# 정보처리기사 실기 전체정리

| | | |
|---|---|---|
| 소프트웨어 설계 | 요구사항 확인<br>화면 설계 | |
| 소프트웨어 개발 | 데이터 입출력 구현<br>통합 구현<br>제품소프트웨어 패키징<br>애플리케이션 관리<br>인터페이스 구현 | 운영체제, DB, 네트워크, 보안, 소프트웨어 개발방식<br><br>최근 기출 풀이 => 파트 분석 |
| 데이터베이스 구축 | SQL응용 | 코딩 => 다 맞아야 함, C언어 JAVA 파이썬 SQL |
| 프로그래밍 언어 활용 | 서버프로그램 구현<br>프로그래밍 언어 활용<br>응용SW 기초기술 활용 | 약술형 키워드 정리 |
| 정보시스템 구축관리 | 소프트웨어 개발 보안 구축 | |

---

## [[ 요구사항 확인 ]]

---

※ 소프트웨어 개발 방법론★★★

소프트웨어 생명주기 SDLC: 시스템의 요구분석부터 유지보수까지,
전 공정을 체계화한 절차

소프트웨어 생명주기 모델 종류:

- 폭포수 모델: 가장 오래됨, 각 단계 마무리 => 다음 단계
- 프로토타이핑 모델: 주요 기능을 프로토타입으로 구현, 고객의 피드백 반영해 S/W 제작
- 나선형 모델: 위험을 최소화, 점진적 시스템 개발
- 반복적 모델: 구축대상을 나누어 병렬적으로 개발 후 통합, 또는 반복적 개발

소프트웨어 개발 방법론: 소프트웨어 개발의 시작부터 시스템을 사용 않는 과정까지,
전 과정을 형상화한 방법론

소프트웨어 개발 방법론 종류:

- 구조적 방법론: 전체 시스템을 기능에 따라 나누어 개발, 이를 통합하는 방법론
  (나씨-슈나이더만 차트: 논리의 기술에 중점을 둔 도형식 표현방법)
- 정보공학 방법론: 정보시스템 개발에 필요한 관리절차와 작업 기법을 체계화한 방법론
- 객체지향 방법론: '객체'라는 기본 단위로 시스템을 분석 및 설계하는 방법론
- 컴포넌트 기반 방법론(CBD): 컴포넌트를 조립해서 하나의 새로운 응용 프로그램을 작성하는 방법론
- 애자일 방법론: 절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하면서
  효율적인 시스템을 개발할 수 있는 신속 적응적 개량 개발 방법론
- 제품 계열 방법론: 특정 제품에 적용하고 싶은 공통된 기능을 정의해 개발하는 방법론,
  임베디드 S/W작성에 유용함.

애자일 Agile 방법론 유형

- XP(eXtreme Programming): 의사소통 개선과 즉각적 피드백으로 소프트웨어 품질을 높이기 위한 방법론
  - XP 5가지 가치: 용기, 단순성, 의사소통, 피드백, 존중
- 스크럼(Scrum): 매일 정해진 시간, 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심 방법론
- 린(Lean): 도요타의 린 시스템 품질기법을 소프트웨어 개발 프로세스에 적용해서
  낭비 요소를 제거하여 품질을 향상시킨 방법론
  - Lean 7가지 가치: 낭비제거, 품질 내재화, 지식 창출, 늦은 확정,
    빠른 인도, 사람 존중, 전체 최적화

객체 지향 분석(OOA): 사용자의 요구사항을 분석하여 요구된 문제와 관련된
모든 클래스(객체), 속성과 연산, 관계를 정의

객체지향 분석 방법론 종류:

- OOSE(Object Oriented Software Engineering): 유스케이스를 모든 모델의 근간으로 활용되는 방법론 / 야콥슨
- OMT(Object Modeling Technology): 그래픽 표기법을 이용하여 소프트웨어 구성요소를 모델링 / 럼바우
  - 분석 절차: 객체 모델링 -> 동적 모델링 -> 기능 모델링
  - 객체 모델링: 객체들 간의 관계를 정의하여 ER 다이어그램을 만드는 과정까지의 모델링
    객체 다이어그램 활용
  - 동적 모델링: 시간의 흐름에 따라 객체들의 동적인 행위를 표현하는 모델링
    상태 다이어그램 사용
  - 기능 모델링: 프로세스들의 자료 흐름을 중심으로 처리 과정을 표현하는 모델링
    자료 흐름도(DFD) 활용

비용 산정 모형 분류

하향식 산정방법: 경험이 많은 전문가에게 비용 산정 의뢰 또는 전문가와 조정자를 통해 비용 산정 / 전문가 판단

델파이 기법: 전문가의 경험적 지식을 통한 문제 해결 및 미래예측을 위한 기법

상향식 산정방법: 세부적인 요구사항과 기능에 따라 필요한 비용 산정

코드 라인 수(LoC, Lines of Code): 원시 코드 라인 수의 낙관치, 중간치, 비관치를 측정하여 예측치를 구해 비용산정

Man Month: 한 사람이 1개월 동안 할 수 있는 양을 기준으로 비용 산정

COCOMO모형: 보헴이 제안한 모형, 프로그램의 규모에 따라 비용 산정

조직형(Organic Mode): 5만(50KDSI)라인 이하

반 분리형(Semi-Detached Mode): 30만(300KDSI)라인 이하

임베디드형(Embedded Mode): 30만(300KDSI)라인 이상

푸트남(Putnam) 모형: 개발주기의 단계별로 요구할 인력의 분포를 가정하는 방식

기능점수(FP) 모형: 소프트웨어 기능을 중대시키는 요인별로 가중치를 부여하여 비용산정

비용 산정 자동화 추정 도구

SLIM: Rayleigh-Norden곡선과 Putnam예측 모델을 기초로 하여 개발된 자동화 추정 도구

ESTIMACS: 다양한 프로젝트와 개인별 요소를 수용하도록 FP모형을 기초로 하여 개발된 자동화 추정 도구

일정관리 모델: 프로젝트가 일정 기한 내에 완료될 수 있도록 관리하는 모델

주 공정법(CPM): 여러 작업의 수행 순서가 얽혀 있는 프로젝트의 일정을 계산하는 기법

주 공정(Critical Path, 임계 경로): 프로젝트의 시작에서 종료까지 가장 긴 시간이 걸리는 경로

PERT: 일의 순서를 계획적으로 정리하기 위한 수렴기법. 비관치, 중간치, 낙관치 이용

중요 연쇄 프로젝트 관리(CCPM): 주 공정 연쇄법으로 자원제약사항을 고려하여 일정을 작성하는 기법

※ 현행 시스템 분석 ★★★

현행 시스템 파악: 현행시스템의 어떤 기술 요소 사용을 하는지 파악하는 활동

현행 시스템 파악 절차: 구성/기능/인터페이스 파악 -> 아키텍처 및 소프트웨어 구성 파악 -> 하드웨어 및 네트워크 구성 파악

소프트웨어 아키텍처: 여러 가지 소프트웨어 구성요소와 그 구성요소가 가진 특성 중 외부에 드러나는 특성,
그리고 구성요소 간의 관계를 표현하는 시스템의 구조나 구조체

소프트웨어 아키텍처 4+1뷰: 고객의 요구사항을 정리해놓은 시나리오를 4개의 관점에서 바라보는 소프트웨어적인 접근 방법

유스케이스 뷰: 유스케이스 또는 아키텍처를 도출하고 설계하며 다른 뷰를 검증하는데 사용되는 뷰

논리 뷰: 시스템의 기능적인 요구사항이 어떻게 제공되는지 설명해주는 뷰

프로세스 뷰: 시스템의 비기능적인 속성으로 자원의 효율적인 사용, 병행 실행, 비동기, 이벤트 처리 등을 표현한 뷰

규현 뷰: 개발 환경 안에서 정적인 소프트웨어 모듈의 구성을 보여주는 뷰,
컴포넌트 구조와 의존성을 부여주고, 부가적인 정보를 정의

배포 뷰: 컴포넌트가 물리적인 아키텍처에 어떻게 배치되는가를 매핑해서 보여주는 뷰

소프트웨어 아키텍처 패턴 유형

계층화 패턴(Layered Patten): 시스템을 계층으로 구분하여 구성하는 패턴

클라이언트-서버 패턴(Client-Server Pattern): 하나의 서버와 다수의 클라이언트로 구성된 패턴

파이프-필터 패턴(Pipe-Filter Pattern): 데이터 스트림을 생성하고 처리하는 시스템에서 사용 가능한 패턴,
재사용성이 좋고 추가가 쉬워 확장에 용이

브로커 패턴(Broker Pattern): 분리된 컴포넌트들로 이루어진 분산 시스템에서 사용,
각 컴포넌트 원격 서비스 실행을 통해 상호작용이 가능

모델-뷰-컨트롤러 패턴(MVC, Model-View-Controller Pattern): 대형 애플리케이션을 3개의 서브 시스템으로 구조화한 패턴,
컴포넌트로 분리되어 있어 서로 영향을 받지 않고 개발 작업 수행 가능

모델(Model): 핵심 기능과 데이터 보관

뷰(View): 사용자에게 정보 표시

컨트롤러(Controller): 사용자로부터 요청을 입력받아 처리

소프트웨어 아키텍처 비용 평가 모델 종류

SAAM: 변경 용이성과 기능성에 집중, 경험이 없는 조직에서도 활용 가능한 비용평가 모델

ATAM: 아키텍처 품질 속성을 만족시키는지 판단 및 품질 속성들의 이해 상충관계까지 평가하는 모델

CBAM: ATAM 바탕의 시스템으로, 경제적 의사결정에 대한 요구를 충족하여 평가 모델

ADR: 소프트웨어 아키텍처 구성요소 간 응집도 평가 모델

ARID: 전체 아키텍처가 아닌 특정 부분에 대한 품질요소에 집중하여 비용 평가 모델

디자인 패턴: 소프트웨어 설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계 방법을 정리한 패턴

디자인 패턴 유형

목적:

생성: 객체 인스턴스 생성에 관여, 클래스 정의와 객체 생성방식을 구조화, 캡슐화를 수행하는 패턴

구조: 클래스나 객체의 조합을 다루는 패턴

행위: 클래스나 객체들이 상호 작용하는 방법과 역할 분담을 다루는 패턴

범위:

클래스: 상속 관계를 다루는 패턴, 컴파일 타임에 정적으로 결정

객체: 객체 간 관련성을 다루는 패턴, 런타임에 동적으로 결정

디자인 패턴 종류

생성패턴: Builder, Prototype, Factory Method, Abstract Factory, Singleton

구조패턴: Bridge, Decorator, Facade, Flyweight, Proxy, Composite, Adapter

행위패턴: Mediator, Interpreter, Iterator, Template Method, Observer, State, Visitor, Command, Strategy, Memento, Chain of Responsibility

운영체제 Operating System: 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스를 담당하는 프로그램

운영체제 종류: PC / 윈도즈, 유닉스, 리눅스

모바일 / 안드로이드, iOS

OSI 계층: 네트워크 통신에서 충돌 문제를 완화하기 위해 국제 표준화 기구(ISO)에서 제시한 모델

응용 계층(Application Layer): 사용자와 네트워크 간 응용서비스 연결, 데이터 연결

표현 계층(Presentation Layer): 데이터 형식 설정과 부호 교환, 암/복호화

세션 계층(Session Layer): 연결 접속 및 동기 제어

전송 계층(Transport Layer): 신뢰성 있는 통신 보장, 데이터 분할과 재조립, 흐름 제어, 혼잡 제어 등 담당

네트워크 계층(Network Layer): 단말 간 데이터 전송을 위한 최적화된 경로 제공

데이터 링크 계층(Data Link Layer): 인접 시스템 간 데이터 전송, 전송오류 제어

물리 계층(Physical Layer): 0과 1의 비트 정보를 회선에 보내기 위한 전기적 신호 변환

DBMS(Database Management System): 데이터의 집합을 만들고, 저장 및 관리할 수 있는 기능들을 제공하는 응용 프로그램

미들웨어(Middleware): 분산 컴퓨팅 환경에서 응용 프로그램과 프로그램이 운영되는 환경 간에 원만한 통신이 이루어질 수 있도록 제어해주는 소프트웨어. 대표적인 미들웨어: WAS

웹 애플리케이션 서버(Web Application Server):

서버 계층에서 애플리케이션이 동작할 수 있는 환경을 제공하고

안정적인 트랜젝션 처리와 관리, 다른 이기종 시스템과의 애플리케이션 연동을 지원하는 서버

※ 요구사항 확인 ★★★

요구공학(Requirements Engineering): 사용자의 요구가 반영된 시스템을 개발하기 위해 사용자 요구사항에 대한 도출, 분석, 명세, 확인 및 검증하는 구조화된 활동

