

1. 요구사항 확인 (중요도: ★)

1-1. 소프트웨어 개발방법론

소프트웨어 생명주기 모델 : 시스템의 요구분석부터 유지보수까지 전 공정을 체계화한 절차

프로세스 : 요구사항 분석 – 설계 – 구현 – 테스트 – 유지보수

모델 종류 : 폭포수 모델, 프로토타이핑 모델, 나선형 모델, 반복적 모델 (폭포나반)

나선형 모델 절차 : 계획 및 정의 – 위험 분석 – 개발 – 고객 평가 (계위개고)

소프트웨어 개발 방법론 종류 : 구조적 방법론, 정보공학 방법론, 객체지향 방법론, 컴포넌트기반 방법론, 애자일 방법론, 제품 계열 방법론

구조적 방법론 : 기능에 따라 나누어 개발, 분할과 정복 방식, 프로세스 중심의 하향식 방법론

정보공학 방법론 : 정보시스템 개발에 필요한 관리 철자를 체계화한 방법론

객체지향 방법론 : 객체라는 기본단위로 시스템을 분석, 설계. 객체, 클래스, 메시지를 사용

컴포넌트 기반 방법론 : 컴포넌트를 조립해서 하나의 응용 프로그램을 작성하는 방법론

애자일 방법론 : 절차보다는 사람이 중심이 되어 신속 적응적 경량 개발방법론

제품 계열 방법론 : 특정 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론

애자일 방법론 유형 : XP(eXtreme Programming), 스크럼, 린

XP : 의사소통 개선과 즉각적 피드백으로 소프트웨어 품질을 높이기 위한 방법론

XP의 5가지 가치 : 용기, 단순성, 의사소통, 피드백, 존중 (용단의피존)

XP의 12가지 기본원리 : 짹 프로그래밍, 공동 코드 소유, 지속적인 통합, 계획 세우기, 작은 릴리즈, 메타포어, 간단한 디자인, 테스트 기반 개발, 리팩토링, 40시간 작업, 고객 상주, 코드 표준

메타포어 : 공통적인 이름 체계와 시스템 서술서를 통해 의사소통을 원활하게 하는 원리

리팩토링 : 프로그램의 기능을 바꾸지 않으면서 중복제거, 단순화 등을 위해 시스템을

재구성

스크럼 : 매일 정해진 시간, 장소에서 짧은 시간 개발을 위한 프로젝트 관리 중심 방법론

스크럼 주요 개념 : 백로그, 스프린트, 스크럼 미팅, 스크럼 마스터, 스프린트 회고, 번다운 차트

린 : 도요타의 린 시스템 품질기법을 소프트웨어 개발 프로세스에 적용해서 낭비 요소를 제거하여 품질을 향상시킨 방법론

린의 7가지 원칙 : 낭비제거, 품질 내제화, 지식 창출, 늦은 확정, 빠른 인도, 사람 존중, 전체 최적화

비용산정 모형 : 하향식 산정방법, 상향식 산정방법

하향식 산정방법 : 전문가에게 비용산정을 의뢰하거나 전문과와 조정가를 통해 산정하는 방식

상향식 산정방법 : 세부적인 요구사항과 기능에 따라 필요한 비용을 계산하는 방식

비용산정 모형 종류 : LoC(lines of code) 모형, Man Month 모형, COCOMO(constructive cost model) 모형, 푸트남 모형, FP(기능 점수) 모형

LoC 모형 : 소프트웨어 각 기능의 원시 코드 라인 수의 낙관치, 중간치, 비관치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 방식

Man Month 모형 : 한 사람이 1개월동안 할 수 있는 일의 양을 기준으로 비용을 산정하는 방식 ([계산 문제](#))

COCOMO 모형 : 보행이 제안한 모형으로 프로그램 규모에 따라 비용을 산정하는 방식

COCOMO의 소프트웨어 개발 유형: 조직형(소규모), 반 분리형(중규모), 임베디드형(초대형 규모)

푸트남 모형 : 소프트웨어 개발주기의 단계별로 요구할 인력의 분포를 가정하는 방식

기능점수 모형: 요구 기능을 증가시키는 인자별로 가중치를 부여하고, 요인별 가중치를 합산하여 총 기능 점수를 계산하여 비용 산정하는 방식

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

일정관리 모델 종류: 주 공정법, PERT(Program Evaluation and review Technique), 중요 연쇄 프로젝트 관리(CCPM)

