  - ① SADT : 구조적 요구 분석을 하기 위해 블록 다이어그램 채택, SoftTech사에서 개발
  - ② SREM = RSL/REVS : 실시간 처리 소프트웨어 시스템에서 요구사항을 명확히 기술하도록 할 목적으로 개발한 도구
  - ③ PSL/PSA : 미시간 대학에서 개발
  - ④ TAGS : 시스템 공학 방법 응용에 대한 자동 접근 방법, 개발 주기의 전 과정에서 이용 가능

10. HIPO (Hierarchy Input Process Output)

- 시스템 실행 과정인 입력 처리 출력의 기능을 표현
- 하향식 소프트웨어 개발을 위한 문서화 도구
- HIPO CHART : 인터페이스를 계층 구조로 표현한 것
  - ① 가시적 도포
  - ② 총체적 도포
  - ③ 세부적 도포

11. UML (Unified Modeling Language)

- 시스템 개발 과정에서 의사소통이 원활하게 이뤄지도록 표준화한 대표적 객체 지향 모델링 언어
- UML 구성요소



< UML 예시 모형 >

### ① 사물(Things)

- 다이어그램 안에서 관계가 형성될 수 있는 대상
- 사물의 종류 (구그주행)
  - 구조사물 : 시스템의 개념적 물리적 요소 표현  
ex) 클래스, 유스케이스, 컴포넌트, 노드
  - 행동사물 : 시간과 공간에 따른 요소들의 행위를 표현  
ex) 상호작용, 상태머신
  - 그룹사물 : 요소들을 그룹으로 묶어서 표현  
ex) 패키지
  - 주해사물 (Annotation Things) : 부가적인 설명이나 제약조건 등을 표현  
ex) 노트

### ② 관계(Relationships)

- 사물과 사물 사이의 연관성을 표현
- 관계의 종류 (일집포의 실연 -> 일취포의 실연)
  - 연관관계 : 2 개 이상의 사물이 서로 연관  
방향성은 화살표, 다중도를 선위에 표기
  - 집합관계 : 하나의 사물이 다른 사물에 포함되어있는 관계, 빈마름모
  - 포함관계 : 포함하는 사물의 변화가 포함되는 사물에게 영향을 주는 관계, 짝찬마름모
  - 일반화관계 : 상위 class 개념, 부모다 구체적인 개념을 자식이라 부름, 빈삼각형
  - 의존관계 : 서로에게 영향을 주는 짧은 시간동안만 연과을 유지 , 점선화살표
  - 실체화 관계 : 사물이 해야하는 기능으로 서로를 그룹화 할 수 있는 관계, 점선 삼각형

### ③ 다이어그램(Diagram)

- 사물과 관계를 도형으로 표현한 것 (정객동상)
- 정적 모델링 -> 구조적 다이어그램 사용
  - 기능을 구현하는데 필요한 자료들의 논리적인 구조로 표현
  - 정적 모델링은 객체들을 클래스로 추상화하여 표현한다.
- 구조적 다이어그램 종류
  - 클래스 다이어그램
    - 클래스 사이의 관계 표현
    - 구성요소
      - ① 클래스 : 클래스명, 속성, 오퍼레이션으로 구성  
각 객체들이 갖는 속성과 오퍼레이션을 표현한 것
      - ② 제약조건 : 클래스 안에서 제약조건을 기술할 때는 {} 사용
      - ③ 관계 : 연관관계, 집합관계, 포함관계, 일반화관계, 의존관계, (일취포의 실연)
    - 연관클래스 : 연관관계에 있는 두 클래스에 추가적으로 표현해야하는 오퍼레이션이 있는 경우 생성되는 클래스
  - 객체 다이어그램 : **럼바우 객체지향** 분석 기법에서 객체 모델링에 사용
  - 컴포넌트 다이어그램 : 컴포넌트 간의 인터페이스 표현, 구현 단계에서 사용
  - 배치 다이어그램 : 물리적 요소들의 위치를 표현함, 구현단계에서 사용
  - 복합체 구조 다이어그램 : 내부 구조를 표현
  - 패키지 다이어그램
    - 유스케이스나 클래스 등을 그룹화한 패키지 간의 의존 관계를 표현한 것
    - 대규모 시스템에서 주요 요소 간의 종속성을 파악하는데 사용
    - 구성요소

- ① 패키지 : 객체들을 그룹화한 것,
  - 단순표기법 : 패키지안에 이름만 표현
  - 확장표기법 : 패키지안에 요소까지 표현
- ② 객체 : 패키지에 포함될 수 있는 다양한 요소
- ③ 의존관계 : 점선 화살표로 연결하여 표현
  - <<import>> : 패키지에 포함된 객체를 직접 가져와 이용하는 관계
  - <<access>> : 인터페이스를 통해 패키지 내의 객체에 접근하여 이용하는 관계

■ 동적모델링 -> 행위다이어그램

- 시스템의 내부 구성요소들의 상태변화 과정과 과정에서 발생하는 상호작용을 표현한것

■ 행위 다이어그램의 종류

- 유스케이스 다이어그램

- 기능 모델링에서 사용

- 사용자의 요구를 분석, 사용자와 사용사례로 구성
- 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점에서 표현
- 구성요소
  - ① 시스템/ 시스템 범위 (□): 유스케이스들을 사각형으로 무어 시스템 범위 표현
  - ② 액터 : 주액터(시스템 사용자), 부액터(서비스 제공하는 외부시스템, 조직 기관)
  - ③ 유스케이스 (o) : 액터에게 제공하는 서비스나 기능
  - ④ 관계 <<include>>, <<extends>>: 포함관계, 확장관계 일반화 관계가 존재

- 시퀀스 다이어그램

- 객체들이 메시지를 주고받으며 상호작용하는 과정을 표현한 것
- 구성요소
  - ① 액터 : 서비스를 요청하는 외부 요소 (사람, 시스템)
  - ② 객체 : 메시지를 주고받는 주체
  - ③ 생명선 : 객체가 메모리에 존재하는 기간
  - ④ 실행상자 : 객체가 메시지를 주고받으며 구동되고 있음을 표현
  - ⑤ 메시지 :
  - ⑥ 객체소멸
  - ⑦ 프레임

- 커뮤니케이션 : 객체들이 주고받는 메시지와 객체들 간의 연관관계 표현
- 상태 다이어그램 : 클래스의 상태 변화, 다른 객체의 상호 작용에 따라 상태 변화 표현

림바우 객체지향 분석기법에서 동적모델링에 활용됨

- 활동다이어그램 (계약 프로세스 생각)

- 기능모델링에서 사용

- 자료흐름도와 유사
- 객체 처리 로직 혹은 처리 흐름을 순서에 따라 표현
- 구성요소
  - ① 액션/액티비티 (□): 액션 : 더 이상 분해할 수 없는 단일 작업  
액티비티 : 몇 개의 액션으로 분리 가능
  - ② 시작 노드 (●)
  - ③ 종료노드 (●)
  - ④ 조건노드 (마름모): 들어오는 흐름은 하나 나가는 흐름은 여러 개
  - ⑤ 병합노드 (마름모): 들어오는 흐름 여러 개 나가는 흐름 하나

- ⑥ 포크노드 (선): 흐름이 분리되어 동시에 수행됨을 표현
- ⑦ 조인노드 (선): 흐름이 다시 합쳐짐을 표현
- ⑧ 스웜레인 (|): 액티비티 수행을 담당하는 주체를 구분하는 선

- 상호작용 개요 다이어그램
- 타이밍 다이어그램

■ 스테레오 타입 :

- UML 에서 표현하는 기본 기능 외에 추가적인 기능을 표현
- << include >> : 포함관계
- << extend >> : 확장관계
- << interface >>, << exception >>, << constructor >>

## 12. 소프트웨어 개발 방법론

- 소프트웨어 개발, 유지보수에 필요한 수행방법과 각종기법 및 도구를 정리하여 표준화 한 것
- 주요 소프트웨어 개발 방법론
  - ① 구조적 방법론
    - 사용자 요구사항을 파악하여 문서화하는 처리 중심의 방법론
    - 타당성검토- 계획- 요구사항- 설계- 구현- 시험- 운영 (타계요설구시운)
  - ② 정보공학 방법론 (data)
    - 계획, 분석, 설계, 구축에 정형화된 기법들을 통합 및 적용하는 자료(data) 중심의 방법론
    - 계획수립 - 분석 - 설계- 구축 (계분설구)
  - ③ 객체지향 방법론
    - 객체를 조립해서 소프트웨어를 구현하는 방법론
    - 구성요소 : 객체, 클래스, 메시지
    - 기본원칙 : 캡슐화, 정보은닉, 추상화, 상속성, 다형성
    - 분석- 설계- 구현-테스트-인도
  - ④ 컴포넌트 기반 방법론
    - 컴포넌트를 조합하여 새로운 어플리케이션을 만드는 방법
    - 컴포넌트 재사용, 확장성 보장, 유지보수 비용 최소화, 생산성 및 품질 향상
    - 준비-분석-설계-구현-테스트-전개-인도
  - ⑤ 제품 계열 방법론
    - 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론
    - 임베디드 소프트웨어를 만드는데 적합
  - ⑥ 애자일 방법론

## 13. 소프트웨어 재사용

- 이미 개발되어 인정받은 소프트웨어를 다른 소프트웨어 개발이나 유지에 사용하는 것
- 합성 중심 : 소프트웨어 부품, 즉 블록을 만들어서 끼워 맞춰 소프트웨어를 완성시키는 방법
- 생성 중심 : 추상화 형태로 써진 명세를 구체화하여 프로그램을 만드는 방법

## 14. 소프트웨어 재공학

- 기존 시스템을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상시킨다.
- 소프트웨어 재공학의 이점
  - ① 품질향상
  - ② 생산성증가
  - ③ 수명연장

④ 오류감소

15. CASE

- 소프트웨어 개발 과정에서 사용되는 과정을 소프트웨어 도구를 사용하여 자동화하는 것
- 소프트웨어 생명주기 전 단계의 연결 지원
- 다양한 소프트웨어 개발 모형 지원
- 그래픽 지원

16. 하향식 비용 산정 기법

- 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 방법
- 하향식 비용 산정 기법
  - ① 전문가 감정 기법 : 두명 이상의 전문가에게 비용 산정 의뢰
  - ② 델파이 기법 : 많은 전문가의 의견을 종합하여 산정하는 기법

17. 상향식 비용산정 기법

- 세부적인 작업 단위별로 비용을 산정한 후 집계하여 비용 산정
- 상향식 비용 산정 기법

① LOC (원시코드 라인수) 기법

- 비관치, 낙관치, 기대치를 측정하여 예측치를 구하여 비용 산정한다

비관치 (가장 많이 측정된 코드 라인수)

낙관치 (가장 적게 측정된 코드 라인수)

기대치 (측정된 모든 코드 라인 수의 평균)

예측치 =  $\frac{a+4m+b}{6}$  (a: 낙관치, m: 기대치, b:비관치)

노력(인월) = 개발기간 × 투입인원 =  $\frac{LOC}{1 \text{ 인당 월평균 생산 코드 라인수}}$

생산성 =  $\frac{LOC}{\text{노력(인월)}}$

② 개발 단계별 인월수 기법

- LOC 기법을 보완하기 위한 기법
- 기능을 구현 시키는데 필요한 노력을 생명 주기의 각 단계별로 산정

③ 수확산정법 기법

- 주요 수학적 산정 기법

I. COCOMO 모형

- LOC 에 의한 비용산정 기법
- 보헴이 제안
- COCOMO 소프트웨어 개발 유형
  - ◆ 조직형(Organic Mode) : 5 만 라인 이하의 소프트웨어 개발 유형,
  - ◆ 반분리형(semi – detached Mode) : 30 만 라인 이하의 소프트웨어 개발 유형  
컴파일러, 인터프리터와 같은 유틸리티 개발에 적합함
  - ◆ 내장형(Embedded Mode) : 30 만 라인 이상의 소프트웨어 개발 유형  
신호기 제어 시스템, 미사일 유도 시스템,실시간 처리시스템 개발에 적합함
- COCOMO 모형의 종류
  - ◆ 기본형 COCOMO : 소프트웨어의 크기와 개발 유형만을 이용하여 비용 산정
  - ◆ 중간형 COCOMO : 기본형 공식을 토대로 사용, 4 가지 특성에 의한 비용 산정  
특성 : 제품의 특성, 컴퓨터 특성, 개발 요원의 특성, 프로젝트 특성
  - ◆ 발전형 COCOMO : 개발 공정 별로 노력을 산출하여 비용 산정  
소프트웨어 환경과 구성요소가 사전에 정의되어 있어야함  
개발 후반부에 주로 적용함