요구사항의 분류

기능적 요구사항: 시스템이 제공하는 기능, 서비스에 대한 요구사항

특정 입력/상황에 대해 시스템이 어떻게 반응/동작 해야하는지에 대한 기술

특성: 기능성, 완전성, 일관성

비기능적 요구사항: 시스템 구축에 대한 제약사항에 대한 요구사항

품질 속성에 관련하여 시스템이 갖춰야할 사항에 관한 기술, 시스템이 준수해야 할 제한 조건에 관한 기술

특성: 신뢰성, 사용성, 효율성, 유지보수성, 이식성, 보안성, 및 품질 관련 요구사항, 제약사항

요구공학 프로세스: 도출 -> 분석 -> 명세 -> 확인 및 검증

요구사항 도출 단계 주요 기법: 인터뷰, 브레인스토밍, 델파이기법, 롤 플레잉, 워크숍, 설문조사

델파이기법: 전문가의 경험적 지식을 통한 문제 해결 및 미래예측을 위한 방법

요구사항 확인 및 검증 단계의 주요 기법

요구사항 검토: 여러 검토자들이 에러, 잘못된 가정, 불명확성, 표준과의 차이 검토

정형기술 검토 활용

동료 검토: 2~3명 리뷰 진행, 요구사항 명세서를 설명하고 이해관계자들이 들으면서 결함을 반견하는 형태로 진행

프로토타이핑 활용: 프로토타입(견본품)을 통해 효과적으로 요구 분석을 수행하면서 명세서를 산출하는 작업

모델 검증: 분석 단계에서 개발된 모델의 품질 검증 필요

테스트 케이스 및 테스트를 통한 확인: 각각의 요구사항을 어떻게 확인할 것인지에 대한 계획을 수립하고 테스트 케이스 작성

CASE 도구 활용 검증: 자동화된 일관성 분석을 제공하는 CASE도구 활용

베이스라인을 통한 검증:

요구사항 변경을 체계적으로 추적하고 통제하는 시점인 베이스라인을 통한 요구사항에 대한 지속적 검증 수행

요구사항 추적표(RTM)를 통한 검증:

요구사항 정의서를 기준으로 개발 단계별 최종 산출물이 어떻게 반영되고, 변경되었는지 확인이 가능한 문서

# [[ 화면 설계 ]]

※ UI 요구사항 확인 ★★★

UI, User Interface: 사용자와 시스템 사이에서 의사소통 할 수 있도록 고안된 물리적, 가상의 매개체

UI 유형

- CLI, Command Line Interface: 명령어를 텍스트로 입력하여 조작하는 사용자 인터페이스
- GUI, Graphical User Interface: 그래픽 환경을 기반으로 한 마우스나 전자펜을 이용한 사용자 인터페이스
- NUI, Natural User Interface: 신체 부위를 이용하는 사용자 인터페이스
- OUI, Organic User Interface: 현실에 존재하는 모든 사물이 입출력장치로 변화할 수 있는 사용자 인터페이스

UI 설계 원칙

- 직관성: 누구나 쉽게 이해하고, 쉽게 사용할 수 있어야 함
- 유효성: 정확하고 완벽하게 사용자의 목표가 달성될 수 있도록 제작
- 학습성: 초보와 숙련자 모두가 쉽게 배우고 사용할 수 있게 제작
- 유연성: 사용자의 요구사항을 최대한 수용하고, 실수를 방지할 수 있도록 제작

UI 설계 지침: 사용자 중심, 일관성, 단순성, 결과 예측 가능, 가시성, 표준화, 접근성, 명확성, 오류 발생 해결

UI 품질 요구사항(ISO/IEC 9126 기반)

- 기능성: 실제 수행 결과와 품질 요구사항과의 차이를 분석, 시스템 동작을 관찰하기 위한 품질 기준
  (적절성, 정밀성, 상호 운용성, 보안성, 호환성)
- 신뢰성: 시스템이 일정한 시간 또는 작동되는 시간 동안 의도하는 기능을 수행함을 보증하는 품질 기준
  (성숙성, 고장 허용성, 회복성)
- 사용성: 사용자와 컴퓨터 사이에 발생하는 어떠한 행위를 정확하고 쉽게 인지할 수 있는 품질 기준
  (이해성, 학습성, 운용성)
- 효율성: 할당된 시간에 한정된 자원으로 얼마나 빨리 처리할 수 있는가에 대한 품질 기준
  (시간 효율성, 자원 효율성)
- 유지보수성: 요구사항을 개선하고 확장하는 데 있어 얼마나 용이한거에 대한 품질 기준
  (분석성, 변경성, 안정성, 시험성)
- 이식성: 다른 플랫폼에서도 추가 작업 없이 얼마나 쉽게 적용 가능한가에 대한 품질 기준
  (적용성, 설치성, 대체성)

UI 표준: 대자인 철학과 원칙 기반하에 전체 시스템에 공통으로 적용되는 화면 간 이동, 화면구성 등에 관한 규약

CRUD(Create, Read, Update, Delete): 컴퓨터 소프트웨어가 가지는 기본적인 데이터 처리 기능

UI 개발을 위한 주요 기법

- 3C 분석: 고객(Customer), 자사(Company), 경쟁사(Competitor)를 비교하고 분석하여,
  자사를 어떻게 차별화해서 경쟁에서 이길 것인가를 분석하는 기법
- SWOT 분석: 기업의 내/외부 환경을 분석하여 Strength(강점) Weakness(약점) Opportunity(기회) Treat(위협) 요인을 규정
  이를 토대로 경영 전략을 수립하는 방법
- 시나리오 플래닝: 상황 변화를 사전에 예측하고 다양한 시나리오를 설계하여 불확실성을 제하는 경영 전략 방법
- 사용성 테스트: 사용자가 직접 제품을 사용하면서 시나리오에 맞춰 과제를 수행한 후 질문에 응답하는 테스트
- 워크숍: 특정 문제나 과제에 대한 새로운 지식, 기술, 아이디어, 방법들을 서로 교환하고 검토하는 세미나

UI 화면 설계 구분

- 와이어프레임: 화면 단위의 레이아웃을 설계하는 작업
  (ppt, 키노트, 스케치, 일러스트)
- 스토리보드: 서비스 구축을 위한 모든 정보(정책, 프로세스, 와이어프레임, 기능 정의 등)가 담겨 있는 설계 산출물
  (ppt, 키노트, 스케치)
- 프로토타입: 정적인 화면(와이어프레임, 스토리보드)에 동적 효과를 적용하여 실제 구현된 것처럼 시뮬레이션 할 수 있는 모형
  전체적인 기능을 간략한 형태로 구현한 시제품(HTML, CSS)

※ UI 설계 ★★★

UML(Unified Modeling Language): 객체 지향 소프트웨어 개발 과정에서 산출물을 명세화, 시각화, 문서화할 때 사용되는 모델링 기술과 방법론을 통합해서 만든 표준화된 범용 모델링 언어

UML 특징: 가시화 언어, 구축 언어, 명세화 언어, 문서화 언어

UML 구성요소: 사물, 관계, 다이어그램

UML 다이어그램

- 구조적 다이어그램 / 정적 다이어그램
  - 클래스, Class: 클래스의 속성 및 연산과 클래스간 정적인 관계를 표현
  - 객체, Object: 클래스에 속한 사물(객체=인스턴스)를 특정 시점의 객체와 객체 사이의 관계로 표현
  - 컴포넌트, Component: 시스템을 구성하는 물리적인 컴포넌트와 그들 사이의 의존 관계 표현
  - 배치, Deployment: 컴포넌트 사이의 종속성을 표현하고, 물리적인 요소들의 위치 표현
  - 복합체 구조, Composite Structure: 클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현
  - 패키지, Package: 유스케이스, 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계
- 행위적 다이어그램 / 동적 다이어그램
  - 유스케이스, Usecase: 시스템이 제공하고 있는 기능 및 그와 관련된 외부 요소
  - 시퀀스, Sequence: 객체 간 동적 상호 작용을 시간적 개념을 중심으로 메시지 흐름을 표현
  - 커뮤니케이션, Communication: 동작에 참여하는 객체들이 주고받는 메시지를 표현하고, 객체 간의 연관까지 표현
  - 상태, State: 자신이 속한 클래스의 상태 변화 or 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는 지 표현
  - 활동, Activity: 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서대로 표현
  - 타이밍, Timing: 객체 상태 변화와 시간 제약을 명시적으로 표현

UI 시나리오 문서의 작성요건: 완전성, 일관성, 이해성, 가독성, 추적 용이성, 수정 용이성

UI 설계 도구의 유형

- 화면 설계 도구: 파워 목업, 발사믹 목업, (카카오)오븐
- 프로토타이핑 도구: UX핀, 액슈어, 네이버 프로토나우
- UI 디자인 도구: 스케치, Adobe XD
- UI 디자인 산출물로 작업하는 프로토타이핑 도구: 인버전, 픽사에이트, 프레이머

---

## [[ 데이터 입출력 구현 ]]

---

※ 논리 데이터 저장소 확인 ★★★

데이터 모델 Data Model: 현실 세계의 정보를 인간과 컴퓨터가 이해할 수 있도록 추상화하여 표현한 모델

데이터 모델 절차: 요구사항 분석 -> 개념적 설계 -> 논리적 설계 -> 물리적 설계

논리 데이터 모델링: 업무의 모습을 모델링 표기법으로 형상화하여 사람이 이해하기 쉽게 표현하는 프로세스

논리적 데이터 모델링 종류

- 관계 데이터 모델: 테이블 형태, 1:1, 1:N, N:M
- 계층 데이터 모델: 트리 형태(상하 관계), 1:N
- 네트워크 데이터 모델: 그래프 형태: N:M

관계 대수: 관계형 데이터베이스에서 원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 절차적 정형 언어

관계 대수 연산자 종류

- 일반 집한 연산자
  - 합집합(Union): ∪ / 교집합(Intersection): ∩ / 차집합(Difference): — / 카티션 프로덕트(CARTESIAN Product): X
- 순수 관계 연산자
  - 셀렉트(Select): σ / 프로젝트(Project): π / 조인(Join): ⋈ / 디비전(Division): ÷

관계 해석: 튜플 관계해석과 도메인 해석을 하는 비절차적 언어

논리 데이터 모델링 속성: 개체(Entity), 속성(Attribute), 관계(Relationship)

개체-관계(E-R) 모델: 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 표현한 모델

정규화 Normalization:

데이터 모델에서 중복성을 제거, 이상 현상을 방지하고, 데이터의 일관성과 정확성을 유지하기 위해 무손실 분해하는 과정

이상현상(Anomaly): 데이터의 중복성으로 인해 릴레이션을 조작할 때 발생하는 비합리적 현상

- 삽입이상: 불필요한 세부정보를 입력하는 경우
- 삭제이상: 원치 않는 다른 정보가 같이 삭제되는 경우
- 갱신이상: 특정부분만 수정되어 중복된 값이 모순을 일으키는 경우

정규화 단계 Normal Form

1정규형(1NF): 도메인이 원자값으로 구성

2정규형(2NF): 부분함수 종속제거(완전 함수적 종속을 만족)

3정규형(3NF): 이행함수 종속제거

보이스 코드 정규형(BCNF, boyce codd normal form): 결정자 후보키가 아닌 함수 종속 제거

4정규형(4NF): 다중 값 종속제거

5정규형(5NF): 조인 종속 제거

반정규화 De-Nomalization: 정규화된 개체, 속성, 관계에 대해 성능향상과 개발운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링 기법

※ 물리 데이터 저장소 설계 ★★★

참조무결성 제약조건: 릴레이션과 릴레이션 사이에 대한 참조의 일관성을 보장하기 위한 조건

제한 Restricted: 다른 테이블이 삭제할 테이블을 참조 중이면 제거하기 않는 옵션

연쇄 Cascade: 참조하는 테이블까지 연쇄적으로 제거하는 옵션

널 값 Set Null: 참조되는 릴레이션에서 튜플을 삭제하고, 참조하는 튜플들의 외래값에 NULL값을 넣는 옵션
만약 NOT NULL 명시 시 삭제 연산이 거절됨

```
ALTER TABLE 테이블 ADD
FOREIGN KEY (외래키)
REFERENCES 참조테이블 (기본키)
ON DELETE [ RESTRICT | CASCADE | SET NULL ]
```

인덱스 Index: 데이터 레코드를 빠르게 접근하기 위해 '키 값, 포인터' 쌍으로 구성되는 데이터 구조

클러스터드 인덱스 Clustered Index: 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식

넌클러스터드 인덱스 Non-Clustered Index: 인덱스의 키 값만 저장되어 있고, 실제 데이터는 정렬되지 않는 방식

뷰 View: 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로 구성된 가상 테이블

클러스터 Cluster: 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법.
클러스터의 분포도가 넓을수록 유리하다.