주 공정법: 여러 작업의 수행 순서가 얹혀있는 프로젝트의 일정을 계산하는 기법

PERT: 일의 순서를 계획치를 정리하기 위한 수령 기법으로 비관치, 중간치, 낙관치의

3점 추정방식을 통해 일정을 관리하는 방법

중요연쇄 프로젝트 관리: 주 공정 연쇄법으로 자원제약사항을 고려하여 일정을 작성하는 방법

임계 경로 계산: 시작부터 종료까지 가장 긴 시간이 걸리는 경로를 계산

1-2. 현행 시스템 분석

현행 시스템 파악: 현행 시스템이 어떤 하위 시스템으로 구성되어 있고, 제공 기능 및 연계 정보는 무엇이며 어떤 기술 요소를 사용하는지를 파악하는 활동

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

소프트웨어 아키텍처: 여러가지 소프트웨어 구성요소와 그 구성요소가 가진 특성 중에서 외부에 드러나는 특성, 구성요소 간의 관계를 표현하는 시스템의 구조

소프트웨어 아키텍처 프레임워크: 소프트웨어 집약적인 시스템에서 아키텍처가 표현해야 하는 내용 및 이들간의 관계를 제공하는 아키텍처 기술 표준

유스케이스: 시스템이 사용자에게 제공해야하는 기능으로서 시스템 요구사항이자 사용자 입장에서 바라본 시스템 기능

소프트웨어 아키텍처 4+1 뷰: 고객의 요구사항을 정리해놓은 시나리오를 4개의 관점에서 바라보는 소프트웨어적인 접근 방법 / 유스케이스 뷰, 논리 뷰, 프로세스 뷰, 구현 뷰, 배포 뷰 (유논프구배)

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

논리 뷰: 시스템의 기능적인 요구사항이 어떻게 되는지 설명해주는 뷰

프로세스 뷰: 시스템의 비기능적인 속성으로서 자원의 효율적인 사용, 이벤트 처리등을 표현

구현 뷰: 개발 환경 안에서 정적인 소프트웨어 모듈의 구성을 보여주는 뷰

배포 뷰: 컴포넌트가 물리적인 아키텍처에 어떻게 배치되는가를 매핑해서 보여주는 뷰

소프트웨어 아키텍처 패턴: 소프트웨어를 설계할 때 참조할 수 있는 전형적인 해결 방식

소프트웨어 아키텍처 패턴 유형

계층화 패턴: 시스템을 계층으로 구분하여 구성. 하위 모듈들은 추상화를 제공하고, 각 계층은 상위 계층에 서비스 제공

클라이언트-서버 패턴: 하나의 서버와 다수의 클라이언트로 구성.

파이프-필터 패턴: 데이터 스트림을 생성하고 처리하는 시스템에서 사용 가능한 패턴. 서브 시스템이 데이터를 받아 처리하고 결과를 다음 서브 시스템으로 넘겨주는 과정을 반복함

브로커 패턴: 분리된 컴포넌트들로 이루어진 분산 시스템에 사용. 컴포넌트 간의 통신을 조정하는 역할 수행

모델-뷰-컨트롤러 패턴(MVC): 대화형 애플리케이션을 모델, 뷰, 컨트롤러 3개의 서브시스템으로 구조화하는 패턴

소프트웨어 아키텍처 비용 평가 모델: 아키텍처 접근법이 품질 속성에 미치는 영향을 판단하고 아키텍처의 적합성을 평가하는 모델

SAAM: 변경 용이성과 기능성에 집중

ATAM: 아키텍처 품질 속성을 만족시키는지 판단 및 품질 속성들의 이해상충관계 평가

CBAM: ATAM 바탕의 시스템 아키텍처 분석 중심으로 경제적 의사결정에 대한 요구를 총족

ADR: 소프트웨어 아키텍처 구성요소 간 응집도를 평가하는 모델

ARID: 전체 아키텍처가 아닌 특정 부분에 대한 품질요소에 집중

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

디자인패턴 구성요소: 패턴의 이름, 문제 및 배경, 솔루션, 사례, 결과, 샘플코드(패문솔 사결샘)

디자인 패턴 종류

-생성

Builder: 복잡한 인스턴스를 조립하여 만드는 구조, 복합 객체를 생성할 때 객체를 생성하는 방법과 구현하는 방법을 분리함으로써 동일한 생성 절차에서 다른 결과를 만들 수 있음

Prototype: 원형을 만들고 이것을 복사한 후 필요한 부분만 수정하여 사용하는 패턴

Factory Method: 상위 클래스에서 객체를 생성하는 인터페이스를 정의, 하위

클래스에서 인스턴스를 생성하는 방식. 하위 클래스에서 함수를 오버라이딩.