## II. Putnam 모형

- 소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 예상하는 모형
- 개발 기간이 늘어날수록 프로젝트 적용 인원의 노력이 감소한다

## III. 기능점수 모형

- 소프트웨어 기능을 증대시키는 요인별로 기능점수를 구한 후 비용을 산정하는 기법
- 알브레히트가 제안
- 소프트웨어 증대 요인

◆ 자료입력, 정보출력, 명령어, 데이터파일, 외부 루틴과의 인터페이스

## IV. 비용 산정 자동화 추정 도구

- SLIM : Putnam 예측 모델을 기초로 개발된 자동화 추정도구
- ESTIMACS : 기능점수 모형을 기초로하여 개발된 자동화 추정 도구

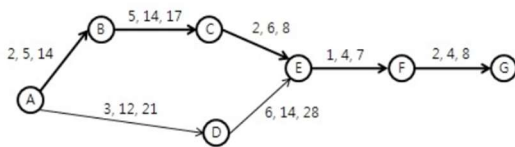
## 18. 프로젝트 일정계획

- 프로세스를 이루는 소작업을 파악하고 예측된 노력을 각 소작업에 분배하여 일정을 정하는 것
- 프로젝트 일정계획에 사용되는 기능

### ① WBS

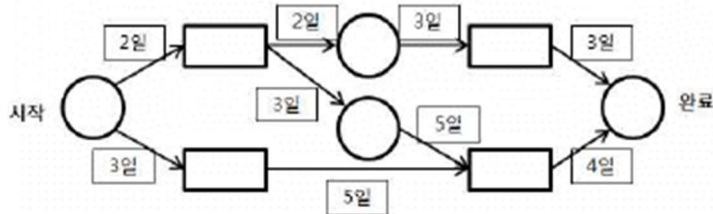
### ② PERT

- 전체 작업의 상호 관계를 표시하는 네트워크
- 노드와 간선으로 구성되며 간선에는 낙관치, 기대치, 비관치 표시
- 작업예측치 =  $\frac{\text{비관치} + 4 \times \text{기대치} + \text{낙관치}}{6}$
- 평방편차 =  $\left( \frac{\text{비관치} - \text{낙관치}}{6} \right)^2$



### ③ CPM (임계경로기법)

- 작업을 나열하고 작업에 필요한 소요기간을 예측하는데 사용하는 기법



### ④ 간트차트

- 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표

## 19. 소프트웨어 개발 표준

- 소프트웨어 개발 단계에서 수행하는 품질 관리에 사용되는 국제 표준

### ① ISO/IEC 12207

- ISO에서 만든 표준 소프트웨어 생명 주기 프로세서
- 기본 생명 주기 프로세스 : 획득, 공급, 개발, 운영, 유지보수
- 지원 생명 주기 프로세스 : 품질보증, 확인, 활동검토, 감사, 문서화, 형상관리
- 조직 생명 주기 프로세스 : 관리, 기반 구조, 훈련, 개선 프로세스

### ② CMMI

- 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델
- 프로세스 성숙도 : 초기-관리-정의-정략적 관리-최적화

### ③ SPICE



- 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준
  - 수행능력 단계 (불수관 확예최)
- : 불완전-수행-관리-확립-예측-최적화

## 20. 프레임워크

- 프레임워크 특성
  - ① 모듈화
  - ② 재사용성
  - ③ 확정성
  - ④ 제어의 역흐름
- 주요기능
  - ① 예외처리
  - ② 트랜잭션처리
  - ③ 메모리공유
  - ④ 데이터소스관리
  - ⑤ 서비스 관리
  - ⑥ 쿼리 서비스
  - ⑦ 로깅서비스
  - ⑧ 사용자 인증 서비스

21. 스프링 프레임워크 : 자바 플랫폼을 위한 오픈소스 경량형 애플리케이션 프레임워크

22. 전자정부 프레임워크 : 대한민국의 공공부문 정보화 사업시 시시스템 구축을 지원하기 위한 기능 및 아키텍처를 제공하는 프레임워크

23. 닷넷 프레임워크 : 윈도우 프로그램의 개발 및 실행환경을 제공하는 프레임워크

24. 이중화 : 예시 서버에 동일하게 복제되어 관리하는 것.

## 2 장. 데이터 입출력 구현

### 1. 데이터저장소

- 데이터를 논리적인 구조로 조직화하거나, 물리적인 공간에 구축한 것을 의미

### 2. 데이터베이스

- 공동으로 사용될 데이터를 중복을 배제하여 통합
- 저장장치에 저장하여 항상 사용할 수 있도록 운영하는 데이터

### 3. DBMS

- 데이터베이스를 관리해주는 소프트웨어
- 필수 기능
  - ① 정의 기능 : 데이터 형과 구조에 대한 정의, 이용방식, 제약 조건 등을 명시하는 기능
  - ② 조작 기능 : 데이터 검색, 삽입, 삭제 등을 위해 제공하는 기능
  - ③ 제어 기능 : 데이터의 무결성, 보안, 권한 검사, 병행 제어를 제공하는 기능

### 4. 데이터의 독립성

- 물리적 독립성 : 물리적 장치를 독립, 디스크를 추가/변경하더라도 응용프로그램은 영향 없음
- 논리적 독립성 : 응용프로그램과 데이터베이스를 독립, 데이터의 논리적 구조를 변경하더라도 응용프로그램에 영향 없음

### 5. 스키마

- 데이터베이스의 구조와 제약조건에 관한 전반적인 명세를 기술한 것
- 스키마 종류
  - ① 외부 스키마 : 사용자나 프로그래머의 입장에서 데이터 베이스의 논리적 구조를 정의한 것
  - ② 개념 스키마 : 데이터 베이스의 전체적인 논리적 구조
  - ③ 내부 스키마 : 물리적 저장장치 입장에서 본 데이터베이스 구조

### 6. 데이터베이스 설계 고려사항 (무일 회보 효능)

- ① 무결성 : 저장된 데이터가 정해진 제약조건을 항상 만족해야 함
- ② 일관성 : 질의에 대한 응답이 변함없이 일정해야 함
- ③ 회복 : 장애가 발생 했을 때 그 직전의 상태로 복구할 수 있어야 함
- ④ 보안 : 외부로부터 데이터를 보호할 수 있어야 함
- ⑤ 효율성 : 응답시간의 단축, 시스템의 생산성, 저장 공간의 최적화 등이 가능해야함
- ⑥ 데이터베이스 확장 : 지속적으로 데이터를 추가할 수 있어야 함

### 7. 데이터베이스 설계 순서 (개논물)

- 요구조건분석- 개념적설계 - 논리적설계 - 물리적설계 - 구현
  - ① 요구조건분석
    - 데이터베이스의 필요한 용도 파악
    - 요구조건 명세 작성
  - ② 개념적 설계
    - 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정
    - 개념스키마 모델링, 트랜잭션 모델링, E-R 모델 수행
  - ③ 논리적 설계
    - 자료들을 컴퓨터에 저장할 수있도록 DBMS 가 지원하는 논리적 자료구조로 변환시키는 과정
    - 논리적 스키마, 트랜잭션의 인터페이스 설계
  - ④ 물리적 설계
    - 논리적 구조로 표현된 데이터를 물리적 구조의 데이터로 변환하는 과정
  - ⑤ 데이터베이스 구현

8. 논리적 설계와 물리적 설계에서 도출된 데이터베이스 스키마를 파일로 생성하는 과정

- 응용 프로그램을 위한 트랜잭션을 작성
- 데이터베이스 접근을 위한 응용 프로그램을 작성

9. 데이터 모델

- 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.
- 구성 요소 : 개체, 속성, 관계
- 모델 종류 : 개념적 데이터모델, 논리적 데이터모델, 물리적데이터모델
- 데이터 모델에 표시할 요소 :
  - ① 구조 : 논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질 표현
  - ② 연산 : 실제 데이터를 처리하는 작업에 대한 명세로 데이터베이스를 조작하는 기본도구
  - ③ 제약조건 : 데이터의 논리적인 제약조건

10. 개념적 데이터 모델

- 사람의 이해를 돕기 위해 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정
- 정보모델이라고도 함

11. 논리적 데이터 모델

- 개념적 구조를 컴퓨터 환경에 맞도록 변환하는 과정
- 데이터 간의 관계 표현에 따라 관계모델, 계층모델, 네트워크 모델로 구분

12. 개체

- 데이터베이스에 표현하려는 것 (테이블이라 생각하면 됨)
- 개념이나 정보 단위 같은 현실 세계의 대상체
- 개체의 구성요소
  - ① 속성=디그리=차수 :  
개체가 가지고 있는 특성 (컬럼), 데이터베이스를 구성하는 가장 작은 논리적 단위
    - 기본속성 : 업무 분석을 통해 정의한 속성
    - 설계속성 : 업무상 존재하지 않고 설계과정에서 도출해내는 속성 (코드 값 생각)
    - 파생속성 : 다른 속성으로부터 계산이나 변형 등에 영향을 받아 발생하는 속성
    - 기본키 속성 : 개체를 유일하게 식별할 수 있는 속성
    - 외래키 속성 : 다른 개체와의 관계에서 포함된 속성
    - 일반 속성 : 개체에 포함되어 있으나 기본키, 외래키에 포함되지 않은 속성
  - ② 개체타입 (레코드 타입) : 속성으로만 기술된 개체의 정의
  - ③ 개체 인스턴스:  
하나의 row, 개체를 구성하고 있는 속성들이 값을 가져 하나의 개체를 나타내는 것
  - ④ 개체세트 : 여러개의 개체 인스턴스

13. 관계

- 개체와 개체 사이의 논리적인 연결
- 관계의 종류
  - ① 종속관계 : 식별과 비식별 관계 존재, 주종 관계를 표현한것
  - ② 중복관계 : 두 개체 사이에 2 번 이상 종속 관계가 발생하는 관계
  - ③ 재귀관계 : 개체가 자기 자신과 관계를 갖는것
  - ④ 배타관계 : 개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계, 배타 AND 관계와 배타 OR 관계로

14. E-R 모델

- 구성 요소 : 개체, 관계, 속성

기 호	의 미
	개체
	속성
	기본키
	관계
	개체 타입과 속성을 연결

### 15. 관계형 데이터베이스

- 2 차원 표를 이용해서 데이터 상호 관계를 정의하는 데이터베이스
- 간결하고 보기 편하다. 다른 데이터베이스로의 변환 용이
- 성능이 떨어짐
- 관계형 데이터 모델 : 2 차원 표를 이용해서 데이터 상호 관계를 정의하는 DB 구조

### 16. 릴레이션 구조

- 속성 : 데이터베이스를 구성하는 가장 작은 논리적 단위, 속성의 수 = 디그리 = 차수
- 튜플 : 릴레이션을 구성하는 행, 튜플의 수 = 카디널리티 = 기수 = 대응수
- 도메인 : 하나의 속성이 취할 수 있는 같은 타입의 원자 값들의 집합
- 릴레이션 스키마 : 릴레이션의 header
- 릴레이션 인스턴스 : 릴레이션의 데이터

### 17. 키

- 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성
- 키의 종류
  - ① 후보키 : 튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합, 유일성 + 최소성
  - ② 기본키 : 후보키 중 선정된 주키,
  - ③ 대체키 : 후보키 - 기본키
  - ④ 슈퍼키 : 속성들의 집합으로 구성된 키, 유일성은 만족하지만 최소성 X
  - ⑤ 외래키 : 다른 릴레이션 기본키를 참조하는 속성