파티션 Partition: 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나눈 것

범위 분할 Range Partitioning: 지정한 열의 값을 기준으로 분할함

해시 분할 Hash Partitioning: 해시 함수를 적용한 결과 값에 따라 데이터를 분할

리스트 분할 List Partitioning: 특정 파티션에 저장될 데이터에 대한 명시적 제어가 가능한 분할 기법

조합 분할 Composite Partitioning: 범위, 해시, 리스트 분할 중 2개 이상의 파티셔닝을 결합하는 방식.
파티션이 너무 클 때 사용

파티션의 장점: 성능 향상, 가용성 향상, 백업 가능, 경합 감소

※ 물리 데이터 저장소 설계 ★★★

데이터베이스 Database: 다수의 인원, 시스템 또는 프로그램이 사용할 목적으로 통합하여 관리되는 데이터의 집합

데이터베이스 정의

통합된 데이터: 자료의 중복을 배제한 데이터의 모임

저장된 데이터: 저장 매체에 저장된 데이터

운영 데이터: 조직의 업무를 수행하는 데 필요한 데이터

공용 데이터: 여러 애플리케이션, 시스템들이 공동으로 사용하는 데이터

데이터베이스 특성

실시간 접근성: 쿼리에 대하여 실시간 응답이 가능해야 함

계속적인 변화: 새로운 데이터의 삽입, 삭제 갱신으로 항상 최신의 데이터를 유지

동시공용: 다수의 사용자가 동시에 같은 내용의 데이터를 이용할 수 있어야함

내용참조: 사용자가 요구하는 데이터 내용으로 데이터를 찾음

데이터베이스 종류

파일 시스템 File System: 파일에 이름을 부여하고 저장이나 검색을 위해 논리적으로 그것들을 어디에 위치시켜야 하는지 등을 정의한 뒤 관리하는 데이터베이스 전 단계의 데이터 관리 방식

관계형 데이터베이스 시스템 RDBMS: 관계형 모델을 기반

종류: Oracle, SQL Server, MySQL, Mari DB

계층형 데이터베이스 시스템 HDBMS: 데이터를 상하 종속적인 관계로 계층화하여 관리

종류: IMS, System2000

네트워크 데이터베이스 관리 시스템 NDBMS: 데이터를 네트워크상의 망상 형태로 표현한 데이터 모델

종류: IDS, IDMS

DBMS Database Management System: 데이터 관리의 복잡성을 해결하는 동시에
데이터 추가, 변경, 검색, 삭제 및 백업, 복구, 보안 등의 기능을 지원하는 소프트웨어

DBMS 유형
- 키-값 Key-Vaule DBMS: Unique: 한 키에 하나의 값
- 컬럼 기반 데이터 저장 Column Family Data Store DBMS:
  Key 안에 (Column, Value) 조합으로 된 여러 개의 필드를 갖는 DBMS
- 문서 저장 Document Store: 값(Value)의 데이터 타입이 문서(Document)라는 타입을 사용하는 DBMS
- 그래프 Graph DBMS: 시멘틱 웹과 온톨로지 분야에서 활용되는 그래프로 데이터를 표현하는 DBMS

DBMS 특징: 무결성, 일관성, 회복성, 보안성, 효율성

빅 데이터 Big Data: 시스템, 서비스, 조직(회사) 등에서 주어진 비용, 시간 내에 처리 가능한 수십 페타바이트 PB 크기의 비정형 데이터
- 빅 데이터 특성 3V: Volume 데이터의 양, Velocity 데이터의 속도, Variety 데이터의 다양성
- 빅 데이터 수집, 저장, 처리기술
  - 비정형/반정형 데이터 수집: 내/외부 정제되지 않은 데이터를 확보하여 수집 및 전송하는 기술
  - 정형 데이터 수집: 내/외부 정제된 대용량 데이터의 수집 및 전송 기술
  - 분산 데이터 저장/처리: 대용량 파일의 효과적인 분산 저장 및 분산 처리 기술
  - 분산 데이터 베이스: HDFS 칼럼 기반 데이터베이스로 실시간 랜덤 조회 및 업데이트 가능
  - HDFS Hadoop Distributed File System:
    대용량 데이터의 집합을 처리하는 응용 프로그램에 적합하도록 설계된 하둡 분산 파일 시스템

NoSQL Not Only SQL: 데이터 저장에 고정된 테이블 스키마가 필요치 않고 조인 연산을 사용할 수 없으며, 수평적 확장이 가능한 DBMS
- NoSQL 특성
  - Basically Available: 언제든지 데이터에 접근할 수 있는 속성 / 가용적
  - Soft-State: 외부에서 전송된 정보를 통해 결정되는 속성
  - Eventually Consistent: 이관성이 유지되는 속성 / 지속적
- NoSQL 유형
  - Key-Value Store: Unique한 키에 하나의 값을 가지고 있는 형태
  - Column Family Data Store: Key안에 (Column, Value) 조합으로 된 여러개의 필드를 갖는 DB
  - Document Store: 값(Value)의 데이터 타입이 문서(Document)라는 타입을 사용하는 DB
  - Graph DBMS: 시멘틱 웹과 온톨로지 분야에서 활용되는 그래프로 데이터를 표현하는 DBMS
    - 시멘틱 웹: 온톨로지를 활용하여 서비스를 기술하고, 온톨로지의 의미적 상호 운용성을 이용하여 서비스 검색, 조합, 중재 기능을 자동화하는 웹
    - 온톨로지: 실세계에 존재하는 모든 개념과 개념들의 속성, 그리고 개념들 간 관계 정보를 컴퓨터가 이해할 수 있도록 서술해 놓은 지식 베이스

데이터 마이닝 Data Minning: 대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아내는 기술
데이터 마이닝 절차: 목적 설정 -> 데이터 준비 -> 가공 -> 마이닝 기법 적용 -> 정보 검증

데이터 마이닝 주요 기법
- 분류 규칙: 과거 데이터로를 토대로 새로운 레코드의 결과 값을 예측하는 기법
- 연과 규칙: 데이터 안에 항목들 간의 종속관계를 찾아내는 기법
- 연속 규칙: 연관 규칙에 시간 관련 정보가 포함된 형태의 기법
- 데이터 군집화: 대상 레코드들을 유사한 특성을 지닌 몇 개의 소그룹으로 분할하는 작업

---

## [[ 통합 구현 ]]

---

※ 연계 요구사항 분석 ★
연계 요구사항 분석: 서로 다른 두 시스템-장치-소프트웨어를 잇는 중계 역할을 하는 연계 시스템과 관련된 요구사항을 분석하는 과정
연계 요구사항 분석 참고문서
- 개체 Entity 정의서: 데이터베이스 개념 모델렝 단계에서 도출한 개체의 타입과 관련 속성, 식별자 등의 정보를 개괄적으로 명세화한 정의서
- 테이블 Table 정의서: 논리 및 물리 모델링 과정 설계 산출물
- 인터페이스 명세서: 인터페이스 정의서에 작성한 항목을 자세히 작성한 것

※ 연계 매커니즘 구성 ★★

연계 매커니즘: 응용 소프트웨어와 연계 대상 모듈 간의 데이터 연계 시 요구사항을 고려한 연계방법과 주기를 설계하기 위한 매커니즘

기능: 데이터를 생성하여 전송하는 송신 시스템과 송신 데이터를 수신하여 DB에 반영하는 수신 시스템으로 구성

주요 연계 기술

직접 연계

DB 링크: 데이터베이스에서 제공하는 DB링크 객체를 이용

DB 연결: 수신 시스템의 WAS에서 송신 시스템 DB로 연결하는 DB커넥션 풀을 생성하고 연계 프로그램에서 해당 DB커넥션 풀 명을 이용하여 연결

API/Open API: 송신 시스템의 DB에서 데이터를 읽어서 제공하는 애플리케이션 프로그래밍 인터페이스 시스템

JDBC: 수신 시스템의 프로그램에서 JDBC 드라이버를 이용하여 송신 시스템 DB와 연결

하이퍼링크: 현재 페이지에서 다른 부분으로 가거나 전혀 다른 페이지로 이동하게 해주는 속성

간접 연계

연계 솔루션(EAI, Enterprise Application Integration): 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해주는 솔루션

Web Service/ESB: 웹 서비스가 설명된 WSDL과 SOAP 프로토콜을 이용한 시스템 간 연계

소켓 Socket: 소켓을 생성하여 포트를 할당하고, 클라이언트의 요청을 연결하여 통신

※ 내외부 연계 모듈 구현 ★★★

EAI(Enterprise Application Integration): 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간의 정보를 전달, 연계 통합이 가능하도록 해주는 솔루션
// 미들웨어를 이용하여 비즈니스 로직을 중심으로 기업 내 애플리케이션을 통합 연계

EAI 구성요소

EAI 플랫폼: 이기종 시스템 간 애플리케이션 상호 운영

어댑터: 다양한 애플리케이션을 연결하는 EAI의 핵심 장치로 데이터 입출력 도구

브로커: 데이터 포맷과 코드를 변환하는 솔루션

메시지 큐: 비동기 메시지를 사용하는 다른 응용 프로그램 사이에서 데이터를 송수신하는 기술

비즈니스 워크플로우: 미리 정의된 기업의 비즈니스 Workflow에 따라 업무를 처리하는 기능

EAI 구축 유형

포인트 투 포인트 Point-to-point: 가장 기초적인 애플리케이션 통합방법, 1:1 단순 통합 방법

허브 앤 스포크 Hub & Spoke: 단일한 접점의 허브 시스템을 통하여 데이터를 전송하는 중앙 집중식 방식

메시지 버스 Message Bus: 애플리케이션 사이 미들웨어를 두어 연계하는 미들웨어 통합 방식

하이브리드 Hybrid: 그룹 내=허브 앤 스포크 방식, 그룹 간=메시지 버스 방식. (통합 방식)

ESB(Enterprise Service Bus): 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간을 하나의 시스템으로 관리 운영할 수 있도록 서비스 중심의 통합을 지향하는 아키텍처. 느슨한 결합 방식 지원
(서비스 변경이 있어도 연결된 다른 서비스에 영향이 없음)
// 미들웨어를 이용하여 서비스 중심으로 서비스를 지원하기 위한 관련 시스템과 유기적 연계

ESB 구축 유형: 버스 방식의 분산형 토폴로지 구성

웹 서비스 Web Service: 네트워크에 분산된 정보를 서비스 형태로 개방하여 표준화된 방식으로 공유하는 기술, 서비스 지향 아키텍처

웹 서비스 유형

SOAP(Simple Object Access Protocol): HTTP, HTTPS, SMTP 사용해 XML기반 메시지를 네트워크 상태에서 교환하는 프로토콜

WSDL(Web Service Description Language): 웹 서비스명, 제공 위치, 메시지 포맷, 프로토콜 정보 등 웹서비스에 대한 상세 정보가 기술된 XML 형식으로 구현되어 있는 언어

UDDI(Universal Description, Discovery and Integration): 웹 서비스에 대한 정보인 WSDL를 등록하고, 검색하기 위한 저장소로 공개적으로 접근, 검색이 가능한 레지스트리

IPC(Inter-Process Communication): 운영체제에서 프로세스 간 서로 데이터를 주고받기 위한 통신 기술

연계 테스트: 송신 시스템과 수신 시스템을 연계하였을 경우 데이터의 정합성과, 데이터 전송 여부에 대한 테스트

---

# [[ 인터페이스 구현 ]]

---

※ 인터페이스 기능 구현 ★★★

컴포넌트 명세서: 컴포넌트 개요, 부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세

인터페이스 명세서: 컴포넌트 명세서에 명시된 인터페이스 클래스의 세부적인 조건 및 기능을 명시한 명세서

JSON(JavaScript Object Notation): 속성-값 쌍 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해
인간이 읽을 수 있는 테스트를 사용하는 개방형 표준 포맷

XML(eXtensible Markup Language): HTML의 단점을 보완한 인터넷 언어로, SGML의 복잡한 단점개선, 특수한 목적을 갖는 마크업 언어

AJAX(Asynchronous JavaScript And XML): 비동기 자바스크립트와 XML
자바스크립트로 웹 서버와 클라이언트 간 비동기적으로 XML데이터를 교환, 조작하는 웹 기술
XMLHttpRequest(비동기 통신 담당 JS객체)를 이용해서 필요한 일부 페이지의 데이터만 로드

REST(Representational State Transfer): 웹과 같은 분산 하이퍼미디어 환경에서 자원의 존재/상태 정보를
표준화된 HTTP메소드로 주고받은 웹 아키텍처

REST 메서드: HTTP메서드 중 CRUD메서드만 사용 ~ POST 생성 / GET 조회 / PUT 수정 / DELETE 삭제

데이터베이스 암호화 알고리즘

- 대칭 키 암호화 알고리즘: 암/복호화에 같은 암호 키를 쓰는 알고리즘
- 비대칭 키 암호화 알고리즘: 공개키는 누구나 알 수 있지만, 비밀키는 키 소유자만 알 수 있게 사용하는 알고리즘
- 해시 암호화 알고리즘: 해시값으로 원래 입력값을 찾아낼 수 없는 일방향성의 특성을 가진 알고리즘