Abstract Factory: 구체적인 클래스에 의존하지 않고 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공하는 패턴

Singleton: 전역변수를 사용하지 않고 객체를 하나만 생성하도록 하며 생성된 객체를 어디에서든지 참조할 수 있도록 하는 디자인 패턴

-구조

Bridge: 기능의 클래스 계층과 구현의 클래스 계층을 연결하고, 구현부에서 추상 계층을 분리하여 추상화된 부분과 실제 구현 부분을 독립적으로 확장할 수 있는 디자인 패턴

Decorator: 기존에 구현되어 있는 클래스에 필요한 기능을 추가해나가는 설계 패턴

Facade: 복잡한 시스템에 대하여 단순한 인터페이스를 제공함으로써 사용자와 시스템간 결합도를 낮추어 시스템 구조에 대한 파악을 쉽게 하는 패턴

Flyweight: 다수의 객체로 생성될 경우 모두가 갖는 본질적인 요소를 클래스화하여 공유함으로써 메모리를 절약하고 '클래스의 경량화'를 목적으로 하는 디자인 패턴

Proxy: '실제 객체에 대한 대리 객체'로 실제 객체에 대한 접근 이전에 필요한 행동을 취할 수 있게 만들며, 메모리 용량을 아낄 수 있고, 정보은닉의 역할을 수행하는 디자인 패턴

Composite: 객체들의 관계를 트리 구조로 구성하여 부분-전체 계층을 표현하는 패턴. 단일객체와 복합객체를 동일하게 취급

Adapter: 기존에 생성된 클래스를 재상요할 수 있도록 중간에서 맞춰주는 역할을 하는 인터페이스를 만들어주는 패턴

-행위(행위 2020년도)

Mediator: 객체간의 통신을 통제하고 지시할 수 있는 역할을 하는 중재를 두고 중재자에게 모든 것을 요구하며 통신의 빈도수를 줄여 객체지향의 목표를 달성하게 해주는 패턴

Interpreter: 여러 형태의 언어 구문을 해석할 수 있게 만드는 디자인 패턴

Iterator: 내부 구조를 노출하지 않고, 복합개체의 원소를 순차적으로 접근 가능하게하는 패턴

Template Method: 어떤 작업을 처리하는 일부분을 서브 클래스로 캡슐화해 전체 일을 수행하는 구조는 바꾸지 않고 특정 단계에서 수행하는 내역을 바꾸는 패턴

Observer: 한 객체의 상태가 바뀌면 그 객체의 의존하는 다른 객체들에 연락이 가고 자동으로 내용이 갱신되는 패턴

State: 객체 상태를 캡슐화하여 클래스화함으로써 그것을 참조하게 하는 방식. 객체의 상태에 따라 행위 내용을 변경

Visitor: 각 클래스 데이터 구조로부터 처리 기능을 분리하여 별도의 클래스를 만들어놓고 메서드가 각 클래스를 돌아다니며 특정 작업을 수행하도록 만드는 패턴

Command: 실행될 기능을 캡슐화함으로써 주어진 여러 기능을 실행할 수 있는 재사용성이 높은 클래스를 설계하는 패턴

Strategy: 알고리즘군을 정의하고 같은 알고리즘을 각각 하나의 클래스로 캡슐화한 다음, 필요할 때 서로 교환해서 사용할 수 있게 하는 패턴

Memento: 클래스 설계 관점에서 객체의 정보를 저장할 필요가 있을 때 적용하는 패턴

Chain of Responsibility: 정적으로 어떤 기능에 대한 처리의 연결이 하드코딩이 되어있을 때 동적으로 연결되어 있는 경우에 따라 다르게 처리될 수 있도록 연결한 패턴

운영체제: 컴퓨터 시스템이 제공하는 모든 하드웨어, 소프트웨어를 사용할 수 있도록 해주고 컴퓨터 사용자와 하드웨어 간의 인터페이스를 담당하는 프로그램

네트워크: 컴퓨터 장치들의 노드 간 연결을 사용하여 서로에게 데이터를 교환할 수 있도록 하는 기술

OSI 7계층: 응용계층(Application) – 표현계층(Presentation) – 세션계층(Session) – 전송계층(Transport) – 네트워크계층(Network) – 데이터링크계층(Data Link) – 물리계층(Physical)

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

DBMS 현행 시스템 분석시 고려사항: 가용성, 성능, 상호 호환성, 기술 지원, 구축 비용 (가성호기구)

미들웨어: 분산 컴퓨팅 환경에서 응용 프로그램과 프로그램이 운영되는 환경 간에 원만한 통신이 이루어질 수 있도록 제어해주는 소프트웨어

WAS(Web Application Server): 서버계층에서 애플리케이션이 동작할 수 있는 환경을 제공하고 안정적인 트랜잭션 처리와 관리, 이기종 시스템과의 연동을 지원하는 서버

가비지 컬렉션: 프로그램이 동적으로 할당했던 메모리 영역중에서 필요없게 된 영역을 해제하는 기능

1-3. 요구사항 확인

요구사항: 기능적 요구사항, 비기능적 요구사항

요구공학 프로세스: 도출 – 분석 – 명세 – 확인 및 검증 (도문명확)