### 18. 무결성

- 데이터베이스에 저장된 데이터 값과 현실 세계의 실제값이 일치하는 정확성
- 무결성 제약조건 : 부정확한 자료가 저장되는 것을 방지하기 위한 제약조건
- 무결성 종류
  - ① 개체 무결성 : 기본키를 구성하는 어떤 속성도 NULL, 중복값을 가질 수 없음
  - ② 참조 무결성 : 외래키 값은 NULL 일수 있음. 참조하는 기본키와 외래키는 동일해야함 (중복 X)
  - ③ 도메인 무결성 : 주어진 속성의 값이 정의된 도메인에 속한 값이어야 함
  - ④ 사용자 정의 무결성 : 속성 값들이 사용자가 정의한 제약조건에 만족되어야 함
  - ⑤ NULL 무결성 : 특정 속성 값이 NULL 이 될 수 없음
  - ⑥ 고유 무결성 : 특정 속성에 대해 각 튜플의 속성이 모두 달라야함
  - ⑦ 키 무결성 : 하나의 릴레이션에는 하나 이상의 키가 존재해야함
  - ⑧ 관계 무결성 : 릴레이션 간의 튜플들 사이의 관계에 대한 적절성 여부를 지정한 규정

### 19. 무결성 강화 방법

- 애플리케이션 : 데이터 생성, 수정, 삭제 시 무결성 검사 코드 추가
- 데이터베이스 트리거 : 트리거 이벤트에 무결성 조건을 실행하는 SQL 추가
- 제약조건 : 데이터베이스에 제약조건을 설정하여 무결성 유지

### 20. 관계 대수

- 원하는 정보를 검색하기 위해 기술하는 절차적 언어
- 순수관계연산자

종류	특징	기호
Select	1. 튜플의 부분집합을 구하여 새로운 릴레이션을 생성하는 연산 2. 튜플을 구하는 것 = 수평 연산	$\delta$ (시그마)
Project	1. 제시된 속성 값만을 추출	$\pi$ (파이)

	2. 연산 결과에 중복이 발생하면 중복 제거 3. 수직 연산	
Join	1. 공통 속성을 가진 두개의 릴레이션을 합쳐 새로운 릴레이션 생성 2. Cartesian Product(교차곱)후 Sele 와 같은 연산	$\bowtie$
Division	$X \supset Y$ 인 릴레이션이 존재할 때 $X \div Y$ 연산	$\div$

- 일반 집합 연산자

종류	특징	기호
합집합 Union	두 릴레이션의 합집합을 구하고 중복 튜플 제거	$\cup$
교집합 Intersection	두 릴레이션의 교집합 연산	$\cap$
차집합 Difference	두 릴레이션의 차집합 연산	$-$
교차곱 Cartesian Product	$E_n$ 릴레이션에 있는 튜플들의 순서 쌍을 구하는 연산	$\times$

21. 관계해석

- 관계 데이터의 연산을 표현하는 방법
- 비절차적 특성을 지닌다

22. 이상

- 테이블에서 데이터 중복이 발생하여 문제가 발생하는 현상
- 이상 종류
  - ① 삽입이상 : 원하지 않는 값들로 인해 삽입이 되지 않는 현상
  - ② 삭제이상 : 연쇄 삭제 발생하는 경우
  - ③ 갱신이상 : 갱신의 실수로 정보에 불일치성 발생

23. 함수적 종속

- $X$  가  $Y$  를 함수적으로 결정한다고 하고  $X \rightarrow Y$  로 표기, 결정자  $\rightarrow$  종속자

24. 정규화

- 테이블의 속성들이 종속적인 관계를 갖고 있을 때 테이블을 무손실 분해하는 과정이다
- 정규화는 중복을 제거하여 삽입, 삭제, 갱신 이상의 발생의 가능성을 줄인다.
- 제 1 정규화- 제 2 정규화 – 제 3 정규화 – BCNF – 제 4 정규화 – 제 5 정규화
- 두부이절다줘(두부이절다조)
  - ① 1 정규화 : 도메인이 원자성 : 도메인을 원자값으로 분해
  - ② 2 정규화 : 부분 함수적 종속 제거
  - ③ 3 정규화 : 이행함수종속제거,
  - ④ BCNF : 결정자이면서 후보키가 아닌 것 제거
  - ⑤ 4 정규화 : 다치 종속 제거
  - ⑥ 5 정규화 : 후보키를 통하지 않는 조인종속제거

25. 반정규화

- 정규화된 데이터 모델을 의도적으로 통합, 중복하여 정규화 원칙을 위배하는 행위

- 효율성증가 및 성능 향상
- 데이터 일관성 및 정합성 저하
- 반정규화 방법
  - ① 테이블 통합
  - ② 테이블 분할
  - ③ 중복테이블 추가 : 작업 효율성 향상
    - 집계테이블, 진행 테이블, 특정 부분만 포함하는 테이블
  - ④ 중복 속성 추가

#### 26. 시스템 카탈로그

- 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스
- 사용자와 시스템 모두 접근 가능

#### 27. 메타 데이터

- 시스템 카탈로그에 저장된 정보
- 메타데이터의 유형
  - 데이터베이스 객체 정보 : 테이블, 인덱스, 뷰
  - 사용자 정보 : 아이디, 패스워드, 접근권한
  - 테이블 제약조건정보
  - 함수, 프로시저, 트리거

#### 28. 데이터 디렉터리

- 데이터 사전에 수록된 데이터에 접근하는데 필요한 정보를 관리하는 시스템
- 시스템만 접근 가능

#### 29. 트랜잭션

- 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산
- 특성
  - ① 원자성 : Commit, Rollback 보장, 데이터베이스에 모두 반영되도록 하던가 반영되지 않도록 복구 하던가
  - ② 일관성 : 성공적으로 끝나면 일관성있는 데이터 베이스 상태로 변환
  - ③ 독립성 : 트랜잭션 실행중 다른 트랜잭션이 끼어들수 없음
  - ④ 영속성 : 영구적으로 반영되어야함

#### 30. CURD 분석

- 프로세스와 테이블간에 CRUD 메트릭스를 만들어서 트랜잭션을 분석하는 것

#### 31. 인덱스

- 데이터 레코드를 빠르게 접근하기 위해 <키값,포인터>쌍으로 구성되는 데이터 구조
- 종류
  - ① 트리기반
  - ② 비트맵 : 인덱스 컬럼의 데이터를 0 또는 1로 변환하여 인덱스로 사용
  - ③ 함수 기반 인덱스
  - ④ 비트맵 조인
  - ⑤ 도메인 : 개발자가 필요한 인덱스를 직접 만들어서 사용

#### 32. 클러스터드 인덱스 : 인덱스 키의 순서에 따라 데이터가 정렬

#### 33. 언 클러스터드 인덱스 : 데이터 정렬되지 않음, 데이터 삽입, 삭제 발생시 순서를 유지하기 위해 사용

#### 34. 뷰

- 테이블로부터 유도된 가상테이블
- 물리적으로 존재 X, 데이터를 안전하게 보호 가능

장점	단점
1. 논리적 데이터 독립성 제공 2. 동일 데이터에 대해 동시에 여러 사용자의 상이한 요구를 지원해줌 3. 데이터 관리 간단, 보안 제공	1. 독립적인 인덱스를 가질수 없음 2. 뷰의 정의를 변경할 수 없음 3. 삽입,삭제,갱신 연산에 제약이 있음

### 35. 클러스터

- 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장방법
- 조회속도 UP, 입력/수정/삭제 속도 BAD
- 데이터 분포도가 넓을수록 유리 ->저장공간 절약 가능
- 단일 테이블 클러스터링 : 처리범위가 넓은 경우
- 다중 테이블 클러스터링 : 조인이 많이 발생하는 경우

### 36. 파티션

- 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것

장점	단점
1. 액세스 범위를 줄여 쿼리 성능 향상 2. 분산저장 -> 디스크 성능 향상	1. 관리 필요 2. 조인 비용 증가 3. 작은 테이블에 파티셔닝을 하면 성능 저하됨

- 파티션 종류 (범해조)
  - ① 범위 분할 : 지정한 열의 값을 기준을 분할
  - ② 해시 분할 : 해시함수를 적용한 결과 값에 따라 분할, 특정 데이터가 어디있는지 판단 불가능
  - ③ 조합 분할 : 범위 분할 후 해시함수를 적용, 범위분할 파티션이 너무 클경우 사용

### 37. 분산데이터베이스

- 하나의 시스템에 속하지만 물리적으로는 여러 개의 사이트에 분산된 데이터베이스
- 목표 (위중병장)
  - ① 위치 투명성 : 실제 위치를 알필요 없음
  - ② 중복 투명성 : 실제로 중복되어 있더라도 사용자는 하나의 데이터만 존재하듯이 사용
  - ③ 병행 투명성 : 다수의 트랜잭션들이 동시에 실행되도 결과는 투명
  - ④ 장애 투명성
- 설계 방법
  - ① 테이블 위치 분산
  - ② 분할 : 테이블의 데이터를 분할하여 분산
    - 분할 규칙 : 완전성, 재구성, 상호중첩배제
    - 수평분할, 수직분할
  - ③ 할당
    - 동일한 분할을 여러 개의 서버에 생성하는 분산 방법

### 38. 데이터베이스 이중화

- 동일한 데이터베이스를 복제하여 관리하는 것
- 이중화 분류
  - ① EAGER 기법 : 변경이 발생하면 모든 데이터베이스에 즉시 전달하여 수정
  - ② LAZY 기법 : 트랜잭션의 수행이 종료되면 각 데이터베이스에 전달되는 기법
- 이중화 구성방법
  - ① 활동 대기 : 하나 활동 하나 대기
  - ② 활동활동 : 두개의 서비스가 활동하다가 장애발생시 다른 DB 가 서비스 제공

39. 클러스터링 : 두대 이상의 서버를 하나의 서버처럼 운영하는 기술
- 고가용 클러스터링 : 장애 발생하면 다른 서버가 받아 처리
  - 병렬 클러스터링 : 전체 처리율을 높이기 위해 두개의 서버가 운영, 분산처리
40. RTO (Recovery Time Objective)
- 복구되어 가동될 때 까지의 소요시간을 의미
41. RPO (Recovery Point Objective)
- 데이터를 복구 할수 있는 기준점을 의미
42. 서버의 종류
- DAS : 직접연결
  - NAS :네트워크로 연결
  - SAN : 둘다 사용
43. 트리
- 정점과 선분으로 구성
  - 사이클로 이뤄지지 않도록 구성된 그래프 형태
  - 구성 요소
    - ① 노드
    - ② 근노드 : 제일 상위 노드
    - ③ 디그리(차수) : 각 노드에서 뻗어나온 가지수
    - ④ 단말노드 = 잎노드 : 자식이 없는 노드
    - ⑤ 비단말 노드 : 자식이 있는 노드
    - ⑥ 조상노드 : 해당 노드에서 근노드까지 이르기까지 있는 노드
    - ⑦ 자식 노드 : 노드의 다음 레벨의 노드들
    - ⑧ 부모 노드
    - ⑨ 형제노드
    - ⑩ 트리의 디그리 = 트리의 차수 : 노드들의 디그리중 가장 많은 수
44. 이진트리
- 차수가 2 이하인 노드들로 구성된 트리
  - 운행방법 (root 의 위치를 생각)
    - ① 전위순행(Preorder) : Root-L-R
    - ② 중위순행(Inorder) : L-Root-R
    - ③ 후위순행(Postorder) : L-R-Root
45. 삽입정렬
- 이미 순서화된 파일에 레코드를 순서에 맞게 삽입시켜 정렬
  - 시간복잡도 :  $O(n^2)$
46. 선택정렬
- 최소값을 찾아 첫번째 레코드 위치에 놓고 다시 최소값을 찾아 정렬하는 방식
  - $O(n^2)$
47. 버블 정렬
- 인접한 두 개의 레코드 키 값을 비교하여 서로 교환하는 정렬 방식
  - $O(n^2)$
48. 쉘 정렬
- 삽입 정렬을 확장한 개념
  - 매개변수의 값으로 서브파일을 구성하고, 각 서브파일을 Insertion 하는 방식으로 순서 배열