중요 인터페이스 데이터의 암호화 전송 보안 기술

- IPSec(IP Security): IP계층(3계층)에서 무결성과 인증을 보장하는 인증 헤더(AH)와 기밀성을 보장하는
암호화(ESP)를 이용하여 양 종단 간(End Point) 구간에 보안 서비스를 제공하는 터널링 프로토콜
- SSL(Secure Sockets Layer) / TLS(Transport Layer Security):
전송계층(4)과 응용계층(7) 사이에서 클라이언트와 서버 간의 웹 데이터 암호화, 기밀성
상호 인증 및 전송 시 데이터 무결성을 보장하는 보안 프로토콜
- S-HTTP(Secure Hypertext Transfer Protocol): 웹상에서 네트워크 트래픽을 암호화하는 주요 방법,
클라이언트와 서버 간에 전송되는 모든 메시지를 암호화하여 전송

※ 인터페이스 구현 검증 ★★

인터페이스 구현 검증 도구

- xUnit: 자바, C++, .Net 등 다양한 언어를 지원하는 단위테스트 프레임워크
- STAF(Software Testing Automation Framework): 서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
- FitNesse: 웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
- NTAF: FitNesse(협업가능)+STAF(재사용, 확장성) = 통합 테스트 자동화 프레임워크
- Selenium: 다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
- watir: 루비(Ruby)기반 웹 애플리케이션 테스트 프레임워크

인터페이스 감시 도구

- 스카우터 SCOUTER: 애플리케이션에 대한 모니터링 및 DB Agent를 통해 오픈 소스 DB 모니터링 기능, 인터페이스 감시 기능
- 제니퍼 Jennifer: 애플리케이션 개발부터 테스트, 오픈, 운영, 안정화까지 전 생애주기 단계동안 성능을 모니터링, 분석 API

---

# [[ SQL 응용 ]]

---

※ 데이터베이스 기본 ★★★

트랜잭션 Transaction: 인가받지 않은 사용자로부터 데이터를 보장하기 위해 DBMS가 가져야 하는 특성,
하나의 논리적 기능을 정상적으로 수행하기 위한 작업의 기본 단위

트랜잭션 특성

- 원자성 Atomicity: 트랜잭션의 연산 전체가 성공 또는 실패되어야 하는 성질(All or Nothing)
- 일관성 Consistency: 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 하는 성질
- 격리성 Isolation: 동시에 실행되는 트랜잭션들이 서로 영향을 미치지 않아야 한다는 성질
- 영속성 Durability: 성공이 완료된 트랜잭션의 결과는 영속적으로 데이터베이스에 저장되어야 하는 성질

트랜잭션 제어어 TCL, Transaction Control Language: 트랜잭션의 결과를 허용하거나 취소하는 목적으로 사용되는 언어

TCL 명령어

COMMIT: 트랜잭션을 메모리에 영구적으로 저장하는 명령어

ROLLBACK: 트랜잭션 내역의 저장을 무효화시키는 명령어

CHECKPOINT(SAVEPOINT): ROLLBACK을 위한 시점을 지정하는 명령어

데이터 정의어 DDL, Data Definition Language: DB를 구축하거나 수정할 목적을 사용하는 언어

DDL 대상

도메인 Domain: 하나의 속성이 가질 수 있는 원자값들의 집합

스키마 Schema: 데이터베이스의 구조, 제약조건 등의 정보를 담고 있는 기본적인 구조
외부 스키마, 개념 스키마, 내부 스키마

테이블 Table: 데이터 저장 공간

뷰 View: 하나 이상의 물리 테이블에서 유도되는 가상의 테이블

인덱스 Index: 검색을 빠르게 하기 위한 데이터 구조

순서 인덱스 Ordered Index: 데이터가 정렬된 순서로 생성되는 인덱스

해시 인덱스 Hash Index: 해시 함수에 의해 직접 데이터에 키 값으로 접근하는 인덱스

비트맵 인덱스 Bitmap Index: bit 값인 0 또는 1로 변환하여 인덱스 키로 사용하는 인덱스

함수기반 인덱스 Functional Index: 수식이나 함수를 적용하여 만든 인덱스

단일 인덱스 Single Index: 하나의 컬럼으로만 구성한 인덱스

결합 인덱스 Concatenated Index: 두 개 이상의 컬럼으로 구성한 인덱스

클러스터드 인덱스 Clustered Index: 인덱스 키의 순서에 따라 데이터 정렬되어 저장하는 방식(검색이 빠름)

넌클러스터드 인덱스 Non-Clustered Index: 인덱스의 키 값만 정렬되어 있고, 실제 데이터는 정렬되지 않는 방식
(데이터 삽입, 삭제 시 데이터 재정렬해야 함)

DDL 명령어: CREATE(생성), ALTER(수정), DROP(삭제)

CASCADE 제거할 요소를 참조하는 모든 개체를 제거 / RESTRICT 다른 개체가 제거할 요소를 참조 중일 때 제거를 취소

데이터 조작어 DML, Data Manipulation Language: 저장된 데이터를 실질적으로 관리하는 데 사용하는 언어

DML 유형: SELECT(조회), INSERT(삽입), UPDATE(수정), DELETE(삭제)

데이터 제어어 DCL, Data Control Language: 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는데 사용하는 언어

DCL 유형: GRANT(사용권한 부여), REVOKE(사용 권한 취소)

※ 응용 SQL 작성하기 ★

데이터 분석 함수 종류

집계 함수: 여러 행 또는 테이블 전체 행으로부터 하나 결과값을 반환하는 함수

그룹 함수: 소그룹 간의 소계 및 중계 등의 중간 합계 분석 데이터를 산출하는 함수

윈도우 함수: 행과 행 간의 관계를 쉽게 정리하기 위해 만든 함수

WINDOW_FUNCTION () OVER > OVER가 필수 키워드로 포함됨

※ 절차형 SQL 활용하기 ★

절차형 SQL, Procedural SQL: SQL언어에서도 절차 지향적인 프로그램이 가능하도록 하는 트랜잭션 언어

절차형 SQL 종류

프로시저 Procedure: 일련의 쿼리들을 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합

사용자 정의 함수 User-Defined Function: SQL 처리를 수행, 수행 결과를 단일 값으로 반환할 수 있는 절차형 SQL

트리거 Trigger: 데이터베이스 시스템에서 삽입, 갱신, 삭제 등의 이벤트가 발생할 때마다
관련 작업이 자동으로 수행되는 절차형 SQL

※ 데이터 조작 프로시저 최적화 ★

쿼리 성능 개선: 최소의 시간으로 원하는 결과를 얻도록 프로시저를 수정하는 작업

SQL 성능 개선 절차: 문제있는 SQL 식별 -> 옵티마이저 통계 확인 -> SQL문 재구성 -> 인덱스 재구성 -> 실행계획 유지관리

옵티마이저 Optimizer: SQL이 가장 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈 / SQL의 실행계획을 만들어주는 역할

규칙 기반 옵티마이저 RBO, Rule Based Optimizer:

사전에 정의해둔 규칙에 의거하여 경로를 찾는 규칙 기반 옵티마이저

비용 기반 옵티마이저 CBO, Cost Based Optimizer:

각 DBMS마다 고유의 알고리즘에 따라 산출되는 비용으로 최적의 경로를 찾는 비용 기반 옵티마이저

힌트 Hint: 실행하려는 SQL문에 사전에 정보를 주어서 SQL문 실행에 빠른 결과를 가져오는 효과를 만드는 문법

옵티마이저의 실행 계획을 원하는 대로 변경, 옵티마이저는 명시적인 힌트를 통해 실행 계획을 변경

# [[ 서버 프로그램 구현 ]]

※ 개발환경 구축 ★★★

개발 도구 분류

빌드 도구: 작성한 코드의 빌드 및 배포를 수행하는 도구(Ant, Maven, Gradle)
구현 도구: 코드의 작성과 디버깅, 수정 등과 같이 작업 시 사용되는 도구(Eclipse, InteliJ, VS)
테스트 도구: 코드의 기능 검증과 전체의 품질을 높이기 위해 사용하는 도구(xUnit, PMD, Sonar)
형상 관리 도구: 산출물에 대한 버전 관리를 위한 도구(Git, SVN, CVS)

서버 하드웨어 개발환경

웹 서버: HTTP를 이용한 요청/응답을 처리(Apache 웹서버, 구글 웹서버)
웹 애플리케이션 서버(AWS): 동적 콘텐츠를 처리하고 제공하기 위해 사용(Tomcat, Jeus, Resin)
데이터베이스 서버: 데이터의 수집, 저장을 위한 용도로 사용(MySql, Oracle, MS-SQL)
파일 서버: 파일 저장 하드웨어로 물리 저장 장치를 활용한 서버(HDD, SSD)

소프트웨어 개발환경

운영체제: 사용자 관점에서 편리하고 유용하게 사용하기 위한 소프트웨어
미들웨어: 웹 서버, JVM과 같은 개발 환경의 구성요소로 활용
DBMS: 데이터의 저장 및 활용을 위해 DBMS를 설치

형상 관리 Configuration Management: 소프트웨어 개발을 위한 전체 과정에서 발생하는 모든 항목의 변경 사항을 관리하기 위한 활동
형상 관리 절차: 형상 식별(대상정의) -> 형상 통제(버전관리) -> 형상 감사(무결성) -> 형상 기록(보고서)

소프트웨어 형상 관리 도구 유형

공유 폴더 방식: 매일 개발이 완료된 파일은 약속된 위치의 공유 폴더에 복사하는 방식(RCS, SCCS) .sccs
클라이언트/서버 방식: 중앙에 버전 관리 시스템을 항시 동작시키는 방식(CVS, SVN)
분산 저장소 방식: 로컬 저장소와 원격 저장소로 분리되어 분산 저장하는 방식(Git)
RCS, Revision Control System: 소스 파일을 관리하는 여러 명령으로 구성되어, 한 파일의 변경 사항을 하나의 파일로 관리, 소스파일의 변경 사항을 추적
SCCS, Source Code Control System: 이전 버전의 오리지널 소스 코드, 저장된 변경사항을 검색할 수 있음
Create, Edit, Delget, Get, Prt

소프트웨어 형상 관리 도구별 특징

CVS(Concurrent Versions System):
서버와 클라이언트로 구성되어 있고, 다수의 인원이 동시에 범용적인 운영체제로 접근 가능한 형상 관리 도구
SVN(Subversion): 하나의 서버에서 소스를 쉽고 유용하게 관리할 수 있게 도와주는 도구
RCS(Revision Control System): 소스 파일 수정을 한 사람만으로 제한, 다수가 동시에 수정 할 수 없게 잠금 방식 형상 관리
Bitkeeper: 중앙 통제 방식으로 대규모 프로젝트에서 빠른 속도를 내도록 개발한 형상 관리 도구
Git: 속도에 중점을 둔 분산형 버전 관리 시스템, 대형 프로젝트에서 효과적이고 유용함
Clear Case: 복수 서버, 복수 클라이언트 구조, 필요한 서버를 하나씩 추가하여 확장성을 기할 수 있음

※ 공통 모듈 구현 ★★

모듈 Module: 하나의 완전한 기능을 수행할 수 있는 독립된 실체
모듈화 Modularity: 프로그램 개발 시 생산성과 최적과, 관리에 용이하게 기능 단위로 분할하는 기법
모듈화 원리: 정보은닉, 분할과 정복, 데이터 추상화, 모듈 독립성

응집도 Cohesion: 모듈의 독립성을 나타내는 정도로, 모듈 내부 구성요소 간 연관 정도
우연적 응집도 Coincidental Cohesion < 논리적 응집도 Logical Cohesion < 시간적 응집도 Temporal Cohesion
< 절차적 응집도 Procedural Cohesion < 통신적 응집도 Communication Cohesion < 순차적 응집도 Sequential Cohesion
< 기능적 응집도 Functional Cohesion

결합도 Coupling: 외부 모듈과의 연관도 또는 모듈 간의 상호의존성. 모듈간의 관련성을 측정하는 척도
내용 결합도 Content Coupling > 공통 결합도 Common Coupling > 외부 결합도 External Coupling
> 제어 결합도 Control Coupling > 스탬프 결합도 Stamp Coupling > 자료 결합도 Data Coupling

응집도는 높을수록 좋고, 결합도는 낮을수록 좋다.