요구사항 도출 기법: 인터뷰, 브레인스토밍, 델파이 기법, 롤 플레잉, 워크숍, 설문

조사

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

요구사항 명세 단계

비정형 명세 기법: 사용자의 요구를 자연어를 기반으로 서술하는 방법

정형 명세 기법: 사용자의 요구를 수학적인 원리와 표기법으로 서술하는 기법

요구사항 명세 원리 및 검증 항목: 명확성, 완전성, 검증 가능성, 일관성, 수정 용이성, 추적 가능성, 개발 후 이용성 (명확성 일수 추가)

정형 기술 검토: 동료 검토, 워크 스루, 인스펙션 (동워인)

인스펙션: 저작자외의 다른 전문가 또는 팀이 검사하여 오류를 찾아내는 공식적 방법

2. 화면 설계 (중요도: ☆)

2-1. UI 요구사항 확인

UI의 개념: 넓은 의미에서 사용자와 시스템 사이에서의 의사소통할 수 있도록 고안된 물리적, 가상의 매개체

CLI(Command Line Interface): 명령어를 텍스트로 입력하여 조작하는 사용자

인터페이스

GUI(Graphic User Interface): 그래픽 환경을 기반으로 한 마우스를 사용하는 인터페이스

NUI(Natural User Interface): 키보드나 마우스없이 신체 부위를 사용하는 인터페이스

OUI(Organic User Interface): 현실에 존재하는 모든 사물이 입출력장치로 변화할 수 있는 인터페이스

UI 설계 원칙: 직관성, **유효성**, 학습성, 유연성 (직유학유)

직관성(Intuitiveness) : 누구나 쉽게 이해하고, 쉽게 사용할 수 있어야 한다.

유효성(Efficiency) : 정확하고 완벽하게 사용자의 목표가 달성 될 수 있도록 제작한다.

학습성(Learnability) : 모두가 쉽게 배우고 사용할 수 있어야 한다.

유연성(Flexibility) : 사용자의 인터랙션을 최대한 포용하고, 실수를 방지할 수 있도록 제작

UI 품질 요구사항: 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성 (기신사효유이)

스토리보드: UI 화면 설계를 위해서 정책이나 프로세스 및 콘텐츠의 구성, 와어어 프레임(UX, UI), 기능에 대한 정의, 데이터베이스의 연동 등 구축하는 서비스를 위한 대부분 정보가 수록된 문서

2-2. UI 설계

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

UML 구성요소: 사물, 관계, 다이어그램 (사관다)

UML 다이어그램

구조적/정적 다이어그램: 클래스, 객체, 컴포넌트, 배치, 복합체 구조, 패키지 다이어그램 (클래스컴포넌트패키지)

클래스 다이어그램: 객체지향 모델링 시 클래스의 속성 및 연산과 클래스 간 정적인 관계를 표현한 다이어그램

객체 다이어그램: 클래스에 속한 사물들을 특정 시점의 객체와 객체 사이의 관계로 표현한 다이어그램

컴포넌트 다이어그램: 시스템을 구성하는 물리적인 컴포넌트와 그들 사이의 의존

관계를 나타내는 다이어그램

배치 다이어그램: 컴포넌트 사이의 종속성을 표현하고, 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현하는 다이어그램

복합체 구조 다이어그램: 클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현하는 다이어그램

패키지 다이어그램: 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한 다이어그램

행위적/동적 다이어그램: 유스케이스, 시퀀스, 커뮤니케이션, 상태, 활동, 타이밍 다이어그램 (유시커상활타)

유스케이스 다이어그램: 시스템이 제공하고 있는 기능 및 그와 관련된 외부 요소를 사용자의 관점에서 표현하는 다이어그램

시퀀스 다이어그램: 객체 간 동적 상호작용을 시간적 개념을 중심으로 메시지 흐름으로 표현한 다이어그램

커뮤니케이션 다이어그램: 동작에 참여하는 객체들이 주고받는 메시지를 표현하고 메시지뿐만 아니라 객체 간의 연관까지 표현하는 다이어그램

상태 다이어그램: 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호작용에 따라 상태가 어떻게 변화하는 표현하는 다이어그램

활동 다이어그램: 시스템이 어떤 기능을 수행하는지를 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서대로 표현하는 다이어그램

타이밍 다이어그램: 객체 상태 변화와 시간 제약을 명시적으로 표현하는 다이어그램

UI 시나리오 문서의 작성 요건: 완전성, 일관성, 이해성, 가독성, 추적 용이성, 수정 용이성 (완일이가 추수)

3. 데이터 입출력 구현 (중요도: ★)