49. 퀵정렬

- 키를 기준으로 작은 값은 왼쪽 큰값은 오른쪽
- $O(n \log 2n)$

50. 힙정렬

- 전이진 트리를 이용한 정렬 방식
- $O(n \log 2n)$

51. 2-way 합병 정렬

- 정렬되어있는 두개의 파일을 하나의 파일로 합병하는 정렬 방식

52. 기수 정렬

- 큐를 이용하여 자릿수 별로 정렬하는 방식

53. XML

- 특수한 목적을 갖는 마크업 언어를 만드는 세 사용되는 다목적 마크업 언어이다

54. SOAP

- 네트워크 상에서 HTTP/HTTPS, SMTP 등을 이용하여 XML을 교환하기 위한 통신 규약

55. WSDL

- 웹 서비스와 관련된 서식이나 프로토콜 등을 표준적인방법으로 기술하고 게시하기 위한 언어

## 4 장. 서버 프로그램 구현

1. 소프트웨어 아키텍처
  - 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템 구조
2. 모듈화
  - 시스템의 기능들을 모듈 단위로 나누는 것
3. 추상화
  - 전체적이고 포괄적인 개념을 설계한 후 점차 구체화 하는 것
  - 추상화 유형(과제데)
    - ① 과정 추상화 : 과정 정의 X, 흐름만 파악할 수 있도록 설계
    - ② 데이터 추상화 : 데이터 추상화 X, 데이터 구조를 대표할수 있는 표현으로 대체
    - ③ 제어추상화 : 절차나 방법 정의 X, 대표할 수 있는 표현으로 대체

### 4. 단계적 분해

- 상위->하위, 상위 개념부터 시작해서 하위 개념으로 구체화시키는 분할 기법

	상위 설계	하위 설계
별칭	아키텍처 설계, 예비 설계	모듈 설계, 상세 설계
설계대상	시스템의 전반적인 구조	시스템의 내부 구조 및 행위
세부목록	구조, DB, 인터페이스	컴포넌트, 자료구조, 알고리즘

5. 정보은닉
  - 모듈 내부에 포함된 자료들의 정보가 감춰져 있어 다른 모듈이 접근하거나 변경 X
  - 모듈은 독립적으로 수행 -> 하나의 모듈이 변경되도 다른 모듈에 영향 X
6. 협약에 의한 설계 (선결불)
  - 선행조건, 결과조건, 불변조건(실행하는 동안 항상 만족)
7. 품질 평가요소의 종류
  - 서비스 : 성능, 보안, 가용성, 기능성, 사용성, 변경용이성, 확장성
  - 비즈니스 : 비용 혜택, 시스템 수명, 목표 시장
  - 아키텍처 : 구축 가능성, 완결성, 시험성
8. 범용성
  - 개발 언어 선정 시 다른 개발 사례가 충분히 존재하고, 이미 여러 곳에서 사용하고 있는 지를 판단하는 기준
9. 소프트 아키텍처의 설계과정
  - 설계 목표 설정 - 시스템 타입 설정 - 아키텍처 패턴 - 서브시스템 구체화 - 구현
  - 아키텍처 패턴
    - ① 레이어 패턴
      - : 시스템을 계층으로 구분, 하위계층(클라이언트)과 상위계층(시스템제공자) 존재
      - : OSI 참조모델
    - ② 클라이언트 - 서버패턴
      - : 하나의 서버에 여러 클라이언트로 구성
      - : 서버와 클라이언트는 TCP/IP 로 데이터 전송
    - ③ 파이프 필터 패턴
      - : 서브시스템이 입력을 받아 처리하고 결과를 다음 서브시스템으로 전달
      - : 데이터 변환, 버퍼링, 동기화 등에 사용, Unix Shell 이 대표적
    - ④ 모델 뷰 컨트롤러 (MVC)
      - : 대화형 애플리케이션에 적합

⑤ Peer to peer

: 멀티 스레딩 방식을 사용

: 하나의 컴포넌트가 서비스를호출하는 클라이언트가 될수도 있고,  
서비스를 제공하는 서버가 될수도 있음

10. 객체지향

- 각 요소들을 객체로 만들어서 이를 조립하여 하나의 소프트웨어를 개발하는 방법
- 객체지향의 특징 : 캡슐화, 상속, 다형성, 연관성
- 구성 요소 (객클메)
  - ① 객체 : 데이터와 이를 처리하기 위한 함수를 묶어놓은 소프트웨어 모듈
  - ② 클래스 : 공통된 속성과 연산을 갖는 집합
  - ③ 메시지 : 상호작용에 이용되는 수단

11. 다형성

- 응용프로그램 상에서 하나의 함수나 연산자가 두개 이상의 서로 다른 클래스의 인스턴스들을 같은 클래스에 속한 인스턴스처럼 수행 할수 있도록 하는 것

12. 연관성 : 두개 이상의 객체들이 상호 참조하는 관계

- 연관성 종류
  - ① Is member of (연관화)
  - ② Is instance of (분류화)
  - ③ Is part of (집단화)
  - ④ Is a (일반화)

13. 객체지향 분석

- 사용자의 요구사항과 관련된 객체, 속성 연산 등을 정의하여 모델링하는 것

14. 객체지향 분석 방법론

- 럼바우 : 객동기 (객체, 동적, 기능)
  - ① 객체모델링
  - ② 동적모델링
  - ③ 기능모델링 : 자료 흐름을 중심으로 처리과정을 표현한 모델
- 부치 : 미시적 거시적 개발
- Jacobson : 유스케이스
- Coad 와 yourdon : E-R 다이어그램사용
- Wirfs – brocks : 고객 명세서 평가

15. 객체지향 설계 원칙 (SOLID)

- SRP : Single Responsibility : 단일 책임
- OCP : Open Closed : 개방 – 폐쇄
- LSP : Liskov substitution : 리스코프 치환 : 자식클래스는 부모클래스의 기능을 수행해야함
- ISP : Interface Separate : 인터페이스 분리
- DIP : Dependency Inversion : 의존 역전

16. 결합도

- 모듈간에 상호 의존하는 정도, 두 모듈 사이의 연관관계
- 결합도가 낮을수록 좋음
- 자료 – 스템프 – 제어 – 외부 – 공통 – 내용
- 자료를 스템프로 찍어 제어하고 외부에 공통 내용을 알린다.

17. 응집도

- 모듈 내부에서 서로 관련되어 있는 정도

- 응집도가 높을수록 좋다
- 우연적 - 논리적 - 시간전 - 절차적 - 통신적(교환) - 순차적 - 기능적
- 우리는 시간이 되면 절차에 맞게 신수능을 본다

#### 18. 팬인

- 하나의 모듈을 제어하는 모듈 수
- 높을수록 재사용 측면에서 설계가 높은 것

#### 19. 팬 아웃

- 하나의 모듈이 제어하는 모듈 수

#### 20. N-S 차트

- 논리의 기술에 중점을 두고 도형을 이용해 표현하는 방법
- 연속, 선택 및 다중 선택, 반복 등의 제어 논리 구조로 표현한다. 화살표 이용 X

#### 21. 단위 모듈

- 한가지 동작을 수행하는 기능들을 모듈로 구현한 것
- 독립적인 컴파일 가능, 다른 모듈에 호출되거나 삽입되기도 한다

#### 22. 공통 모듈

- 여러 프로그램에서 공통으로 사용하는 모듈
- 명세기법 종류 (정명완일추)
  - ① 정확성, 명확성, 완정성
  - ② 일관성 : 공통 기능들 간 상호 충돌이 발생하지 않도록 작성하는 기법
  - ③ 추적성 : 출처, 관련 시스템 등의 관계를 파악 할 수 있도록 작성해야 한다.

#### 23. 재사용

- 이미 개발된 기능들을 새로운 시스템이나 기능에 사용하기 위해 최적화 하는 것
- 규모에 따른 분류
  - ① 함수와 객체 : 클래스 단위
  - ② 컴포넌트 : 컴포넌트 수정 없이 인터페이스를 이용한 방법
  - ③ 애플리케이션 : 공통된 기능을 제공하는 어플리케이션을 공유 하는 방식

#### 24. 코드

- 주요기능 : 식별, 분류, 배열, 표준화, 간소화
- 코드의 종류
  - ① 순차코드 : 일정 기준에 따라 최초의 자료부터 순차적으로 일련번호를 부여
  - ② 블록코드 : 공통성 있는 것끼리 블록으로 나누고 블록 내에서 일련번호를 부여
  - ③ 10 진코드 : 필요한 만큼 10 진 분할을 이용하여 일련번호 부여
  - ④ 그룹 분류 코드 : 일정 기준에 따라 대/중/소로 구분하여 그룹안에서 코드를 부여
  - ⑤ 연상코드 : 연상되는 명칭이나 약호를 사용하여 코드부여
  - ⑥ 표의 코드 부여 : 대상의 물리적 수치를 이용하여 코드 부여
  - ⑦ 합성코드 : 두개의 코드를 합성하여 코드 부여

#### 25. GOF (Gang of Four)

- 23 가지 디자인 패턴을 생성,구조, 행위로 나눔

#### 26. 디자인 패턴

- 디자인 패턴의 구성요소 : 문제 및 배경, 샘플코드, 사례
- 생성패턴
  - ① 클래스나 객체의 생성과 참조과정을 정의한패턴
    - Abstract Factory (추상): 그룹으로 묶어서 한버네 수정, Factory method 를 사용
    - Factory method : 서브클래스에 위임하는 패턴

- Builder : 생성단계를 캡슐화하여 공정을 동일하게 이용
- Prototype
- Singleton
- 구조패턴
  - ① 여러 객체를 모아 구조화 시키는 패턴, 새로운 기능을 제공하기도 함
    - Adapter : 코드변환
    - Bridge : 기능클래스와 구현클래스를 분리하여 결합도를 낮춤
    - Composite : 트리구조, 복합객체와 단일객체를 다룰 수 있음
    - Decorator : 오버라이딩, 소스를 변경하지 않고 기능을 확장하도록 하는 패턴  
(데코 : 꾸미는거)
    - Façade
    - Flyweight : 메모리를 가볍게 유지하게 위한 패턴
    - Proxy : 네트워크 통신을 이용한
- 행위패턴
  - ① 큰 작업을 여러 개의 객체로 분리한 방법
  - ② 알고리즘 수행에 주로 이용
  - ③ 객체 사이의 결합도를 최소화
    - Chain of Responsibility : 여러 개의 처리기를 두고 순서대로 처리해가는 패턴
    - Command : 재사용성이 높은 클래스를 설계하는 패턴
    - Interpreter : 언어의 문법을 정의하고 해석
    - Iterator : 동일한 인터페이스를 통해 접근할 수 있도록 하는 패턴
    - Observer : 한 객체의 상태가 바뀌면 그 객체에 의존하고 있는 다른 객체들도 자동으로 바뀜
    - Memento : ctrl + z : 그 문신아저씨가 기억하는 방법 암튼 기억
    - Mediator, Memento, State, Strategy, Template Method, Visitor

## 27. 배치 프로그램

- 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하도록 만든 프로그램
- 배치 프로그램 필수 요소
  - ① 대용량 데이터, 자동화, 견고성, 안정성/신뢰성, 성능

## 28. 배치 스케줄러 = 잡 스케줄러

- 배치 프로그램이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구
- 배치 스케줄러 종류
  - ① 스프링 배치
  - ② Quartz : 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄처리를 위한 도구
  - ③ Cron : 리눅스의 기본 스케줄러
    - Crontab
      - A. 분 시 일 월 요일 명령어 순으로 작성 (요일은 0: 일요일 ~ 6 :토요일 )
      - B. \* \* \* \*/root/com\_2.sh -> 매분 매시 매일 매요일 마다 root/com\_2.sh 를 실행
      - C. 30 \*/3 \* \* \* /root/com\_2.sh -> 매요일 매월 매일 3 시간마다 30 분에 com\_2.sh 가 실행
      - D. \* 18-23 20 \* \* /root.com\_2.sh -> 매요일 매월 20 일 18-23 시 동안 매분 실행