공통 모듈 테스트: IDE 도구를 활용하여 개별 공통 모듈에 대한 디버깅을 수행

통합 개발 환경 IDE, Integrated Development Environment: 개발에 필요한 다양한 틀을 하나의 인터페이스로 통합하여 제공 (Eclipse, VS, Android, Studio, IDEA)

Junit: 자바 프로그래밍 언어용 단위테스트 도구

※ 배치 프로그램 구현 ★

배치 프로그램, Batch Program: 사용자와의 상호 작용 없이 일련의 작업들을 작업 단위로 묶어, 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법

배치 프로그램 유형

이벤트 배치: 사전에 정의해 둔 조건 충족 시 자동으로 실행

온디맨드 배치: 사용자의 명시적 요구가 있을 때마다 실행

정기 배치: 정해진 기간에 정기적으로 실행(시간, 요일 등)

배치 스케줄러 Batch Scheduler: 일괄 처리(Bacth Processing) 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원하는 도구

배치 스케줄러 종류

스프링 배치 Spring Batch: 오픈 소스 프레임워크

쿼츠 스케줄러 Quartz Scheduler: 수행할 작업과 수행 시간을 관리하는 요소들을 분리하여 일괄 처리 작업에 유연성을 제공(오픈 소스 프레임워크)

Corn 표현식: 크론 표현식을 통해 배치 수행 시간을 시간 및 주기 등으로 설정

리눅스/유닉스 크론 표현식: 분, 시간, 일, 월, 요일, 연도

쿼츠 크론 표현식: 초, 분, 시간, 일, 월, 요일, 연도

---

## [[ 소프트웨어 개발 보안 구축 ]]

---

※ 소프트웨어 개발 보안 설계 ★★★

SW 개발 보안: 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동

SW 개발 보안 생명주기: 요구사항 명세 -> 설계 -> 구현 -> 테스트 -> 유지보수

SW 개발 보안 3대 요소

| | |
|---|---|
| 기밀성 Confidentiality: | 시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용 |
| 무결성 Integrity: | 시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있음 |
| 가용성 Availavility: | 인가받은 사용자는 시스템 내의 정보와 자원을 언제라도 사용할 수 있음 |

DoS, Denial of Service 공격: 시스템을 공격해서 해당 시스템의 자원을 부족하게 해, 의도된 용도로 사용하지 못하게 하는 공격

DoS 공격의 종류

SYN 플러딩, SYN Flooding: 서버의 동시 가용 사용자수를 SYN 패킷만 보내 점유, 사용자가 서버를 사용 불가능하게 하는 공격

UDP 플러딩, UDP Flooding: 대량의 UDP 패킷 생성, 임의의 포트 번호로 전송, 응답 메시지 생성=>지속적으로 자원을 고갈

스머프 Smurf/스머핑 Smurfing: 출발지 주소를 대상의 IP로 설정, 네트워크 전체에 ICMP Echo패킷을 브로드캐스팅해 마비시킴

죽음의 핑 PoD, Ping of Death: ICMP 패킷을 비정상적으로 크게 전송해, 정상적인 서비스를 못하도록 공격

랜드 어택 Land Attack: 출발지 IP와 목적지 IP를 같은 패킷 주소로 만들어 보내, 시스템의 가용성을 침해하는 공격

티어 드롭 Tear Drop: IP패킷의 재조합 과정에서 잘못된 정보로 인해, 수신 시스템이 문제를 발생하도록 만드는 공격

봉크 Bonk/보잉크 Boink: 프로토콜의 오류 제어를 이용한 공격기법

DDos, Distributed Denial of Service: 여러 대의 공격자를 분산 배치하여 동시에 동작하게 함으로서 특정 사이트를 공격하는 기법

DDos 공격도구

Trinoo: 많은 소스로부터 통합된 UDP flood 서비스 거부 공격을 유발하는 데 사용하는 도구

Tribe Flood Network: 많은 소스에서 하나 혹은 여러 개의 목표 시스템에 대해 서비스 공격을 수행할 수 있는 도구

Stacheldraht: 분산 서비스 거부 에이전트 역할을 하는 Linux 및 Solaris 시스템용 멀웨어 도구

DoS와 DDoS 차이

DoS: 직접 공격, DDoS: 공격하도록 제시

DoS는 한 사람에 의해 공격을 감행, DDoS는 수많은 감염 호스트를 통해 공격을 감행함

DRDoS, Distributed Reflection DoS: 공격자는 출발지 IP를 공격대상 IP로 위조하여, 다수의 반사 서버로 요청 정보를 전송, 공격 대상자는 반사 서버로부터 다량의 응답을 받아서 서비스 거부(DoS)되는 공격

세션 하이재킹 Session Hijacking: TCP의 세션 관리 취약점을 이용한 공격 기법, 케빈 미트닉 사용

애플리케이션 공격기법

HTTP GET 플러딩: 과도한 GET메시지를 이용해 웹 서버의 과부하를 유발시키는 공격
Slowloris: HTTP GET 메서드를 사용하여, 헤더의 최종 끝을 알리는 개행 문자열 전송하지 않고,
대상 웹 서버와 연결 상태를 장시간 지속시키고, 연결자원을 모두 소진시키는 서비스 거부 공격
Hulk DoS: 공격자가 웹 페이지 주소를 지속적으로 변경하면서, 다량으로 GET요청을 발생시키는 서비스 거부 공격
Hash DoS: 다량의 파라미터를 POST로 웹서버로 전달, 다수의 해시 충돌을 발생시켜 자원을 소모시키는 서비스 거부 공격

네트워크 공격

스니핑: 공격대상의 데이터만 몰래 들여다보는 수동적 공격 기법
네트워크 스캐너, 스니퍼: 네트워크 하드웨어 및 소프트웨어 구성의 취약점 파악을 위해
공격자가 취약점을 탐색하는 공격 도구
패스워드 크래킹
사전 크래킹: ID와 PW가 될 가능성이 있는 단어를 파일로 만들어 파일의 단어를 대입하여 크랙하는 공격 기법
무차별 크래킹: 무작위로 패스워드 자리에 대입하여 패스워드를 알아내는 공격 기법
패스워드 하이브리드 공격: 사전 공격+무차별 공격
레인보우 테이블 공격: 크래킹 하고자 하는 해시 값을 테이블에서 검색해서 역으로 패스워드를 찾는 공격 기법
IP 스푸핑: 침입자가 인증된 컴퓨팅 시스템인 것처럼 속여서 인증된 호스트의 IP 주소로 위조하여 타깃에 전송하는 공격 기법
ARP 스푸핑: 공격자가 특정 호스트의 MAC 주소를 자신의 MAC 주소로 위조한 ARP Reply를 만들어 희생자에게 지속적 전송
ICMP Redirect 공격: 스니핑 시스템을 네트워크에 존재하는 또 다른 라우터라고 알림으로서 패킷의 흐름을 바꾸는 공격 기법
트로이 목마: 악성 루틴이 숨어있는 프로그램, 실행하면 악성 코드를 실행

버퍼 오버플로우 Buffer Overflow 공격: 메모리에 할당된 버퍼 크기를 초과하는 양의 데이터를 입력하여 프로세스의 흐름을 변경시켜
악성 코드를 실행시키는 공격 기법
백도어 Backdoor: 허가받지 않고 시스템에 접속하는 권리, 정상적인 인증 절차를 우회하는 기법

보안 관련 용어

스피어 피싱 Spear-Phishing: 발송 메일의 본문 링크나 첨부된 파일을 클릭하도록 유도, 사용자의 개인정보를 탈취 공격기법
스미싱 Smishing: SMS를 이용하여 개인 비밀정보를 요구하거나 휴대폰 소액 결제를 유도하는 피싱 공격
큐싱 Qshing: QR코드를 통해 악성 앱을 내려받도록 유도하여 금융 정보 등을 빼내는 피싱 공격
봇넷 Botnet: 악성 프로그램이 감염되어 있는 컴퓨터들이 네트워크로 연결된 형태
APT, Advanced Persistent Threat 공격: 다양한 수단을 통한 지속적이고 지능적인 맞춤형 공격 기법
공급망 공격 Supply Chain Attack: SW개발사의 네트워크에 침투하여 악의적 코드를 삽입, 서버 배포하여
사용자가 설치 또는 업데이트 시에 자동적으로 감염되도록 하는 공격 기법
제로데이 공격 Zero Day Attack: 보안 취약점이 발견되어 널리 공표되기 전에 해당 취약점을 악용하는 보안 공격 기법
웜 Worm: 스스로 복제하여 네트워크 등의 연결을 통하여 전파하는 악성 소프트웨어 프로그램
악성 봇 Malicious Bot: 해커의 명령에 의해 원격에서 제어 또는 실행이 가능한 프로그램 또는 코드
사이버 킬체인 Cyber Kill Chain: 공격형 방위 시스템, APT공격방어 분석 모델
랜섬웨어 Ransomeware: 시스템의 파일을 암호화하여 인질처럼 잡고 몸값을 요구하는 악성 소프트웨어
이블 트윈 Evil Twin 공격: 핫스팟에 연결한 무선 사용자들의 정보를 탈취하는 무선 네트워크 공격 기법

서버 인증의 기능: 스니핑 방지(SSL인증서 설치), 피싱 방지, 데이터 변조 방지, 기업 신뢰도 향상(기업 인증)

인증 기술의 유형

지식기반 인증: 사용자가 기억하고 있는 지식(ID/PW)
소지기반 인증: 소지하고 있는 사용자 물품(공인인증서, OTP)
생체기반 인증: 고유한 사용자의 생체 정보(홍채, 얼굴, 지문)
특정기반 인증: 사용자의 특징을 활용(서명, 몸짓)

접근 통제 기법

식별 Identification: 자신이 누구라고 시스템에 밝히는 행위
인증 Authentication: 주체의 신원을 검증하기 위한 활동
인가 Authorization: 인증된 주체에게 접근을 허용하는 활동
책임추적성 Accountability: 주체의 접근을 추적하고 행동을 기록하는 활동

서버 접근 통제 유형

임의적 접근 통제 DAC, Discretionary Access Control:

개인, 그룹의 Identity에 근거하여 객체에 대한 접근을 제한하는 방법

강제적 접근 통제 MAC, Mandatory Access Control:

주체가 갖는 접근 허가 권한에 근거하여 객체에 대한 접근을 제한하는 방법

역할 기반 접근 통제 RBAC, Role Based Access Control:

중앙 관리자가 조직 내 맡은 역할에 기초하여 자원에 대한 접근을 제한하는 방법

요구기능/ Admin 기능: RBAC 요소들과 관계들을 생성, 삭제, 관리 가능

Admin 리뷰: RBAC 요소들과 관계들을 조회 가능

System level 기능: Role활성화, 비활성화, Role활성화 제약조건 등

접근 통제 보호 모델

벨-라파둘라 모델 BLP, Bell-LaPadula Confidentiality Model: (기밀성 모델)

미 국방부지원 보안 모델, 보안 요소 중 기밀성을 강조, 강제적 정책에 의해 접근 통제하는 모델

보안정책: 정보가 높은 레벨에서 낮은 레벨로 흐르는 것을 방지

No Read Up: 보안수준이 낮은 주체는 보안 수준이 높은 객체를 읽어서는 안 됨

No Write Down: 보안수준이 높은 주체는 보안 수준이 낮은 객체에 기록하면 안 됨

비바 모델 Viva Integrity Model: (무결성 모델):

무결성을 보장하는 최초 모델

No Read Down: 높은 등급의 주체는 낮은 등급의 객체를 읽을 수 없음

No Write Up: 낮은 등급의 주체는 상위 등급의 객체를 수정할 수 없음

클락-윌슨 모델 Clark-Wilson Integrity Model:

무결성 중심의 상업용 모델, 금융, 예측 가능하고 완전하게 처리되는 자료처리 정책

암호 알고리즘 Encryption Algorithm: 데이터의 무결성 및 기밀성 확보를 위해 정보를 쉽게 해독할 수 없는 형태로 변환하는 기법

양방향 방식: 대칭 키 암호 방식, 비대칭 키 암호 방식

일방향 해시함수 방식: MDC, MAC(Message Authentication Code)