3-1. 논리 데이터 저장소 확인

데이터 모델: 데이터 모델은 현실세계의 정보를 인간과 컴퓨터가 이해할 수 있도록 추상화하여 표현한 모델

데이터 모델 절차: 개념적, 논리적, 물리적 데이터 모델

개념적 데이터 모델: 현실세계에 대한 이식을 추상적, 개념적으로 표현하여 개념적

구조를 도출하는 데이터 모델

논리적 데이터 모델: 업무의 모습을 모델링 표기법으로 형상화하여 사람이 이해하기 쉽게 표현한 데이터 모델

물리적 데이터 모델: 논리 데이터 모델을 특정 DBMS의 특징 및 성능을 고려하여 물리적인 스키마를 만드는 일련의 데이터 모델

정규화: 관계형 데이터 모델에서 데이터의 중복성을 제거하여 이상현상을 방지하고, 데이터의 일관성과 정확성을 유지하기 위해 무손실 분해를 하는 과정

반 정규화: 정규화된 개체, 속성, 관계에 대해 성능 향상과 개발 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법

논리적 데이터 모델링 종류: 관계, 계층, 네트워크 데이터 모델

일반집합 연산자: 합집합, 교집합, 차집합, 카티션 프로덕트 (합교차카)

순수관계 연산자: 셀렉트, 프로젝트, 조인, 디비전 (셀프조디)

관계 해석: 튜프 관계 해석과 도메인 관계 해석을 하는 비절차적 언어

논리 데이터 모델링 속성: 개체, 속성, 관계 (개속관)

이상 현상: 데이터의 중복성으로 인해 릴레이션을 조작할 때 발생하는 비합리적 현상

삽입 이상: 정보 저장시 정보의 불필요한 세부정보를 입력해야하는 경우

삭제 이상: 정보 삭제시 원치 않는 다른 정보가 같이 삭제되는 경우

갱신 이상: 중복 데이터 중에서 특정 부분만 수정되어 중복된 값이 모순을 일으키는 경우

데이터베이스 정규화 단계 (원부이 결다조)

1정규형: 원자값으로 구성

2정규형: 부분 함수 종속 제거

3정규형: 이행함수 종속 제거

보이스-코드 정규형: 결정자 후보 키가 아닌 함수 종속 제거

4정규형: 다중 값 종속 제거

5정규형: 조인 종속 제거

3-2. 물리 데이터 저장소 설계

참조무결성 제약조건: 릴레이션 사이에 대해 참조의 일관성을 보장하기 위한 조건. 두

개의 릴레이션이 기본키, 외래키를 통해 참조 관계를 형성할 경우, 참조하는 외래키의 값은 항상 참조되는 릴레이션에 기본키로 존재해야 한다.

인덱스: 검색 연산의 최적화를 위해 데이터베이스 내 열에 대한 정보를 구성한 데이터구조

파티션의 종류: 레인지, 해시, 리스트, 컴포지트 파티셔닝 (레해리컴)

레인지 파티셔닝: 연속적인 숫자나 날짜를 기준으로 하는 파티셔닝 기법

해시 파티셔닝: 파티션 키의 해시 함수 값에 의한 파티셔닝 기법

리스트 파티셔닝: 특정 파티션에 저장 될 데이터에 대한 명시적 제어가 가능한 파티셔닝 기법

컴포지트 파티셔닝: 위 3개중 2개 이상의 파티셔닝을 결합하는 파티셔닝 기법

파티션의 장점: 성능 향상, 가용성 향상, 백업 가능, 경합 감소 (성가백합)

3-3. 데이터베이스 기초 활용하기

데이터베이스: 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 수행하는데 필요한 데이터

공용 데이터: 여러 애플리케이션, 시스템들이 공동으로 사용하는 데이터

데이터베이스의 특성: 실시간 접근성, 계속적인 변화, 동시 공용, 내용 참조

데이터베이스의 종류: 파일 시스템, 관계형 DBMS, 계층형 DBMS, 네트워크 DBMS

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

DBMS 유형: 키-값 DBMS, 컬럼 기반 데이터 저장 DBMS, 문서 저장 DBMS, 그래프 DBMS

DBMS 특징: 데이터 무결성, 일관성, 회복성, 보안성, 효율성

빅데이터: 시스템, 서비스, 회사 등에서 주어진 비용, 시간 내에 처리 가능한 데이터 범위를 넘어서는 수십 페타바이트 크기의 비정형 데이터

HDFS: 대용량 데이터의 집합을 처리하는 응용 프로그램에 적합하도록 설계된 하둡 분산 파일 시스템

맵 리듀스: 구글에서 대용량 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 목적으로 제작하여 2004년에 발표한 소프트웨어 프레임워크

NoSQL(Not Only SQL): 데이터 저장에 고정된 테이블 스키마가 필요하지 않고 조인 연상을 사용할 수 있으며 수평적으로 확장이 가능한 DBMS

데이터 마이닝: 대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아내는 기술

데이터 마이닝 주요 기법: 분류 규칙, 연관 규칙, 연속 규칙, 데이터 군집화(분연연데)