## 29. 테스트 케이스

- 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지 확인하기 위함
- 설계된 입력값, 실행조건, 기대결과 등으로 구성된 테스트 항목에 대한 명세서

## 30. Shared Memory

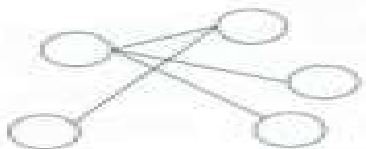

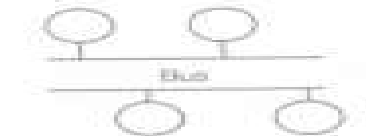
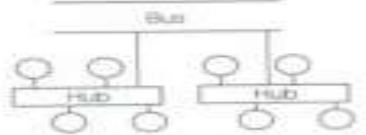
- IPC 에서 공유된 메모리를 이용하여 둘 이상의 프로세스가 통신할 수 있도록 기능을 제공

### 31. API

- 소프트웨어 개발에서 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해놓은 인터페이스

## 5 장. 인터페이스 구현

1. 인터페이스 요구사항 검증 수행순서
  - 요구사항 검토 계획 수립 - 검토 및 오류수정 - 베이스라인 설정
2. 인터페이스 요구사항 명세서의 구성요소
  - 연계방식, 인터페이스 주기, 연계대상 시스템, 인터페이스 이름
3. 요구사항 검증 방법
  - 요구사항 검토 방법
    - ① 동료검토
    - ② 워크스루 : 회의를 통해 결함 발견
    - ③ 인스펙션 : 전문가들이 요구사항 검토
    - ④ 프로토타이핑
    - ⑤ 테스트설계
    - ⑥ CASE 도구 활용
4. 인터페이스 요구사항 검증의 주요항목
  - 완전성, 일관성, 명확성, 기능성, 검증가능성, 추적가능성, 변경용이성
5. 미들웨어
  - 응용프로그램과 운영체제 사이에서 서비스를 제공하는 소프트웨어
  - 미들웨어의 종류
    - ① DB
    - ② RPC(Remote Procedure Call) : 원격 프로시저를 호출하는 미들웨어
    - ③ MOM(Message Oriented Middleware) : 비동기형 메시지 전달,  
분산시스템에서 데이터 동기화를 위해 사용
    - ④ TP-Monitor : 트랜잭션 처리를 감시하는 미들웨어, 항공기 예약에서 사용
    - ⑤ ORB : 코바의 표준 스펙을 구현한 객체 지향 미들웨어
    - ⑥ WAS : 동적인 콘텐츠를 처리하기 위한 미들웨어
6. 모듈 연계
  - EAI

유형	특징	그림
Point – to – point	<ul style="list-style-type: none"> <li>- 중간에 미들웨어를 두지 않고 연결</li> <li>- 변경 재사용이 어려움</li> </ul>	
Hub & Spoke	<ul style="list-style-type: none"> <li>- 중앙 집중적 방식</li> <li>- 모든 데이터 전송을 보장</li> <li>- 확장 및 유지보수 용이</li> <li>- 허브 장애 시 전체 시스템 영향</li> </ul>	
Message Bus (ESB)	<ul style="list-style-type: none"> <li>- 미들웨어(버스)를 두어 처리</li> <li>- 미들웨어를 통한 통합 가능</li> <li>- 확장성 대용량 처리 가능</li> </ul>	
HyBrid	<ul style="list-style-type: none"> <li>- 그룹 내에선 Hub&amp;Spoke 방식/ 그룹 간에는 Message Bus 방식 사용</li> <li>- 표준 통합 기술</li> <li>- 데이터 병목현상 최소화 가능</li> </ul>	

- ESB

- ① Bus 중심으로 각각 프로토콜이 호환되게끔 변환이 가능하다
- ② 애플리케이션 간 표준 기반의 인터페이스를 제공하는 솔루션
- 웹서비스
  - ① 네트워크 정보를 표준화된 서비스 형태로 만들어서 공유하는 기술

#### 7. AJAX

- 클라이언트와 서버 간에 XML 데이터를 주고받는 비동기 통신 기술
- 웹페이지 일부 영역만 업데이트 가능

#### 8. 인터페이스 보안 기능 적용

- 네트워크 영역
  - ① 스니핑 등을 이용한 데이터 탈취 및 변조 위협을 방지하기 위해 네트워크 트래픽에 대한 암호화 설정
  - ② SSL: TCP/IP 계층과 애플리케이션 계층 사이에서 인증 프로토콜
  - ③ S-HTTP : 클라이언트와 서버 간에 전송되는 모든 메시지를 암호화하는 프로토콜
  - ④ IPSec : IP 패킷 단위의 데이터 변조 방지 및 은닉 기능 제공
- 어플리케이션 영역
  - ① 코드 상의 보안 취약점을 보완
- 데이터베이스
  - ① 접근권한과 프로시저, 트리거 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능 적용
  - ② 암호화 익명화 사용

#### 9. 데이터 무결성 검사 도구

- 인터페이스 보안 취약점 분석, 파일 변경 유무 확인
- Tripwire, Slipwire, AIDE, Samhain, Claymore, Fcheck

#### 10. 인터페이스 구현 검증 도구

도구	기능
xUnit	Java, C++, Net 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	컴포넌트 <b>재사용</b> 등 다양한 환경을 지원하는 테스트 프레임워크 크로스 플랫폼이나 분산 소프트웨어에서 테스트 환경 가능
FitNesse	웹 <b>기반</b> 테스트 케이스 설계, 실행, 결과 등을 지원하는 테스트 프레임워크
NTAF	STAF + FitNesse
Selenium	다양한 브라우저 및 개발 언어 지원
Watir	Ruby 를 사용

#### 11. 인터페이스 구현 감시도구

- 동작 상태는 APM 을 사용하여 감시 가능
- APM
  - 1. 애플리케이션 성능 관리를 위한 도구
  - 2. 스카우터 : OS 자원에 대한 모니터링
  - 3. 제니퍼 : 애플리케이션 전 단계에 걸쳐 성능을 모니터링하고 분석하는 소프트웨어

#### 12. 인터페이스 연계 기술

DB Link	DB 에서 제공하는 DB Link 객체를 이용하는 방식
API/OPEN API	송신 시스템의 DB 에서 데이터를 읽어와 제공하는 애플리케이션 프로그래밍 인터페이스
연계솔루션	EAI 서버와 송수신 시스템에 설치되는 클라이언트를 이용하는 방식
소켓	서버가 통신을 위한 소켓을 생성하여 포트를 할당하고 통신하는 네트워크 기술
웹서비스	WSDL, UDDI, SOAP 프로토콜을 이용하여 연계하는 서비스



## 6 장. 화면설계

### 1. UI

- 사용자와 시스템 간의 상호작용이 이뤄지도록 도와주는 장치나 소프트웨어
- UI 구분 : CLI, GUI, NUI
- UI 기본 원칙
  - ① 직관성 : 누구나 쉽게 이해하고 사용할 수 있어야 함
  - ② 유효성 : 사용자의 목적을 정확하고 완벽하게 달성해야 함
  - ③ 학습성 : 누구나 쉽게 배우고 익힐 수있어야함
  - ④ 유연성 : 사용자의 요구사항을 최대한 수용하고 실수를 최소화 함

### 2. 와이어 프레임

- 페이지에 대한 대략적인 레이아웃을 설계하는 도구
- Ex) 파워포인트, 일러스트, 포토샵

### 3. 목업

- 실제 화면과 유사하게 만든 정적인 형태의 모형
- Ex) 파워 목업, 발사믹 목업

### 4. 스토리보드

- 와이어프레임에 대한 설명, 페이지 간 이동 흐름 등을 추가한 문서
- Ex) 파워포인트, Axure

### 5. 프로토타입 종류

- 페이퍼 프로토타입 : 직접 손으로 작성하는 아날로그적 방법, 제작 기간이 짧을 때 사용
- 디지털 프로토타입 : 파워포인트 등의 프로그램 사용, 산출물이 필요한 경우

### 6. 유스케이스

- 사용자의 요구사항을 기능 단위로 표현

### 7. 소프트웨어 품질 표준

ISO/IEC 9126	소프트웨어 품질 특성과 평가를 위한 국제 표준
ISO/IEC 25010	ISO/IEC 9126 에 호환성과 보안성을 강화
ISO/IEC 12119	패키지 소프트웨어의 품질 요구사항 및 테스트를 위한 국제표준
ISO/IEC 14598	평가 필요절차를 규정한 표준

### 8. ISO/IEC 9126 품질 특성

- 기능성 : 기능을 제공하는지 여부, 보안성,
- 신뢰성 : 오류없이 실행, 성숙성, 고장 허용성, 회복성
- 사용성 : 향후 다시 사용하고 싶은 정도, 이해성, 학습성, 운용성, 친밀성
- 효율성 : 시간 효율성, 자원효율성
- 유지보수성 : 확장 용이, 분석성, 변경성, 안정성, 시험성
- 이식성 : 다른 환경에서 적용 가능 여부, 적용성, 설치성, 대체성, 공존성

### 9. UI 스타일 가이드 작성 순서

- 구동환경정의 - 레이아웃 정의 - 네비게이션 정의- 기능정의 -구성요소 정의
- 구렛나루 내기 하자구

## 7 장. 애플리케이션 테스트 관리

### 1. 애플리케이션 테스트 기본원리

- ① 완벽한 테스트 불가능 : 소프트웨어에 결함이 없다고 증명할 수 없음
- ② 파레토 법칙 : 20% 해당하는 코드에서 전체 결함 80%가 발견
- ③ 살충제 패러독스 : 동일한 테스트 케이스로 반복하면 더 이상 결함 발견안됨
- ④ 테스트는 정황 의존 : 정황에 따라 다르게 테스트 해야함
- ⑤ 오류-부재의 귀변 : 오류가 없어도 요구사항을 만족하지 못하면 품질은 높지 않음
- ⑥ 테스트와 위험은 반비례
- ⑦ 테스트 점진적 확대
- ⑧ 테스트의 별도 팀 수행

### 2. 프로그램 실행 여부에 따른 테스트

- ① 정적 테스트 : 워크스루, 인스펙션, 코드검사
- ② 동적 테스트 : 블랙박스 테스트, 화이트박스 테스트

### 3. 테스트 기반에 따른 테스트 ( 테스트 구경은 명구경이다!)

- ① 명세기반 테스트 : 요구사항에 대한 명세를 테스트케이스로 구현
  - ex) 동등분할, 경계 값 분석
- ② 구조기반 테스트 : 논리 흐름에 따라 테스트 케이스 구현
  - ex) 구문기반, 결정 기반, 조건기반
- ③ 경험기반 : 테스터의 경험을 기반으로
  - ex) 에러추정, 체크리스트, 탐색적 테스트

### 4. 시각에 따른 테스트

- ① 검증 테스트 : 개발자 기반
- ② 확인 테스트 : 사용자 기반

### 5. 목적에 따른 테스트

- ① 회복 테스트 : 여러가지 결함에 실패 후 복구 확인 테스트
- ② 안전 테스트 : 불법 침입으로부터 안전한지 테스트
- ③ 강도 테스트 : 과도한 정보량 혹은 빈도에 대한 테스트
- ④ 성능 테스트 : 실시간 성능이나 효율성 진단 테스트
- ⑤ 구조 테스트 : 소스 코드의 복잡도 등을 평가
- ⑥ 회귀 테스트 : 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트
- ⑦ 병행 테스트 : 기존 소프트웨어와의 병행 테스트