대칭 키 암호 방식 Symmetric-key Algorithm: 암호화와 복호화에 같은 암호 키를 쓰는 알고리즘

블록 암호방식 Block Cipher: 고정 길이 블록을 암호화, 반복하는 알고리즘

EDS, AES(Advanced Encryption Standard), SEED

일방향(단방향) 암호 방식(해시 암호 방식): 임의 길이의 정보를 입력받아, 고정된 길이의 암호문(해시값)을 출력하는 암호 방식

MAC, Message Authentication Code: 키를 사용하는 메시지 인증 코드로, 메시지의 무결성과 송신자의 인증 보장

MDC, Modification Detection Code: 키를 사용하지 않는 변경 감지 코드로, 메시지의 무결성 보장

대칭키 암호화 알고리즘

DES, Data Encryption Standard, 데이터 암호화 표준: 1975년 IBM 개발, 대칭 키 기반

SEED, 시드 블록 암호 알고리즘: 1999년 한국인터넷진흥원(KISA) 개발

AES, Advanced Encryption Standard: 2001년 미국 표준기술 연구소(NIST) 개발

ARIA: 2004년 국가정보원과 산학연구협회가 개발

IDEA, International Data Encryption Algorithm, 국제 암호 알고리즘: DES 대체, 스위스 연방기술기관 개발

LFSR, Linear Feedback Shift Register, 선형 귀환 시프트 레지스터: 선형함수 계산, 스트림 암호화 알고리즘

비대칭 키 암호화 알고리즘

디피-헬만: 최초의 공개키 알고리즘

RSA: 1977년 MIT 개발

ElGamal: 1984년 개발

ECC: 1985년 RSA 대안으로 개발

해시 암호화 알고리즘

MD5: MD4 개선한 암호화 알고리즘, 파일의 무결성 검사에 사용

SHA-1: 1993년 NSA에 미국 정부 표준 지정

SHA-256/384/512: 256비트의 해시값을 생성하는 해시함수

HAS-160: 국내 표준 서명 알고리즘

HAVAL: 메시지를 1024bits 블록으로 나눔

IPSec(Internet Protocol Security): 무결성과 인증을 보장하는 인증 헤더와 기밀성을 보장하는 암호화를 이용한 IP 보안 프로토콜
SSL, Secure Socket Layer/TLS, Transport Layer Security:
클라이언트와 서버간의 웹데이터 암호화(기밀성), 상호 인증 및 전송 시 데이터 무결성을 보장하는 보안 프로토콜
S-HTTP(Secure Hypertext Transfer Protocol): 웹 상에서 네트워크 트래픽을 암호화하는 방법

※ 소프트웨어 개발 보안 구현 ★★★

시큐어 코딩Secure Coding 가이드: 설계 및 구현 단계에서 해킹 등의 공격을 유발할 수 있는 잠재적인 보안 취약점을 사전에 제거,
외부 공격으로부터 안전한 소프트웨어를 개발하는 기법
보안 취약점: 운영 단계의 보안 리스크
보안 약점: 개발 단계의 보안 리스크

입력 데이터 검증 및 표현: 입력 데이터로 인해 발생하는 문제들을 예방하기 위해, 구현 단계에서 검증해야 하는 보안 점검 항목
입력 데이터 검증 및 표현 취약점
XSS, Cross Site Script: 검증되지 않은 외부 입력 데이터가 포함된 웹페이지를 사용자가 열람함으로서
웹페이지에 포함된 부적절한 스크립트가 실행되는 공격
사이트 간 요청 위조, CSRF, Cross-Site Request Forgery:
사용자가 자신이 의지와는 무관하게 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 공격
대책: 입력화면 폼 POS방식 사용, 세션별 CSRF토큰 사용
SQL삽입(Injection): 악의적인 SQL구문 삽입, 실행 > 데이터베이스의 접근을 통해 정보를 탈취하거나 조작 등 행위를 하는 공격
대책: 변수 타입 지정, 사용자 입력값 모두 체크하여 필터링

보안 기능: 소프트웨어 개발 단계에서 인증, 접근제어, 기밀성, 암호화, 권한 관리 등을 적절하게 구현하기 위한 보안 점검 항목
에러 처리: 프로그램 실행 시 발생하는 에러 예외 처리불가, 에러에 중요한 정보가 포함될 때 발생하는 취약점 예방, 보안 점검 항목
세션 통제: 세션과 관련되어 발생할 수 있는 취약점을 예방하기 위한 보안 점검 항목
코드 오류: 개발자의 실수로 발생하는 에러를 예외 처리하지 못하거나, 에러에 중요한 정보가 포함될 때 발생할
캡슐화: 외부에 은닉이 필요한 중요한 데이터가 인가되지 않은 사용자에게 노출되지 않게 보안 취약점 예방을 위한 보안 검증 항목
API오용: 보안이 취약한 API를 오용하여 발생할 수 있는 보안 취약점 예방을 위한 보안 검증 항목

네트워크 보안 솔루션
방화벽 Firewall: 기업 내부, 외부 간 트래픽을 모니터링하여 시스템의 접근을 허용/차단하는 시스템
웹 방화벽 WAF, Web Application Firewall: 웹 애플리케이션 보안에 특화된 보안 장비
네트워크 접근 제어 NAC, Network Access Control: 단말기가 내부 네트워크에 접속을 시도할 때 제어하고 통제 기능, 솔루션
침임 탐지 시스템 IDS, Intrusion Detection System: 네트워크 발생 이벤트 모니터링, 보안정책 위반 행위 실시간 탐지 시스템
무선 침입 방지 시스템 WIPS, Wireless Intrusion Prevention System:
무선 단말기의 접속 자동 탐지 및 차단, 보안이 취약 무선공유기를 탐지하는 시스템
통합 보안 시스템 UTM, Unified Threat Management: 다양한 보안 장비의 기능을 하나의 장비로 통합하여 제공하는 시스템
가상사설망 VPN, Virtual Private Network: 인터넷과 같은 공중망에 인증, 암호화, 터널링 기술로 전용망 효과, 보안 솔루션

시스템 보안 솔루션
스팸 차단 솔루션 Anti-Spam Solution: 메일 서버 앞단에 위치, Proxy 메일 서버로 동작
보안 운영체제 Secure OS: 컴퓨터 운영체제의 커널에 보안기능을 추가한 솔루션

콘텐츠 유출 방지 솔루션
보안 USB: 정보 유출 방지 등의 보안 기능을 갖춘 USB메모리
데이터 유출 방지 DLP, Data Loss Prevention: 조직 내부의 중요 자료가 외부로 빠져나가는 것을 탐지하고 차단하는 솔루션
디지털 저작권 관리 DRM, Digital Right Management: 디지털 저작물에 대한 보호와 관리를 위한 솔루션

소프트웨어 개발 보안테스트 유형
정적 분석 Static Analysis: SW를 실행하지 않고 보안 약점을 분석, 개발 단계 / 멈춰있는 소스코드를 분석
동적 분석 Dynamic Testing : SW를 실행환경에서 보안 약점 분석, 시험 단계 / 실행하는 과정에서 문제가 발생하지 않는지

비즈니스 연속성 계획 BCP, Business Continuity Plan:
각종 재해, 장애, 재난으로부터 위기관리 기반 재해복구, 업무복구 및 재개, 비상계획을 통해 비즈니스 연속성을 보장하는 체계

# [[ 애플리케이션 테스트 관리 ]]

※ 애플리케이션 테스트 케이스 설계 ★★★

애플리케이션 테스트: 애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차

애플리케이션 테스트 원리

- 완벽한 테스팅은 불가능: 결함은 줄일 수 있으나, 없다고 증명할 수 없음
- 파레토 법칙 Pareto Principle: 20%에 해당하는 코드에서 전체 결함의 80%가 발견된다는 법칙
- 살충제 패러독스 Pesticide Paradox: 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 현상
- 정황 의존성: 소프트웨어 성격에 맞게 테스트 실시
- 오류-부재의 궤변: 요구사항을 충족시켜주지 못하면, 결함이 없다고 해도 품질이 높다고 볼 수 없다.

프로그램 실행 여부에 따른 분류

- 정적 테스트 Static Test: 테스트 대상을 실행하지 않고 구조를 분석하여, 논리성을 검증하는 테스트(리뷰, 정적 분석)
- 동적 테스트 Dynamic Test: 소프트웨어를 실행하는 방식으로 테스트를 수행하여, 결함을 검출하는 테스트 (화이트박스 테스트, 블랙박스 테스트, 경험기반 테스트)

화이트박스 테스트 White-Box Test: 원시 코드의 논리적인 모든 경로를 테스트하여, 테스트 케이스를 설계하는 방법(구조 검사)

- 구문(문장) 커버리지: 프로그램 내의 모든 명령문을 적어도 한 번 수행하는 커버리지
- 결정(선택, 분기) 커버리지: 결정 포인트 내의 전체 조건식이 적어도 한번은 참과 거짓의 결과가 되도록 수행하는 커버리지
- 조건 커버리지: 결정 포인트 내의 각 개별 조건식이 적어도 한번은 참과 거짓의 결과가 되도록 수행하는 커버리지
- 조건/결정 커버리지: 전체 조건식 & 개별 조건식 모두 참 한번, 거짓 한 번 결과가 되도록 수행하는 커버리지
- 변경 조건/결정 커버리지: 개별 조건식이 다른 개별 조건식에 영향 받지x, 전체 조건식에 독립적으로 영향을 주는 커버리지
- 다중 조건 커버리지: 결정 조건 내 모든 개별 조건식의 모든 가능한 조합을 100%보장하는 커버리지
- 기본 경로 커버리지: 수행 가능한 모든 경로를 테스트하는 기법
- 제어 흐름 테스트: 프로그램 제어 구조를 그래프 형태로 나타내어 내부 로직을 테스트하는 기법
- 데이터 흐름 테스트: 제어 흐름 그래프에 사용현황을 추가한 테스트 기법

블랙박스 테스트 Black-Box Test:

- 동등 분할 테스트: 입력 데이터 영역을 유사한 도메인별로 유효값/무효값 그룹핑해 대푯값 테스트 케이스를 도출, 테스트 기법
- 경곗값 분석 테스트: 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법
- 결정 테이블 테스트: 요구사항의 논리와 발생조건을 테이블 형태로 나열하여, 조건과 행위를 모두 조합해 테스트
- 상태 전이 테스트: 이벤트에 의해 어느 한 상태에서 다른 상태로 전이 되는 경우의 수를 수행하는 테스트
- 유스케이스 테스트: 유스케이스로 모델링 되어 있을 때 프로세스 흐름을 기반으로 테스트 케이스를 명세화해 수행하는 테스트
- 분류 트리 테스트: SW의 일부 또는 전체를 트리구조로 분석 및 표현하여 테스트 케이스 설계하는 테스트
- 페어와이즈pairwise 테스트: 테스트 데이터 값들 간에 최소한 한 번 씩을 조합하는 방식
- 원인-결과 그래프 테스트: 그래프를 활용, 입력 데이터 간 관계 및 출력에 미치는 영향 분석, 효용성 높은 케이스 선정 테스트
- 비교 테스트: 여러 버전의 프로그램에 같은 입력값을 넣어 동일한 데이터가 나오는지 비교하는 테스트

테스트 시각에 따른 분류

- 검증 Verification: 소프트웨어 개발 과정을 테스트, 개발자 또는 시험자의 시각
- 확인 Validation: 소프트웨어 결과를 테스트, 사용자 시각

테스트 목적에 따른 분류

- 회복 테스트 Recovery Testing: 시스템에 고의로 실패를 유도하고, 시스템의 정상적 복귀 여부를 테스트하는 기법
- 안전 테스트 Security Testing: 불법 소프트웨어가 접근해 시스템을 파괴 못하도록 소스코드 내의 보안적인 결함을 미리 점검
- 성능 테스트 Performance Testing: 시스템 응답시간, 특정 시간 내 처리 업무량, 시스템 반응 속도 등을 측정하는 테스트 기법
- 구조 테스트 Structure Testing: 시스템 내부 논리 경로, 소스코드의 복잡도를 평가하는 테스트 기법
- 회귀 테스트 Regression Testing: 시스템의 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트
- 병행 테스트 Parallel Testing: 변경된 시스템과 기존 시스템에 동일한 데이터를 입력 후 결과를 비교하는 테스트 기법

성능 테스트의 상세 유형

- 부하 테스트 Load Testing: 시스템에 부가를 계속 증가시켜, 시스템의 임계점을 찾는 테스트
- 강도 테스트 Stress Testing: 임계점 이상 부하를 가하여 비정상적 상황에서의 처리 테스트
- 스파이크 테스트 Spike Testing: 짧은 시간에 사용자가 몰릴 때 시스템의 반응 측정 테스트
- 내구성 테스트 Endurance Testing: 오랜 시간 동안 시스템에 높은 부하를 가하여 시스템의 반응을 테스트

테스트 종류에 따른 분류

명세 기반 테스트(블랙박스 테스트): 프로그램의 요구사항 명세서를 기반으로 테스트 케이스를 선정하여 테스트하는 기법
구조 기반 테스트(화이트박스 테스트): 소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트 기법
경험 기반 테스트(블랙박스 테스트): 유사 소프트웨어나 기술 등에 대한 테스터의 경험을 기반으로 수행하는 테스트

테스트 케이스 Test Case: 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 테스트 항목에 대한 명세서
테스트 오라클 Test Oracle: 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법

참 오라클 True Oracle: 모든 입력값에 대하여 기대하는 결과를 제공하는 오라클
샘플링 오라클 Sampling Oracle: 특정한 몇 개의 입력값에 대해서만 기대하는 결과를 제공해주는 오라클
휴리스티 오라클 Heuristic Oracle: Heuristic=추정, 특정 입력값에 대해 올바른 결과 제공, 나머지값에 대해 추정으로 처리함
일관성 검사 Consistent Oracle: 애플리케이션 변경이 있을 때, 수행 전과 후의 결과값이 동일한지 확인하는 오라클