4. 통합 구현 (중요도: ★)

4-1. 연계 데이터 구성

연계 요구사항 분석: 서로 다른 두 시스템, 장치, 소프트웨어를 이어주는 중계 역할을 하는 연계 시스템과 관련된 요구사항을 분석하는 과정

중계 서버: 송신 시스템과 수신 시스템 사이에서 데이터를 송수신하고 연계 데이터의 송수신 현황을 모니터링하는 시스템

인터페이스 데이터 공통부: 인터페이스 표준항목을 포함

인터페이스 데이터 개별부: 송수신 시스템에서 업무 처리에 필요한 데이터 포함

인터페이스 데이터 종료부: 전송 데이터의 끝을 표시하는 문자를 포함하여 종료 표시

4-2. 연계 메커니즘 구성

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

연계 방식: 직접 연계, 간접 연계

주요 연계 기술

DB 링크: 데이터베이스에서 제공하는 DB 링크 객체를 이용

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

API: 송신 시스템의 DB에서 데이터를 읽어서 제공하는 애플리케이션 인터페이스 프로그램

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

하이퍼 링크: 현재 페이지에서 다른 부분으로 가거나 다른 페이지로 이동하게 해주는 속성

EAI: 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들간의 정보전달, 연계, 통합을 가능하게 해주는 솔루션

ESB: 웹 서비스가 설명된 WSDL과 SOAP 프로토콜을 이용한 시스템 연계

소켓: 소켓을 생성하여 포트를 할당하고 클라이언트의 요청을 연결하여 통신

4-3. 내외부 연계 모듈 구현

EAI(Enterprise Application Intergration): 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간의 정보를 전달, 연계, 통합이 가능하도록 해주는 솔루션

EAI 구성요소: EAI 플랫폼, 어댑터, 브로커, 메시지 큐, 비즈니스 워크플로우

EAI 플랫폼: 데이터의 신뢰성 있는 전송을 위한 메시지 큐와 트랜잭션 미들웨어 기능 수행

어댑터: 다양한 기업에서 자체적으로 개발한 애플리케이션을 연결하는 EAI의 핵심 장치로 데이터 입출력 도구

브로커: 시스템 상호간 데이터가 전송될 때, 데이터 포맷과 코드를 변환하는 솔루션

메시지 큐: 비동기 메시지를 사용하는 다른 응용프로그램 사이에서 데이터를 송수신하는 기술

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

EAI 구축 유형: 포인트 투 포인트, 허브 앤 스포크, 메시지 버스, 하이브리드 (포허메하)

포인트 투 포인트: 가장 기초적인 애플리케이션 통합방법으로 1:1 단순 통합방법

허브 앤 스포크: 단일한 접점의 허브 시스템을 통하여 데이터를 전송시키는 중앙

집중식 방식

메시지 버스: 애플리케이션 사이 미들웨어를 두어 연계하는 미들웨어 통합 방식

하이브리드: 그룹 내는 허브 앤 스포크 방식을 사용, 그룹 간에는 메시지 버스 방식을 사용하는 통합 방식

ESB(Enterprise Service Bus): 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들은 하나의 시스템으로 운영할 수 있도록 서비스 중심의 통합을 지향하는 아키텍처.

미들웨어를 중심으로 각각 프로토콜이 호환할 수 있도록 애플리케이션의 통합을 느슨한 결합 방식으로 지원하는 방식

웹 서비스: 네트워크에 분산된 정보를 서비스 형태로 개방하여 표준화된 방식으로 공유하는 기술로써 서비스 지향 아키텍처 개념을 실현하는 대표적인 기술

웹 서비스 유형: SOAP, WSDL, UDDI

SOAP(Simple Object Access Protocol): HTTP, HTTPS, SMTP 등을 사용하여 XML 기반의 메시지를 네트워크 상태에서 교환하는 프로토콜

WSDL(Web Service Description Language): 웹 서비스 명, 제공 위치, 메시지 포맷, 프로토콜 정보등 웹 서비스에 대한 상세 정보가 기술된 XML형식으로 구현되어 있는 언어

UDDI(Universal Description, Discovery and Integration): 웹 서비스에 대한 정보인 WSDL을 등록하고 검색하기 위한 저장소로 공개적 접근, 검색이 가능한 레지스트리이자 표준

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

5. 인터페이스 구현 (중요도: ☆)

5-2. 인터페이스 기능 구현

JSON: 키-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷

XML: HTML의 단점을 보완한 인터넷 언어로, SGML의 복잡한 단점을 개선한 특수한 목적을 갖는 마크업 언어이다.