### 6. 화이트박스 테스트 종류

- ① 기초경로 검사
- ② 제어구조검사 : 제어구조검사, 로프검사, 데이터 흐름검사

### 7. 화이트박스 테스트 검증 기준

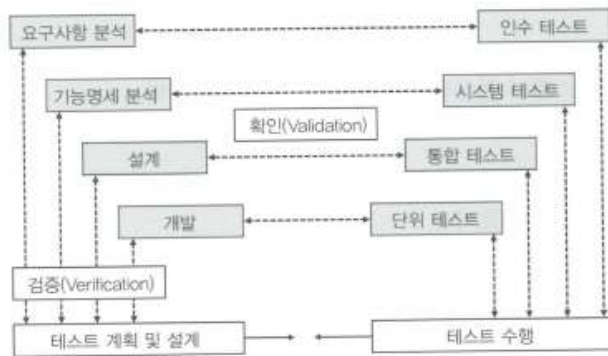
- ① 문장 검증 기준 : 모든 구문이 한번 이상 수행
- ② 분기 검증 기준 : 모든 조건문이 한번 이상 수행
- ③ 조건 검증 기준 : 모든 조건문의 True 와 False 가 각 한번 이상 수행
- ④ 분기/조건 기준 : 모든 조건문과 각 조건문에 포함된 개별 조건식의 결과가 True/False 경우 수행

### 8. 블랙박스 테스트 검증 종류

- : 소프트웨어 인터페이스에서 실시되는 기능테스트,, 소프트웨어의 기능이 의도대로 작동되는지 확인
- ① 동치분할 검사 : 입력하고 맞는 결과값이 출력되는지 확인
- ② 경계값 분석 : 경계값에서 오류가 발생할 확률이 높기 때문에 하는 확인

- ③ 원인-효과 그래프 검사 : 입력데이터 간의 관계와 출력에 영향을 미치는 상황을 분석한 다음 테스트 케이스 선정하여 검사
- ④ 오류 예측 : 과거의 경험으로
- ⑤ 비교검사 : 여러버전의 프로그램에 동일한 테스트 자료 제공하여 결과값 비교

9. V 모델 테스트 순서 : 요기설구-단통시인, 단위-통합-시스템-인수



\*확인 : 개발된 소프트웨어가 고객의 요구사항에 맞게 구현되어있는지

\*검증 : roqkfehls 소프트웨어가 개발자 입장에서 명세서에 맞게 만들어졌는지 보는 것

① 인수테스트

- 사용자인수테스트
- 운영상의 인수테스트 : 백업/복원 시스템, 재난 복구, 사용자 관리 등을 확인
- 계약 인수 테스트 : 계약 상의 조건을 준수하는지
- 규정 인수
- 알파테스트 : 개발자의 장소에서 사용자가 개발자앞에서 행하는 테스트
- 베타테스트 : 사용자가 여러명의 사용자 앞에서 행하는 테스트

10. 통합테스트

① 비점진적 통합테스트 : 미리 결합되어 있는 프로그램을 전체 테스트 하는 방법

② 점진적 통합테스트

- 하향식통합테스트 : 상위모듈 -> 하위모듈 방향으로 테스트
  - 주요제어 모듈의 종속 모듈들은 스텝으로 대체하여 테스트
  - 회귀 테스트 진행
- 상향식통합테스트 : 하위 -> 상위모듈 방향으로 테스트
  - 하위모듈들은 클러스터로 결합
  - 드라이버를 사용
  - 테스트 완료되면 클러스터는 프로그램 구조의 상위로 이동하여 결합
  - 드라이버는 실제모듈로 대체
- 혼합식 통합테스트
- 회귀 테스트

\* 드라이버와 스텝 : 모듈개발이 안된 경우 실행

- ① 드라이버 : 실행을 가능하게 하는 가동기
- ② 스텝 : 개발되지 않은 가짜 모듈

11. 테스트 프로세스

- 테스트계획 - 테스트분석 - 시나리오작성 - 테스트 수행 - 결과 평가 - 결함추적 및 관리

- ① 테스트 케이스 : 테스트 항목에 대한 명세서
- ② 테스트 시나리오 : 테스트 케이스를 순서에 따라 여러 개의 테스트 케이스를 묶은 집합

③ 테스트 오라클 : 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참값을 대입하여 비교

- 제한된 검증
- 수학적 기법
- 자동화 기능
- 테스트 오라클 종류

- ① 참 오라클
- ② 샘플링 오라클 : 경계값 및 구간별 예상값 결과 작성시 사용하는 오라클
- ③ 추정 오라클
- ④ 일관성 검사 오라클

## 12. 결함관리 프로세스 (에에에-결결결결)

- 에러발견 - 에러등록 - 에러분석- 결함확정 - 결함할당 - 결함조치 - 결함 조치 검토

## 13. 시간 복잡도 : 프로세스가 수행하는 연산횟수를 수치화한 것

- ① 빅오 표기법 : 알고리즘 실행시간이 최악일 때 를 표기하는 방법
- ② 세타 표기법 : 알고리즘 실행시간이 평균
- ③ 오메가 표기법 : 알고리즘 실행시간 최상

## 14. 순환 복잡도 ( 맥케이브 순환도) : 논리적인 복잡도를 측정하기 위한

- ①  $V = E - N + 2$  ( E: 화살표, N 노드수 )
- ② 인접 행렬로 찾는 복잡도

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{array}{l} 2-1=1 \\ 2-1=1 \\ 1-1=0 \\ 1-1=0 \\ 1-1=0 \\ 1-1=0 \\ \text{무시} \end{array} \begin{array}{l} + \\ \\ \\ \\ \\ \\ \hline \end{array}$$

$2+1=3$  (복잡도)

## 15. 나쁜 코드

- ① 스파게티 코드
- ② 외계인 코드 : 오래되어 참고문서 또는 개발자가 없어 유지보수가 힘든 코드

## 16. 클린코드 작성원칙

- ① 가독성
- ② 단순성
- ③ 의존성 매제 : 코드가 다른 모듈에 미치는 영향을 최소화함
- ④ 중복성 최소화
- ⑤ 추상화

## 17. 코드 최적화 유형

- ① 클래스 분할 배치 : 응집도를 높이고 크기를 작게
- ② 느슨한 결합 : 추상화된 자료구조와 메소드를 사용하여 의존성 최소화

## 18. 소스코드 품질 분석 도구

- ① 정적 분석도구 : 코드 실행 X,  
ex ) pmd , cppcheck, checkStyle, ccm,
- ② 동적 분석 도구 : 코드 실행, 메모리 누수,, 스레드 결합 등을 분석  
ex ) Avalanche, Valgrind

## 19. 테스트 하네스

- 테스트 자동화 도구 중 하나

- 애플리케이션의 컴포넌트 및 모듈을 테스트 하는 환경의 일부분
- 테스트를 지원하기 위해 생성된 코드와 데이터

#### 20. 결함관리 측정 지표

- ① 결함 분포 : 컴포넌트의 특정 속성에 해당하는 결함 수 측정
- ② 결함 추세 : 테스트 진행 시간에 따른 결함 수의 추이 분석
- ③ 결함 에이징 : 특정 결함 상태로 지속되는 시간 측정

#### 21. 애플리케이션 성능

- 사용자가 요구한 기능에 대해 최소한의 자원을 사용하여 최대한 많은 기능을 처리하는 정도
- 애플리케이션을 측정하기 위한 지표
  - ① 처리량, 응답속도, 경과시간, 자원 사용률

## 9 장. 소프트웨어 개발 보안 구축

### 1. Secure SDLC

- SDLC: 소프트웨어 개발 생명주기
- Secure SDLC : SLDC에 보안 강화를 위한 프로세스를 포함
- Secure SDLC 방법론
  - ① CLASP : SDLC 초기 단계에서 보안을 강화하기 위해 개발된 방법론
  - ② SDL : 마이크로소프트 사에서 개발한 SDLC를 개선한 방법론
  - ③ Seven TouchPoint : SW 보안 모범 사례를 통합한 소프트웨어 개발 보안 방법론
- SDLC 보안 활동 단계
  - ① 요구사항 분석 - 설계 - 구현 - 테스트 - 유지보수

### 2. 소프트웨어 개발 보안 요소 (기무가 : 3요소)

- 기밀성(인가된 사람들에게만 접근 허용), 무결성(인가된 사람들만 수정), 가용성(인가받은 사용자는 언제든지 사용 가능),
- 인증, 부인 방지(수신 증거 제공)

### 3. 시큐어 코딩

- 소프트웨어의 구현 단계에서 발생할 수 있는 보안 취약점들을 최소화 하기 위해 보안 요소들을 고려하여 코드를 구현한 것.
- 안정성과 신뢰성 확보

### 4. 입력 데이터 검증

- 입력 데이터로 인해 발생하는 문제들을 예방하기 점검 항목들
- SQL 삽입 : 웹 응용프로그램에 SQL을 삽입하여 내부 DB 데이터 유출
- 경로조작 및 자원 사입 : 데이터 입력경로를 조작하여 서버 자원 수정 삭제
- 크로스사이트 스크립팅 : 악의적 스크립트 삽입
- 운영체제 명령어 삽입 : 외부 입력값을 통해 권한을 탈취하거나 시스템 장애 유발
- 위험한 형식 파일 업로드 : 스크립트 파일을 업로드해서 시스템에 손상
- 신뢰되지 않은 URL 주소로 자동 접속 : 피싱사이트로 유도하는 보안 약점
- 메모리 버퍼 오버플로우 : 할당된 메모리 범위를 넘어선 위치에서 자료를 읽거나 쓰려할 때

### 5. 보안 기능

- ① 적절한 인증 없이 중요기능 허용
- ② 부적절한 인가 : 접근 제어 기능이 없는 실행경로를 통해 정보를 탈취 할 수 있음
- ③ 중요한 자원에 대한 잘못된 권한 설정
- ④ 취약한 암호화 알고리즘 사용
- ⑤ 중요정보 평문 저장 및 전송
- ⑥ 하드코드 된 암호화 키

### 6. 암호 알고리즘



## 7. 해시암호의 종류

- ① SHA 시리즈 : NSA이 설계, 미국 국립 표준 기술 연구소에 의해 발표
- ② MD5 : 블록크기가 512 비트, 키 길이는 128

## 8. 양방향 알고리즘 종류

- ① RSA : 큰 숫자를 소인수 분해하기 어렵다는 것에 기반하여 만들어짐
- ② SEED : 한국인터넷진흥원에서 개발한 블록 암호화 알고리즘  
:블록크기 128비트, 키 길이에 따라 128, 256으로 분류
- ③ DES : 개인키 암호화 알고리즘으로 64비트 블록크기와 56비트의 키 길이를 갖는다.  
: 현재는 컴퓨터의 성능이 향상되어 해독이 쉬워졌고, 3번을 반복해서 사용하는 알고리즘 발표
- ④ AES : 미국 표준 기술 연구소(NIST)에서 발표, DES의 한계를 느끼고 발표  
: 블록 크기 128비트, 키 길이는 128,192,256으로 구분

## 9. DOS (Denial Of Service, 서비스 거부)

- 대량의 데이터를 한 곳의 서버에 집중적으로 전송

## 9. Ping Of Death

- 패킷의 크기를 인터넷 프로토콜 허용범위 이상으로 전송 -> 네트워크 마비

## 10. SMURFING (스머핑)

- IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄

## 11. SYN Flooding

- 3-way-handshake 과정을 의도적으로 중단-> 무한대기 상태로 놓여 정상적인 서비스 수행 X
- \*3-way-handshake : 송신지와 수신지 간의 3단계 통신 방법
  - 1 단계 : 송신지에서 수신지로 SYN 전송
  - 2 단계 : 수신지에서 송신지로 SYN+ACK 패킷 전송
  - 3 단계 : 송신지에서 수신지로 ACK 패킷 전송

## 12. Tear Drop

- 패킷 크기가 클경우 이를 분할 전송하는데 이대 분할 순서를 저장하는 Fragment Offset에 Offset을 변경하여 수신측에서 분할된 패킷을 조립하는데 과부하가 걸려 시스템이 다운됨