테스트 레벨 Test Level: 함께 편성되고 관리되는 테스트 활동 그룹
테스트 레벨 종류

단위 테스트: 사용자 요구사항에 대한 모듈, 서브루틴 등을 테스트하는 단계
통합 테스트: 단위테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트
시스템 테스트: 개발된 소프트웨어가 정상적으로 수행되는지 검증하는 테스트
인수 테스트: 계약상의 요구사항이 만족되었는지 확인하기 위한 테스트

소프트웨어 개발 단계: 요구사항 -> 분석 -> 설계 -> 구현
테스트 단계: 단위 테스트(unit) -> 통합 테스트(integration) -> 시스템 테스트(블랙박스/화이트박스) -> 인수 테스트(알파/베타)
인수 테스트

사용자 인수 테스트: 사용자가 시스템 사용의 적절성 여부를 확인
운영상의 인수 테스트: 시스템 관리자가 시스템 인수 시 수행하는 테스트 기법
계약 인수 테스트: 계약상의 인수/검수 조건을 준수하는지 여부를 확인
규정 인수 테스트: 소프트웨어가 정부 지침, 법류, 규정 등에 맞게 개발되었는지 확인
알파 테스트: 개발자의 장소에서 사용자가 개발자와 함께 행하는 테스트 기법
베타 테스트: 실제 사용자에게 대상 소프트웨어를 사용하게 하고 피드백을 받는 테스트

테스트 시나리오 Test Scenario: 테스트 수행을 위한 여러 테스트 케이스의 집합, 테스트 케이스의 동작 순서를 기술한 문서이며 테스트를 위한 절차를 명세한 문서

※ 애플리케이션 테스트 케이스 설계 ★★★

단위 테스트 Unit Test: 개별적인 모듈 또는 컴포넌트를 테스트
목Mock 객체 생성 프레임워크:

객체 지향 프로그램에서는 컴포넌트 테스트 수행 시, 테스트되는 메서드는 다른 클래스의 객체에 의존
메서드를 고립화하여 테스트가 불가함 => 독립적 컴포넌트 테스트를 위해서 Mock 객체 필요(STUB의 객체지향ver)
(유형: 더미 객체, 테스트 스텁, 테스트 드라이버, 테스트 스파이, 가짜 객체)

통합 테스트 Integration Test: 단위 테스트가 완료된 모듈들을 결합, 하나의 시스템으로 완성시키는 과정에서의 테스트

| | |
|---|---|
| 빅뱅 테스트: 모든 모듈을 동시에 통합 후 테스트 수행 | 실제 모듈로 테스트 |
| 상향식 테스트: 최하위 모듈부터 점진적으로 상위 모듈과 테스트 | 테스트 드라이버 필요 |
| 하향식 테스트: 최상위 모듈부터 점진적으로 하위 모듈과 테스트 | 테스트 스텁 필요 |
| 샌드위치 테스트: 상위는 하향식+하위는 상향식 테스트 | 테스트 스텁+테스트 드라이버 필요 |

테스트 자동화 도구:

반복적 테스트 작업을 스크립트 형태로 구현, 테스트 시간 단축과 인력 투입 비용을 최소화하고 쉽고 효율적인 테스트를 수행
정적 분석 도구 Static Analysis Tools: 만들어진 애플리케이션을 실행하지 않고 분석하는 도구
테스트 실행 도구 Test Execution Tools: 테스트를 위해 작성된 스크립트를 실행, 스크립트 언어를 사용해 테스트 실행 도구
성능 테스트 도구 Performance Test Tools: 가상의 사용자를 생성, 테스트를 수행해 목표를 달성했는지 확인하는 도구
테스트 통제 도구 Test Control Tools: 테스트 계획 및 관리, 테스트 수행, 결함 관리 등을 수행하는 도구
테스트 하네스 도구 Test Harness Tools: 테스트 실행될 환경을 시뮬레이션, 컴포넌트&모듈 정상적 테스트되도록 하는 도구

테스트 하네스 구성요소: 테스트 드라이버, 테스트 스텁, 테스트 슈트, 테스트 케이스, 테스트 시나리오, 테스트 스크립트, 목 오브젝트
소프트웨어 결함: 개발자 오류로 인해, 문서 또는 코딩 결점으로 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생하는 현상
테스트 결함 관리: 단계별 테스트 수행 후 발생한 결함의 재발 방지+유사 결함 발견 시 처리시간 단축을 위해, 결함 추적 + 관리 활동

결함 분석 방법

구체화: 결함의 원인을 찾기 위해 결함을 발생시킨 입력값, 테스트 절차, 테스트 환경을 명확히 파악하는 방법

고립화: 입력값, 테스트 절차, 테스트 환경 중 어떤 요소가 결함 발생에 영향을 미치는지 분석하는 방법

일반화: 결함 발생에 영향을 주는 요소를 최대한 일반화 시키는 방법

테스트 커버리지 Test Coverage: 주어진 테스트 케이스에 의해 수행되는 소프트웨어 테스트 범위를 측정, 테스트 품질 측정 기준

기능 기반 커버리지: 전체 기능을 모수로 설정하고, 실제 테스트가 수행된 기능의 수를 측정하는 방법

라인 커버리지: 전체 소스 코드의 라인 수를 모수로, 테스트 시나리오가 수행한 소스 코드의 라인수를 측정하는 방법

코드 커버리지: 소스 코드의 구문, 조건, 결정 등의 구조 코드 자체가 얼마나 테스트되었는지 측정하는 방법

결함 심각도별 분류:

단수 결함(미관상 안 좋음)->경미한 결함(표준위반)->보통 결함(사소한 오작동)->주요결함(기능장애)->치명적 결함(데이터 손실)

결함 우선순위: 낮음(Low) -> 보통(Medium) -> 높음(High) -> 결정적(Critical)

※ 애플리케이션 성능 개선 ★★★

애플리케이션 성능 측정 지표

처리량 Throughput: 애플리케이션이 주어진 시간에 처리할 수 있는 트랜잭션의 수

응답 시간 Response Time: 사용자 입력이 끝난 후, 애플리케이션의 응답 출력이 개시될 때까지의 시간

경과 시간 Turnaround Time: 애플리케이션에 사용자가 요구 입력시점 -> 트랜잭션 처리 후 결과 출력 완료까지 걸리는 시간

자원 사용률 Resource Usage: 애플리케이션이 트랜잭션을 처리하는 동안, CPU 사용량, 메모리 사용량, 네트워크 사용량

데이터베이스 관련 성능 저하 원인

데이터베이스 락 DB Lock: 대량의 데이터 조회, 과도한 업데이트, 인덱스 생성 시 발생하는 현상

불필요한 데이터베이스 패치 DB Fetch: 실제 필요한 데이터보다 많은 대량의 데이터 요청이 들어올 경우, 응답 시간 저하 현상

연결 누수 Connection Leak: DB 연결과 관련된 JDBC 객체를 사용 후 종료하지 않을 경우 발생

부적절한 커넥션 풀 크기 Connection Pool Size: 너무 작거나 크게 설정한 경우, 성능 저하 현상이 발생할 가능성 존재

확정 Commit 관련: 트랜잭션이 확정되지 않고, 커넥션 풀에 반환될 때 성능 저하 가능성 존재

배드 코드 Bad Code: 프로그램 로직이 복잡하고 다른 개발자들이 이해하기 어려운 코드

외계인 코드: 아주 오래되거나 참고문서 또는 개발자가 없어 유지보수 작업이 아주 어려운 코드

스파게티 코드: 소스 코드가 복잡하게 얽힌 모습, 정상 작동하지만 코드의 작동을 파악하기 어려운 코드

알 수 없는 변수명: 변수나 메서드에 대한 이름 정의를 알 수 없는 코드

로직 중복: 동일한 처리 로직이 중복되게 작성된 코드

클린 코드 Clean Code: 가독성이 높고, 단순하며, 의존성을 줄이고, 중복을 최소화하여 깔끔하게 잘 정리된 코드

클린 코드 특징: 가독성이 높아 쉽게 이해됨, 개선도 쉬움, 버그 찾기도 쉬워 프로그래밍 속도가 빨라짐

클린 코드 작성 원칙: 가독성, 단순성, 의존성 최소, 중복성 제거, 추상화

리팩토링 Refactoring: 기능을 변경하지 않고, 복잡한 소스코드를 수정, 보완하여 가용성 및 가독성을 높이는 기법

---

## [[ 애플리케이션 테스트 관리 ]]

---

※ 운영체제의 특징 ★★★

운영체제 OS, Operating System: 사용자가 컴퓨터의 하드웨어를 쉽게 사용할 수 있도록 인터페이스를 제공

운영체제 종류: 윈도즈 Windows, 유닉스 Unix, 리눅스 Linux, 맥 MAC, 안드로이드 Android

메모리 관리 기법

반입 기법: 메모리 적재 시기 결정 When

배치 기법: 메모리 적재 위치 결정 Where

할당 기법: 메모리 적재 방법 결정 How

교체 기법: 메모리 교체 대상 결정 Who

메모리 배치 기법

최초 적합 First-fit: 프로세스가 적재될 수 있는 가용 공간 중에서 첫 번째 분할에 할당하는 방식

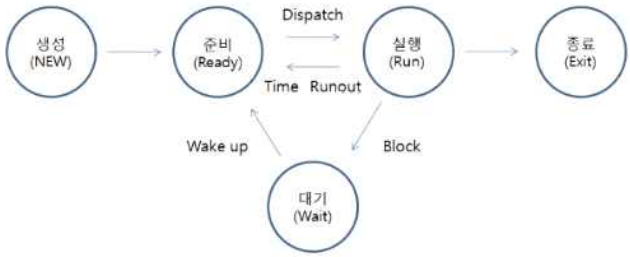
최적 적합 Best-fit: 가용 공간 중에서 가장 크기가 비슷한 공간을 선택하여 프로세스를 적제하는 방식

최악 적합 Worst-fit: 프로세스의 가용 공간 중에서 가장 큰 공간에 할당

프로세스 상태

생성 상태 Create: 사용자에 의해 프로세스가 생성된 상태
준비 상태 Ready: CPU를 할당받을 수 있는 상태
실행 상태 Running: 프로세스가 CPU를 할당받아 동작 중인 상태
대기 상태 Waiting: 프로세스 실행 중 입출력 처리 등으로 CPU를 양도하고, 입출력 처리 완료까지 대기 리스트에서 대기
완료 상태 Complete: 프로세스가 CPU를 할당받아 주어진 시간 내에 완전히 수행을 종료한 상태

프로세스 상태 전이




프로세스 스케줄링의 유형

선점형 스케줄링 Preemptive Scheduling: 우선순위가 높은 프로세스가 CPU를 점유하는 스케줄링
RR, 라운드로빈, Round Robbin: 모든 프로세스 같은 우선순위를 가지고, time slice 기반으로 스케줄링
SRT, Shortest Remaining Time: 가장 짧은 시간이 소요되는 프로세스를 먼저 수행 first
다단계 큐, Multi Level Queue: 여러 개의 큐를 이용하여 상위 단계 작업에 의한 하위 단계 작업이 선점
다단계 피드백 큐, Multi Level Feedback Queue: 큐마다 서로 다른 CPU시간 할당량 부여, FIFO+라운드로빈 혼합
비선점 스케줄링 Non-preemptive Scheduling: 한 프로세스가 CPU를 할당? 작업 종료 전까지 다른 프로세스 CPU점유 불가능!
우선순위 Priority: 프로세스별 우선순위에 따라 CPU할당
기한부 Deadline: 작업들이 명시된 기한 내에 완료되도록 계획
FCFS, First Come First Service: 프로세스가 대기 큐에 도착한 순서에 따라 CPU할당 = FIFO
SJF, Shortest Job First: 가장 짧은 작업부터 수행, 평균 대기 시간 최소화, 기아현상 발생
HRN, Highest Response Ratio Next: 대기 중인 프로세스 중 현재 응답률 가장 높은 것 선택, 기아현생 최소화

가상화 Virtualization: 물리적인 리소스들을 사용자에게 하나로 보이게 하거나, 하나의 물리적인 리소스를 여러 개로 보이게 하는 기술
클라우드 컴퓨팅 Cloud Computing: 인터넷을 통해 가상화된 컴퓨터 리소스 제공
클라우드 컴퓨팅 유형
인프라형 서비스 IaaS, Infrastructure as a Service: 서버, 스토리지 같은 시스템 자원을 클라우드로 제공하는 서비스
플랫폼형 서비스 PaaS, Platform as a Service: 애플리케이션 개발, 실행, 관리하는 플랫폼을 제공하는 서비스
소프트웨어형 서비스 SaaS: Software as a Service: 클라이언트를 통해 접속하여 소프트웨어 서비스 형태로 이용하는 서비스