AJAX: 자바스크립트를 사용하여 사용하여 웹 서버와 클라이언트 간 비동기적으로 XML 데이터를 교환하고 조작하기 위한 웹 기술

REST: 웹과 같은 분산 하이퍼미디어 환경에서 자원의 존재/상태 정보를 표준화된 HTTP 메서드로 주고받는 웹 아키텍처

데이터베이스 암호화 알고리즘

대칭 키 암호화 알고리즘: 암,복호화에 같은 암호 키를 쓰는 알고리즘

비대칭 키 암호화 알고리즘: 공개키는 누구나 알 수 있지만, 그에 대응하는 비밀키는 키의 소유자만이 알 수 있도록 공개키와 비밀키를 사용하는 알고리즘

해시 암호화 알고리즘: 해시값으로 원래 입력값을 찾아낼 수 없는 일방향성의 특성을 가진 알고리즘

데이터베이스 암호화 기법

API 방식: 애플리케이션 레벨에서 암호 모듈을 적용하는 애플리케이션 수정 방식

Plug-in 방식: 암,복호화 모듈이 DB서버에 설치된 방식

TDE 방식: DB서버의 DBMS 커널이 자체적으로 암,복호화 기능을 수행하는 방식

Hybrid 방식 API방식과 Plug-in 방식을 결합하는 방식

인터페이스 데이터 암호화 전송을 위한 보안 기술: IPSec, SSL/TLS, S-HTTP

6. 프로그래밍 언어 활용 (중요도: ★★★)

파이썬 `print()`: 기본적으로 개행. `endl()`이 있어야 개행x

파이썬에서 메서드 내부에서 아무것도 실행하지 않을 경우 `pass` 를 써주어야함

명령형 언어: 컴퓨터에 저장된 명령어들이 순차적으로 실행되는 프로그래밍 방식

객체지향 언어: 객체 간의 메시지 통신을 이용하여 프로그래밍 하는 방식

함수형 언어: 수학적 수식과 같은 함수들로 프로그램을 구성하여 호출하는 방식

논리형 언어: 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행하는 개념에 기반한 프로그래밍 방식

라이브러리: 효율적인 프로그램 개발을 위해 필요한 프로그램을 모은 집합체

패키지: 모듈을 디렉토리 형식으로 구조화한 라이브러리

카멜 표기법: 식별자 표기 시에 여러 단어가 이어지면 첫 단어 시작만 소문자, 각 다른 단어의 첫 글자는 대문자로 지정하는 표기법

파스칼 표기법: 식별자 표기시에 여러 단어가 이어지면 각 단어의 첫 글자는 대문자로 지정하는 표기법

스네이크 표기법: 식별자 표기시에 여러 단어가 이어지면 단어 사이에 언더바를 넣는 표기법

헝가리안 표기법: 식별자 표기시에 두어에 자료형을 붙이는 방법

★ 명심해야 할 점

1. 반복문 숫자를 잘 보자
2. 부등호 모양을 잘 보자
3. 파이썬 `print`는 기본적으로 개행이다

7. SQL 응용 (중요도: ★★★)

7-1. 데이터베이스 기본

트랜잭션: 인가받지 않은 사용자로부터 데이터를 보장하기 위해 DBMS가 가져야 하는 특성이자, 하나의 논리적 기능을 정삭적으로 수행하기 위한 작업의 기본단위

트랜잭션의 특성: 원자성, 일관성, 격리성, 영속성 (ACID)

원자성(Atomicity): 하나라도 실패할 경우 전체가 취소되어야하는 특성

일관성(Consistency): 트랜잭션이 실행 성공 후 항상 일관된 데이터베이스 상태를 보존해야하는 특성

격리성(Isolation): 트랜잭션 실행 중 생성하는 연산의 중간 결과를 다른 트랜잭션이 접근 불가능한 특성

영속성(Durability): 성공이 완료된 트랜잭션의 결과는 영속적으로 데이터베이스에 저장하는 특성

트랜잭션의 상태: 활동, 부분완료, 완료, 실패, 철회 (활부완실철)

트랜잭션 제어: 커밋, 롤백, 체크포인트

병행 제어: 다수 사용자 환경에서 여러 트랜잭션을 수행할 때, 데이터베이스 일관성 유지를 위해 상호작용을 제어하는 기법

병행 제어 기법의 종류: 로킹, 낙관적 검증, 타임 스탬프 순서, 다중버전 동시성 제어

로킹: 다중 트랜잭션 환경에서 DB의 일관성과 무결성을 유지하기 위해 트랜잭션의 순차적 진행을 보장하는 직렬화 기법

낙관적 검증: 트랜잭션이 검증을 수행하지 않고 실행되고 트랜잭션 종료 시 검증을 수행하여 데이터베이스에 반영하는 기법

타임스탬프 순서: 트랜잭션이 읽거나 갱신한 데이터에 대해 트랜잭션이 실행되기 전에 타임 스팸프를 주여하여 부여된 시간에 따라 트랜잭션 작업을 수행하는 기법