## 13. LAND Attack

- 패킷을 전송할 때 송신지와 수신지를 모두 공격대상

## 14. DDOS

- 여러 곳에 분산된 공격 지점에서 한곳의 서버에 대해 분산 서비스 공격 수행
- 공격 도구
  - ① Trin00 : 초기형태의 데몬 UDP Flooding 공격 수행
  - ② TFN
  - ③ TFN2K
  - ④ Stacheldraht : 암호화된 통신을 수행

## 15. 네트워크 침해 공격 관련 용어

- 스미싱 : 문자를 이용해 신용정보 유도
- 스피어 피싱 : 위장 메일을 지속적으로 발송, 링크 클릭 유도
- 무작위 대입 공격 : 암호키를 알기위해 무작위 데이터 넣기
- 큐싱 : 큐알코드 해킹
- 스니핑(코를 킁킁거리다, 냄새를 맡다) : 네트워크 중간에 패킷정보 도청

- 크로스 사이팅 스크립팅

#### 16. 정보보안 침해 공격 관련 용어

- 좀비 PC
- C&C 서버 : 해커가 원격지에서 감염된 좀비 PC에서 명령을 내리기 위해 사용되는 서버
- 봇넷 : 악의적으로 사용될 수 있는 컴퓨터들이 네트워크로 연결된 형태
- 웜 : 자기자신을 복제
- 제로데이공격 : 보안 취약점을 공격 : 공격의 신속성을 의미
- 키로거 공격 (Key Logger) : 사용자의 키보드 움직임 탐지
- 랜섬웨어 : 사용자의 컴퓨터에 잡임해 내부 분서를 암호화
- 백도어 : 시스템 설계자가 액세스 편의를 위해 만들어놓는 경로로 공격  
: 무결성검사, 열린포트 검사, 로그분석, SetUID 파일검사
- 트로이목마 : 정상적인 기능을 하는 프로그램으로 위장, 복제능력 없음

#### 17. 방화벽

- 내부의 네트워크와 인터넷 간에 전송되는 정보를 선별하여 수용,거부하는 기능을 가진 차단 시스템

#### 18. 침입 탐지 시스템

- 컴퓨터 시스템의 비정상적인 사용, 오용 등을 실시간으로 탐지하는 시스템
  - ① 오용 탐지 : 미리 입력해둔 공격 패턴 감지
  - ② 이상 탐지 : 비정상적인 행위나 자원 사용 감지

#### 19. 침입 방지 시스템

- 비정상적인 트래픽을 능동적으로 차단하고 격리하는 보안 솔루션
- 방화벽과 침입탐지 시스템을 결합

#### 20. 데이터 유출 방지

- 내부 정보의 외부 유출을 방지하는 보안 솔루션
- 사내 직원이 사용하는 PC와 네트워크 상의 모든 정보를 검색하고 통제해 외부로의 유출을 막는다.

#### 21. 웹 방화벽

- 웹 기반 공격을 방어할 목적으로 만들어진 웹 서버에 특화된 방화벽

#### 22. VPN

- 공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션

#### 23. NAC

- 네트워크에 접속하는 내부 PC의 일관된 보안 관리 기능을 제공하는 보안 솔루션
- 내부 PC의 소프트웨어 사용 현황을 관리하여 불법적인 소프트웨어 설치를 방해한다.

#### 24. ESM

- 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션



## 11 장. 응용 SW 기초 기술 활용

### 1. 운영체제

- 컴퓨터와 시스템 자원들을 효율적으로 관리하여 효과적으로 사용할 수 있도록 한다
- 사용자와 컴퓨터 사이의 인터페이스로 동작하는 소프트웨어이다
- 운영체제의 목적
  - ① 처리능력 : 일정 시간동안 처리하는 일의 양
  - ② 반환시간 : 시스템에 의뢰한 시간부터 처리 완료될때까지 걸린 시간
  - ③ 사용 가능성도 : 즉시 사용 가능한 정도
  - ④ 신뢰도 : 정확하게 해결하는 정도
- 운영체제 기능
  - ① 프로세스, 기억장치, I/O장치 파일 정보 등의 자원을 관리
  - ② 스케줄링 기능 제공
  - ③ 사용자와 시스템 간의 인터페이스 제공

### 2. Window

- 마이크로소프트사가 개발한 운영체제
- 특징
  - ① GUI
  - ② 선점형 멀티테스킹 : CPU가 이용시간 제어, 프로그램을 강제종료하고 자원반환
  - ③ PnP (Plug and Play) : I/O를 연결 했을 때 자동으로 운영체제가 환경을 구성해주는 기능
  - ④ OLE(Object Linking and embedding) : 프로그램에 작성된 문자나 그림등의 객체를 작성 중인 문서에 연결하거나 삽입할 수 있음
  - ⑤ 255자의 긴 파일명
  - ⑥ Single User

### 3. Unix

- MIT , Bell 연구소 등이 공동 개발한 운영체제
- Multi-user, Multi-tasking을 지원한다
- Tree 구조의 파일 시스템
- 구성요소
  - ① 커널 : 운영체제라 생각, CPU 스케줄링 관리 등을 수행
  - ② 셸 : 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기  
ex) C shell, Korn Shell
  - ③ 유틸리티 프로그램: 외부 명령에 해당  
ex) 에디터, 컴파일러, 인터프리터, 디버거

### 4. LINUX

- Unix를 기반으로 개발한 운영체제
- 프로그램 소스 코드가 무료로 공개 -> 다양한 플랫폼에 설치 가능, 재배포 가능

### 5. 기억장치 (반배교)

- 반입전력 : 프로그램이나 데이터를 언제 주기억장치에 적재할 것인지를 결정하는 전략
  - ① 요구반입 : 참조를 요구할 때 적재하는 방법
  - ② 예상반입 : 참조를 예상하여 적재하는 방법

- 배치전략 : 주기억장치의 어디에 위치 시킬 것인지를 결정하는 전략
  - ① 최초적합
  - ② 최적적합
  - ③ 최악적합
- 교체전략 이미 사용되고 있는 영역 중 어느 영역을 교체하여 상용할 것인지를 결정하는 전략
  - ① FIFO :
  - ② OPT (Optimal replacement) : 가장 오랫동안 사용하지 않을 페이지를 교체
  - ③ LRU (Least Recently Used) : 최근에 가장 오랫동안 사용하지 않은 페이지, 스택이나 계수기 사용
  - ④ LFU (Least Frequently Used): 사용빈도가 가장 적은 페이지
  - ⑤ NUR(Not Used Recently) 최근에 사용하지 않은 페이지 교체, **상태비트와 변형비트** 사용
  - ⑥ SCR (Seconde Chance Replacement) : 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되  
는 페이지의 교체를 방지하기 위한 방법

## 6. 가상 기억장치

- 보조기억장치의 일부를 주기억장치처럼 사용하는 것
  - > 용량이 작은 주기억장치를 마치 큰용량을 가진 것 처럼 사용한다
- 프로그램을 여러 개의 작은 블록 단위로 나누어서 보조기억장치에 저장
  - > 프로그램 실행시 요구되는 블록만 주기억장치에 할당하여 처리
- 가상 기억장치의 구현방법

페이징	세그멘테이션
<ul style="list-style-type: none"> <li>- 크기로 나눈 후 주기억장치의 영역에 적재</li> <li>- 내부단편화 발생</li> <li>- 주소변환을 위한 맵테이블 필요</li> <li>- 페이지:프로그램을 일정한 크기로 나눈 단위</li> <li>- 페이지 프레임 : 페이지 크기로 일정하게 나누어진 주기억장치의 단위</li> </ul>	<ul style="list-style-type: none"> <li>- 다양한 크기로 나눈 후 주기억장치의 영역에 적재</li> <li>- 외부단편화 발생</li> <li>- 세그먼트 맵 테이블 필요</li> <li>- 저장공간 절약</li> <li>- 각 세그먼트는 고유한 이름과 크기를 가짐</li> </ul>

- 페이징 크기에 따른 장단점

페이지가 작을 경우	페이지가 클 경우
<ul style="list-style-type: none"> <li>- 단편화 감소</li> <li>- 주기억장치로 옮기는 시간이 줄어 듦</li> <li>- 불필요한 내용이 적재될 확률이 적음               <ul style="list-style-type: none"> <li>-&gt; 효율적인 워킹 셋 구현 가능</li> </ul> </li> <li>- 맵테이블 크기 커짐, 매핑 속도 느려짐</li> <li>- 디스크 접근 횟수 증가 -&gt;입출력 시간 늘어남</li> </ul>	<ul style="list-style-type: none"> <li>- 맵테이블 크기가 작아짐 -&gt; 매핑속도 증가</li> <li>- 디스크 접근 횟수 작아짐               <ul style="list-style-type: none"> <li>-&gt; 입출력 효율성증가</li> </ul> </li> <li>- 단편화 증가</li> <li>- 주기억장치로 이동하는 시간 늘어남</li> </ul>

## 7. Locality (구역성, 지역성)

- 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다
- 시간구역성: 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상
- 공간구역성: 하나의 페이지를 집중적으로 액세스하는 현상

## 8. 워킹셋

- 프로세스가 일정시간동안 자주 참조하는 페이지의 집합
- 자주 참조되는 워킹셋을 주기억장치에 상주시킴으로서 페이지 부재 및 교체 현상을 줄여들게 한다

## 9. 스래싱

- 프로세스 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU 사용률은 특정 시점까지는 높아지지만 어느 시점부터는 스래싱이 나타나 CPU 이용률은 급격하게 감소한다.



## 10. 프로세스

- 실행중인 프로그램
- PCB 를 가진 프로그램
- 실기억장치에 저장된 프로그램
- 프로시저가 활동중인 것
- 비동기적 행위를 일으키는 주체
- 운영체제가 관리하는 실행단위

## 11. 스래드

- 시스템의 여러 자원을 할당받아 실행하는 프로그램단위
- 프로세스의 일부 특성을 갖고 있기 때문에 경량 프로세스라고도 불림

## 12. PCB (Process Control Block)

- 운영체제가 프로세스에 대한 정보를 저장해 놓는곳
- PCB 에 저장되어 있는 정보
  - ① 프로세스의 현재상태
  - ② 포인터 (PC)
  - ③ 프로세스 고유 식별자
  - ④ 스케줄링 및 프로세스의 우선순위
  - ⑤ CPU 레지스터정보
  - ⑥ 주기억장치 관리 정보
  - ⑦ 입출력 상태 정보
  - ⑧ 계정정보

## 13. 프로세스 상태 전이



\* Dispatch : 준비상태의 프로세스가 프로세서를 할당받아 실행 상태로 전이

\* wake up : I/O 입력을 기다리던 프로세스가 입력을 받아 다시 Ready 상태로 전이

\* Spooling : 속도가 느린 입출력장치에 데이터를 보내지 않고 한꺼번에 입출력 하기 위해서 디스크에 저장하는 과정

\* Traffic Controller : 프로세스의 상태에 대한 조사와 통보 담방

#### 14. 스케줄링

- 시스템의 여러자원을 해당 프로세스에게 할당하는 작업
- 장기 스케줄링 : 어떤 프로세스를 결정하여 준비상태 큐로 보내는 작업
- 중기 스케줄링 : 어떤 프로세스가 CPU 를 할당받을지 결정하는 작업
- 단기 스케줄링 : 프로세스가 실행되기 위해 CPU 를 할당받는 시기와 특정 프로세스를 지정하는 작업

#### 15. 스케줄링의 목적

- 공정성, 처리율 증가, CPU 이용률 증가,
- 우선순위 제도 : 우선순위가 높은 프로세스를 먼저 실행
- 오버헤드 최소화, 응답시간 최소화, 반환시간최소화, 대기시간 최소화
- 균형 있는 자원의 사용 : 메모리, 입출력장치 등의 자원을 균형있게 사용함
- 무한 연기 회피

#### 16. 비선점 스케줄링 : 이미 할당된 CPU 를 다른 프로세스가 강제로 빼앗을 수 없음

- FCFS
- SJF
- HRN : 우선순위 계산식 사용 = 
$$\frac{(\text{대기시간} + \text{서비스시간})}{\text{서비스시간}}$$