※ 네트워크 계층 구조 파악 ★★★

네트워크 Network: 원하는 정보를 원하는 수신자 또는 기기에 정확하게 전송하기 위한 인프라
광대역 네트워크 WAN: LAN에 비해 전송거리가 넓음
근거리 네트워크 LAN: 한 건물 또는 작은 지역을 커버

OSI 모형, Open Systems Interconnection Reference Model / OSI 7계층 = OSI 7 Layer
응용 계층 Application Layer: 사용자와 네트워크 간 응용서비스 연결, 데이터 생성
표현 계층 Presentation Layer: 데이터 형식 설정, 부호교환, 암/복호화
세션 계층 Session Layer: 송수신 간의 논리적인 연결
전송 계층 Transport Layer: 송수신 프로세스 간의 연결
네트워크 계층 Network Layer: 단말기 간 데이터 전송을 위한 최적화된 경로 제공
데이터링크 계층 Data Link Layer: 인접 시스템 간 데이터 전송, 전송 오류 제어
물리 계층 Physical Layer: 0과 1의 비트 정보를 회선에 보내기 위한 전기적 신호 변환

프로토콜 Protocol: 서로 다른 시스템이나 기기들 간의 데이터 교환을 원활히 하기 위해 표준화된 통신규약
프로토콜 기본 3요소
구문 Syntax: 시스템 간의 정보 전송을 위한 데이터 형식, 코딩, 신호 레벨 등의 규정
의미 Semantic: 시스템 간의 정보 전송을 위한 제어 정보로, 조정과 에러 처리를 위한 규정
타이밍 Timing: 시스템 간의 정보 전송을 위한 속도 조절과 순서 관리 규정

네트워크 프로토콜 Network Protocol: 컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고 받는 양식과 규칙의 체계

데이터 링크 계층 프로토콜

HDLC, High-level Data Link Control: 점대 점 방식이나 다중방식 통신에 사용되며, 동기식 비트 중심

PPP, Point-to-Point Protocol: 네트워크 분야에서 두 통신 노드 간의 직접적인 연결

프레임 릴레이 Frame Relay: 프로토콜 처리를 간략화, 데이터 처리속도의 향상 및 전송 지연을 감소시킨 고속데이터 전송기술

ATM, Asynchronous Transport Mode: 54바이트 셀 단위로 전달하는 비동기 시분할 다중화 방식의 패킷형 전송 기술

네트워크 계층 프로토콜

IP, Internet Protocol: 송수신 간의 패킷 단위, 데이터를 교환하는 네트워크에서 정보를 주고 받는데 사용

ARP, Address Resolution Protocol: IP네트워크상에서 IP주소를 MAC주소(물리주소)로 변환하는 프로토콜

RARP, Reverse Address Resolution Protocol: 서버로부터 IP주소를 요청하기 위해 사용하는 프로토콜

ICMP, Internet Control Message Protocol: IP 패킷을 처리할 때 발생하는 문제를 알려주는 프로토콜

IGMP, Internet Group Management Protocol: 호스트 컴퓨터와 인접 라우터가 멀티캐스트 그룹 멤버십을 구성하는데 사용

라우팅 프로토콜, Routing Protocol: 데이터 전송을 위해 최적의 경로를 설정해주는 라우터 간의 상호 통신 프로토콜

IPv4, Internet Protocol version 4: 인터넷에서 사용되는 패킷 교환, 네트워크상 데이터를 교환 32비트 주소체계, 네트워크 계층 프로토콜

8비트씩 4부분(옥텟)으로 나뉜 10진수

0~255까지 3자리수 표현 / 254

유니캐스트, 멀티캐스트, 브로드캐스트

IPv4 클래스 분류: IP낭비 발생 => 현재 사용X, 서브넷마스크 개념 도입

| | | |
|---|---|---|
| A클래스: 국가나 대형 통신망에 사용 | 1~127 | 옥텟3 |
| B클래스: 중대형 통신망에 사용 | 128~191 | 옥텟2 |
| C클래스: 소규모 통신망에 사용 | 192~223 | 옥텟1 |
| D클래스: 멀티캐스트 용도로 예약된 주소 | 224~239 | 멀티캐스트 |
| E클래스: 연구를 위해 예약된 주소 | 240~255 | (공용X) |

IPv4에서 IPv6으로 전환기술

듀얼스택 Dual Stack: IPv4+IPv6 모두 처리

터널링 Tunneling: IPv6망이 인접 IPv4망을 거쳐갈 때 터널을 통해 통신 (도입 초기 소규모 섬 형태의 IPv6 망의 통신)

변환 Translation: IPv6시스템이 IPv4 수신자가 이해할 수 있는, 또는 반대로 헤더 변환하는 기술

라우팅 프로토콜

내부 라우팅 프로토콜 IGP, Interior Gateway Protocol

RIP, Routing Information Protocol: AS(자율시스템)내 사용, 거리 벡터 알고리즘 기초해 개발된 내부라우팅 프로토콜

OSPF, Open Shortest Path First: 자신을 기준으로 링크 상태 알고리즘을 적용, 최단경로를 찾는 라우팅 프로토콜

외부 라우팅 프로토콜 EGP, Exterior Gateway Protocol

BGP, Border Gateway Protocol: AS상호 간에 경로 정보를 교환하기 위한 라우팅 프로토콜

TCP, Transmission Control Protocol, 특징 => 신뢰성 보장, 연결 지향적 특징, 흐름 제어, 혼잡 제어

UDP, User Datagram Protocol, 특징 => 비신뢰성, 순서화되지 않은 데이터그램 서비스 제공, 실시간 응용 및 멀티태스크 가능, 단순 헤어

표현계층 프로토콜: JPEG 이미지를 위해 만들어진 표준 규격 / MPEG, 멀티미디어(비디오, 오디오)를 위해 만들어진 표준 규격

응용계층 프로토콜

HTTP, Hyper Text Transfer Protocol: 텍스트 기반의 통신규약, 하이퍼텍스트를 빠르게 교환하기 위한 프로토콜

FTP, File Transfer Protocol: TCP/IP를 가지고 서버와 클라이언트 사이의 파일을 전송하기 위한 프로토콜

SMTP, Simple Mail Transfer Protocol: 인터넷에서 TCP포트 25번을 사용, 이메일을 전송하는 프로토콜

POP3, Post Office Protocol version3: 원격 서버로부터 TCP/IP 연결을 통해 이메일을 수신하는 프로토콜

IMAP, Internet Messaging Access Protocol: 원격 서버로부터 TCP/IP 연결을 통해 이메일을 수신하는 프로토콜

Telnet: 인터넷이나 로컬 영역 네트워크 연결에 사용되는 네트워크 프로토콜

SSH, Secure Shell: 서로 연결되어 있는 컴퓨터 간 원격 명령 실행이나 쉘 서비스 등을 수행, Telnet보다 보안이 강력함

SNMP, Simple Network Management Protocol: 라우터나 허브 등 네트워크 장치로부터 정보를 수집 및 관리

패킷 교환 방식 Packet Switching: 작은 블록 패킷으로 데이터를 전송, 데이터를 전송하는 동안만 네트워크 자원을 사용(메일, 메시지)

패킷 교환 방식 기술

X.25: 통신을 원하는 두 단말장치가 패킷 교환망을 통해 패킷을 원활히 전달하기 위한 통신 프로토콜

프레임 릴레이: ISDN(종합정보통신망)을 사용하기 위한 프로토콜로 ITU-T에 의해 표준으로 작성

ATM, Asynchronous Transfer Mode: 비동기 전송 모드, 광대역 전송에 쓰이는 스위칭 기법

서킷 교환 방식 Circuit Switching: 네트워크 리소스를 특정 사용층이 독점하도록 하는 통신 방식(영상, 비디오)
애드 혹 네트워크 Ad-Hoc Network: 노드들에 의해 자율적으로 구성되는 기반 구조가 없는 네트워크

※ 개발환경 인프라 구성 방식 ★
개발환경 인프라 구성 방식
온프레미스 방식 On-Premise: 외부 인터넷망이 차단된 상태에서 인트라넷망만을 활요하여 개발환경을 구축하는 방식
클라우드 방식 Cloud: 클라우드 공급 서비스를 제공하는 회사들(아마존, 구글, MS)의 서비스를 임대하여 개발 환경을 구축
하이브리드 방식 Hybrid: 온프레미스+클라우드 방식을 혼용한 방식

---

## [[ 제품 소프트웨어 패키징 ]]

---

※ 제품 소프트웨어 패키징하기 ★★
제품 소프트웨어 패키징 Product Software Packaging: 개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 포장
사용자 중심의 모듈 패키징 프로세스:
기능 식별 -> 모듈화 -> 빌드 진행 -> 사용자 환경 분석 -> 패키징 적용 시험 -> 패키징 변성 개선

릴리즈 노트 Release Note: 고객에게 개발 과정에서 정리된 제품의 릴리즈 정보를 제공하는 문서
릴리즈 노트 작성 프로세스:
모듈 식별 > 릴리즈 정보 확인 > 릴리즈 노트 개요 작성 > 영향도 체크 > 정식 릴리즈 노트 작성 > 추가 개선 항목 식별
제품 소프트웨어 패키징 도구: 배포 패키징 시에 디지털 콘텐츠 지적 재산권 보호, 관리하는 기능 제공.. 안전한 유통 및 배포 보장 도구
패키징 도구 활용 시 고려 사항
암호화/보안 고려: 내부 콘텐츠에 대한 암호화 및 보안 고려
이기종 연동을 고려: 이기종 콘텐츠 및 단말기 간 DRM 연동 고려
사용자 편의성 고려: 사용자 입장에서 문제를 고려, 최대한 효율적으로 적용
적합한 암호화 알고리즘 적용: 제품 소프트웨어 종류에 맞는 알고리즘 선택, 배포 시 범용성에 지장이 없도록 고려

저작권 Copyright: 창작물인 저작물에 대한 배타적 독점적 권리로 타인의 침해를 받지 않을 고유한 권한
디지털 저작권 관리 DRM, Digital Right Management: 저작권자가 배포한 디지털 콘텐츠, 의도한 용도로 사용되도록 관리 및 보호
디지털 저작권 관리(DRM) 구성요소
콘텐츠 제공자 Content Provider: 콘텐츠를 제공하는 저작권자
콘텐츠 소비자 Content Customer: 콘텐츠를 구매하여 사용하는 주체
콘텐츠 분배자 Content Distributor: 암호화된 콘텐츠를 유통하는 곳 또는 사람
클리어링 하우스 Clearing House: 저작권에 사용 권한, 라이선스 발급, 암호화된 키 관리, 사용량 결제 관리 수행하는 곳
DRM 콘텐츠 DRM Content 서비스할 암호화된 콘텐츠, 관련된 메타데이터, 콘텐츠 사용정보를 패키징 구성한 콘텐츠
패키저 Packager: 콘텐츠를 메타데이터와 함께 배포 가능한 단위로 묶은 도구
DRM 컨트롤러 DRM Controller: 배포된 디지털 콘텐츠의 이용 권한을 통제
보안 컨테이너 Security Container: 원본 콘텐츠를 안전하게 유통하기 위한 전자적 보안 장치

패키징 도구 구성 세부 기술
암호화: 콘텐츠 및 라이선스를 암호화하고 전자 서명
키 관리: 콘텐츠를 암호화한 키에 대한 저장 및 분배
암호화 파일 생성: 암호화한 콘텐츠로 생성하기 위한 기술
식별 기술: 콘텐츠에 대한 식별 체계 표현 기술
저작권 표현: 라이선스의 내용 표현 기술
정책 관리: 라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
크랙방지: 크랙에 의한 콘텐츠 사용 방지 기술
인증: 라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

※ 제품 소프트웨어 매뉴얼 작성 및 버전 등록 ★
제품 소프트웨어 매뉴얼: 사용자 측면에서 패키징 이후 설치, 제품 소프트웨어를 사용하는 데 필요한 주요 내용을 기록한 문서
제품 소프트웨어 사용자 매뉴얼: 사용자가 소프트웨어 사용에 필요한 내용을 포함한 문서
제품 소프트웨어 배포본: 사용자가 사용하기 편하도록 배포 정보를 포함하여 개발된 컴포넌트 또는 패키지가 제품화된 형태
백업 유형
전체 백업 Full Backup: 백업받고자 하는 데이터 전체에 대한 백업을 하는 방식
차등 백업 Differecntial Backup: 마지막 전체 백업 이후 변경된 모든 데이터를 백업하는 방식
증분 백업 Incremental Backup: 정해진 시간을 기준으로 그 이후에 변경된 파일만을 백업하는 방식