#### 17. 선점 스케줄링 : CPU 가 강제로 빼앗아 사용할 수 있는 스케줄링

- 라운드로빈, SRT, 선점우선순위, 다단계 큐, 다단계 피드백

#### 18. 인터넷

- 전세계 수많은 컴퓨터와 네트워크가 연결된 광범위한 컴퓨터 통신망

#### 19. IP 주소

- 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소
- A class : 국가나 대형 통신망에 사용, 32bit- 8bit = 24bit , 2^24 개의 호스트 사용 가능(0~127) : 128
- B class : 중대형 통신망에 사용 24bit-8bit= 16bit, 즉 2^16 개의 호스트 사용 가능 (128~191) : 64
- C class : 소규모 통신망에서 사용 16bit-8bit = 8bit , 즉 2^8 개의 호스트 사용가능 (192~ 223) : 32
- D class : 멀티태스킹용 (224~239) 16
- E class : 실험적 주소이며 공용되지 않음

#### 20. 서브네팅

- 할당된 네트워크 주소를 여러 개의 작은 네트워크로 나누어 사용하는 것
- Broadcast : IP 주소 범위에서 가장 마지막 주소를 의미
- ex) 192.168.1.0/24 네트워크를 FLSM 방식을 이용하여 3 개의 Subnet 으로 나누시오

#### 21. IPv6

IPv4	IPv6
32 비트 IP 주소체계	128 비트 IP 주소 체계
8 비트씩 4 부분	16 비트씩 8 부분
패킷 크기 64KB 제한	패킷 제한 없음
ABC 클래스 단위의 비순차적 할당	순차적 할당
품질제어 지원수단 없음	품질 보장 용이
TCP 헤더크기 20~60byte	40byte 고정

- 유니캐스트 : 1:1 통신
- 멀티캐스트 : 1:N 통신
- 애니캐스트 : 1:1 통신 , 근거리

## 22. 도메인 네임

- 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현
- DNS(Domain Name System) : 컴퓨터가 이해할 수 있는 IP 주소로 변환하는 역할

## 23. OSI 7 계층 ( 물데네 전세표응)

: ISO 에서 제아한 통신 규약

상위계층	7. 응용계층	- 사용자 데이터 생성 - E-mail, Http, FTP, 원격접속
	6. 표현계층	- 코드 변화, 구문검색, 데이터 압축, 코드번역
	5. 세션계층	- 데이터 교환 관리 - 동기점을 사용해 데이터 회복을 함
하위계층	4. 전송계층	- 투명한 데이터 전송 제공 - TCP/UDP
	3. 네트워크계층	- 흐름제어, 트래픽 제, 데이터 교환 및 중계 - 라우터, X.25
	2. 데이터링크 계층	- 오류제어(검출,회복), 흐름제어 - 신뢰성있고 효율적인 프레임 데이터 전송 - 체크섬필드가 오류검출을 위해 헤더에 존재 - 브리지, 스위치, LAN
	1. 물리계층	- 기계적, 전기적, 기능적, 절차적 기능 - 통신 케이블, 전송 신호 방식, 물리적 장비 - 허브, 리피터

\* 허브 : 가까운 거리의 컴퓨터를 연결하는 장치로 회선을 통합하여 관리

: 신호를 증폭 기능을 하는 리피터의 역할을 포함한다

\* 리피터 : 수신한 신호를 재생시키거나 출력 전압을 높여 전송하는 장치

\* 브리지 : LAN 과 LAN 을 연결, 트래픽 병목형상을 줄인다.

\* 스위치 : LAN 과 LAN 을 연결하여 더 큰 LAN 을 생성

\* 라우터 : 데이터 전송의 최적 경로를 선택하는 장치

\* 게이트웨이 : 프로토콜 구조가 다른 네트워크를 연결하는 장치

## 24. 프로토콜

- 데이터 교환을 원활하게 수행할 수 있도록 표준화 시켜놓은 통신 규약
- 기본요소(시구의)
  - ① 시간 : 두 기기 간의 통신 속도, 메시지 순서 제어 등을 규정
  - ② 구문 : 전송하고자 하는 데이터 형식, 부호화, 신호 레벨 등을 규정
  - ③ 의미 : 두 기기 간의 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보 규정

## 25. TCP/IP 프로토콜

OSI	TCP/IP	프로토콜	
7. 응용계층	응용계층	FTP, HTTP, Telnet, POP/SMTP	NFS, TFTP
6. 표현계층			
5. 세션계층			
4. 전송계층	전송 계층	TCP	UDP
3. 네트워크계층	인터넷계층	데이터 전송을 위한 주소지정, 경로 설정을 제공 IP, ICMP, IGMP, ARP, RARP	
2. 데이터링크 계층	네트워크 액세스 계층	실제 데이터를 송 수신하는 역할 이더넷, IEEE802, HDLC, ARQ	
1. 물리계층			

- TCP
  - 전송계층에 해당
  - 패킷단위로 데이터 교환
  - 에러 유무 검사 -> 잘못 전송된 패킷에 대해 재전송 요구 -> 투명성이 보장되는 통신 제공
  - 스ক্র임 전송 기능을 제공한다
  - TCP 통신 수립을 3 단계 핸드셰이킹이라고 한다 -> 이걸 해킹하는게 SYN Flooding
    - 1 단계 : 수신측에 플래그 비트 SYN 을 1 로 설정하여 전송
    - 2 단계 : 수신측은 받았으면 ACK 를 1 로 송신측에 전송
    - 송신측은 ACK 를 다시 1 로 하여 전송
- UDP
  - 신뢰성을 보장하지 않는 비접속형 통신
  - 송신순서와 수신순서가 다를 수 있음
  - ACK 가 없음
  - 대표 서비스 : SNMP, DNS, TFTP, NFS,
- 전자메일
  - POP/SMTP (수신/송신)서버 사용
  - MIME : 멀티미디어 전자우편을 주고받기 위한 표준
  - IMAP : POP 보다 뛰어남, 양방향 통신 제공
- HTTP
- FTP :
- Telnet : 원격접속, 가상 터미널 기능 존재
- 유즈넷 : 공지사항 전송
- ARP : 물리적인 주소로 번역해주는 프로토콜
- RARP : IP 주소로 번역해주는 프로토콜
- IP : 전송할 데이터에 주소를 지정, 경로를 설정, 비연결형인 데이터 그램->신뢰성 X
- ICMP: ping 에서 사용, IP 패킷을 처리할 때 발생하는 문제를 알려주는 프로토콜 (Echo, Ping, Tracert)
- DNS: 도메인, 공백사용불가, 한글과 숫자를 섞어서 생성 가능
- 이더넷 : CSMA/CD 방식의 LAN
- HDLC : 비트위주의 데이터 링크 제어 프로토콜
- X.25 : 패킷 교환망을 통한 DTE 와 호스트 간의 인터페이스를 제공하는 프로토콜

## 26. TCP/IP 전달 구조

Data	TCP 헤더	IP 헤더	MAC 주소
------	--------	-------	--------

- Data 구조 : 16bit, 24bit, 32bit, ASCII, EBCDIC, Binary 등으로 구분
- TCP 헤더: 송수신자의 포트번호, 순서번호, 응답번호 등 전달되는 정보를 파악
  - ACK : 확인 응답 메시지
  - PSH: 데이터를 상위 계층에 즉시 전달하도록 지시할 때 사용
  - RST: 유효하지 않은 패킷에 대한 응답용
  - SYN: 동기화를 할 때 사용
- IP 헤더:
  - IP 버전과 송신측/수신측 IP, 프로토콜의 종류, 서비스 정타입 정보 기억
- MAC: 데이터 링크 계층에서 통신을 위한 인터페이스에 할당된 고유 식별자

## 27. 네트워크 구축

- 네트워크 : 두대 이상의 컴퓨터를 연결하여 자원을 공유하는것
- 성형 : Point-to- point, 중앙 컴퓨터가 있고 단말장치들이 연결되는 중앙 집중식 네트워크
- 링형(루프형): 이웃하는 것끼리 연결시킨 형태, (point to point
- 버스형 : 한개의 통신회선에 여러 대의 단말장치 연결,  
기밀 보장이 어려움, 통신회선의 길이에 제한 존재
- 계층형 : 트리형으로 연결시키는 형태, 분산 처리 시스템을 구성하는 방식
- 망형 : 모든 지점의 컴퓨터와 단말장치를 서로 연결하는 형태, 노드당 n-1 개의 간선 필요,

## 28. 네트워크 분류

- 근거리 통신망 : 버스형, 링형 사용
- 광대역 통신망 : 근거리 통신망을 확장시킨 버전 , 속도 느림

## 29. NAT

- 한개의 정식 IP 주소에 대량의 가상 사설 IP 주소를 할당 및 연결하는 기능

## 30. 경로제어 프로토콜

IGP (Interior Gateway protocol)	-하나의 자율 시스템내의 라우팅에 사용되는 프로토콜 -RIP: 벡터라우팅 프로토콜 최대 홉수를 15로 제한 -> 대규모 네트워크에서 사용 불가 - OSPF (open shortest path first) : RIP 단점 보완 다익스트라 알고리즘 사용 라우팅 정보에 변화 발생시 해당 정보만 네트워크 내의 모든 라우터에 알림
EGP (Exterior --)	-자율시스템 간의 라우팅, 즉 게이트웨이 간의 라우팅에 사용되는 프로토콜
BGP(Border --)	-EGP의 단점을 보완하기 위해 만들어짐 -전체경로 제어표(라우팅 테이블) 교환, 전송

## 31. 트래픽 제어

- 전송되는 패킷의 흐름 또는 양을 조절하는 기능
- 종류
  - ① 흐름제어 : 송수신 측 사이에 전송되는 패킷의 양이나 속도를 규제
    - 정지-대기: 수신확인신호(ACK)를 받은 후 다음 패킷전송, 하나의 패킷만 전송 가능
    - 슬라이딩 윈도우 : 수신 통지를 이용하여 송신 데이터 양을 조절

② 폭주제어 : 네트워크 내의 패킷 수를 조절하는 것

- 느린 시작
- 혼잡회피 : 임계값에 도달하게 되면 회피

③ 교착상태 방지

32. 회복 : 데이터베이스가 손상되기 전 상태로 복구하는 작업

- 연기갱신기법 : 트랜잭션이 완료 될때까지 갱신을 연기, Redo 작업만 가능
- 즉각갱신기법 : 즉시 실제 데이터베이스에 반영, 갱신된 내용들은 LOG 보관, Redo, Undo 가능
- 그림자페이지대체 : 일정 크기의 페이지 단위로 구성하여 페이지마다 복사본관리
- 검사점 기법 : 특정단계에서 재실행 할 수있도록

33. 병행 제어

- 로킹 : 트랜잭션들이 액세스 하기전에 LOCK 을 요청해서 로킹단위를 액세스 가능
- 타임스탬프 : 갱신한 데이터에 대해 트랜잭션이 실행을 시작하기 전에 시간표를 부여  
부여된 시간표에 맞춰 트랜잭션 수행
- 최적 병행 :
- 다중 버전 : 타임 스탬프 개념 용

34. 로킹 단위

- 로킹단위가 크면 병행수준 낮아짐
- 로킹 단위가 작으면 오버헤드가 증가하지만 병행성 수준 높아짐

35. 교착상태

- 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상
- 교착상태 발생의 필요충분 조건
  - ① 상호배제 : 한번에 한 개의 프로세스만이 공유 자원을 사용
  - ② 점유와 대기 : 자원을 점유하고 있으면서 다른 자원을 점유하기 위해서는 대기 프로세스 존재 필요
  - ③ 비선점 : 강제로 빼앗을수 없음
  - ④ 환형 대기
- 해결 방법
  - ① 예방방법 : 위의 네가지 기법중 하나를 제거, 자원의 낭비가 가장 심함
  - ② 회피기법 :
    - 은행원 알고리즘 : 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는 데서 유래한 기법
  - ③ 발견 기법
  - ④ 회복기법