다중버전 동시성 제어: 트랜잭션의 타임스탬프와 접근하려는 데이터의 타임스탬프를 비교하여 직렬가능성이 보장되는 적절한 버전을 선택하여 접근하도록 하는 기법

회복 기법: 트랜잭션을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구시키는 작업

회복 기법 종류: 로그 기반 회복 기법, 체크 포인트 회복 기법, 그림자 페이징 회복 기법 (회로체크)

로그 기반 회복 기법: 지연 갱신 회복 기법 – 트랜잭션이 완료되기 전까지 DB에 기록하지 않는 기법 / 즉각 갱신 회복 기법 – 트랜잭션 갱신 결과를 바로 DB에 반영하는 기법

체크 포인트 회복 기법: 장애 발생시 검사점 이후에 처리된 트랜잭션에 대해서만 장애

발생 이전의 상태로 복원시키는 회복 기법

그림자 페이징 회복 기법: 데이터베이스 트랜잭션 수행 시 복제본을 생성하여 데이터베이스 장애 시 이를 이용해 복구하는 기법

데이터 정의어 대상: 도메인, 스키마, 테이블, 뷰, 인덱스

도메인: 하나의 속성이 가질 수 있는 원자값들의 집합

스키마: 데이터베이스의 구조, 제약조건 등의 정보를 담고있는 기본적인 구조 (외부 스키마, 개념 스키마, 내부 스키마)

테이블: 데이터 저장 공간

뷰: 하나 이상의 물리 테이블에서 유도되는 가상의 테이블

인덱스: 검색을 빠르게 하기 위한 구조

인덱스 종류: 순서, 해시, 비트맵, 함수기반, 단일, 결합, 클러스터드 인덱스 (순회비함 단결클)

순서 인덱스: 데이터가 정렬된 순서로 생성되는 인덱스

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

비트맵 인덱스: 각 컬럼에 적은 개수 값이 저장된 경우 선택하는 인덱스

함수기반 인덱스: 수식이나 함수를 적용하여 만든 인덱스

단일 인덱스: 하나의 컬럼으로만 구성한 인덱스

결합 인덱스: 두 개이상의 컬럼으로 구성한 인덱스

클러스터드 인덱스: 기본 키를 기준으로 레코드를 묶어서 저장하는 인덱스

인덱스의 스캔 방식: 범위 스캔, 전체 스캔, 단일 스캔, 생략 스캔

인덱스 범위 스캔: 인덱스 루트 블록에서 리프블록까지 수직적으로 탐색한 후에 리프 블록을 필요한 범위만 스캔하는 방식

인덱스 전체 스캔: 수직적 탐색 없이 인덱스 리프 블록을 처음부터 끝까지 수평적으로 탐색하는 방식

인덱스 단일 스캔: 수직적 탐색만으로 데이터를 찾는 스캔 방식

인덱스 생략 스캔: 선두 컬럼이 조건 절에 빠졌어도 인덱스를 활용하는 스캔 방식

CREATE TABLE, DROP TABLE, TRUNCATE TABLE

ALTER TABLE 테이블명 ADD/MODIFY/DROP (테이블 수정)

INSERT INTO 테이블명(속성명1, ...) VALUES(데이터1, ...);

UPDATE 테이블명 SET 속성명=데이터 WHERE 조건;

DELETE FROM 테이블명 WHERE 조건;

* 쿼리문 작성시 끝에 ; 꼭 써주기

7-2. 응용 **SQL** 작성하기

데이터 분석 함수의 종류: 집계 함수, 그룹 함수, 윈도 함수

집계 함수: 여러 행 또는 테이블 전체 행으로부터 하나의 결괏값을 반환하는 함수

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

윈도 함수: 데이터베이스를 사용한 온라인 분석 처리 용도로 사용하기 위한 함수

그룹 함수 종류: ROLLUP, CUBE, GROUPING SETS

윈도 함수 종류: 순위 함수, 행 순서 함수, 그룹 내 비율 함수 (순행비)

순위 함수: RANK, DENSE_RANK, ROW_NUMBER

RANK: 동일 순위시 후순위 넘어감

DENSE_RANK: 동일 순위시 후순위 넘어가지 않음

ROW_NUMBER: 동일 값이 존재해도 무관하게 연속 번호 부여

행 순서 함수 종류: FIRST_VALUE, LAST_VALUE, LAG, LEAD

FIRST_VALUE: 파티션별 윈도에서 가장 먼저 나오는 값

LASE_VALUE: 파티션별 윈도에서 가장 늦게 나오는 값

LAG: 파티션 별 윈도에서 이전 로우의 값 반환

LEAD: 파티션 별 윈도에서 이후 로우의 값 반환

7-3. 절차형 **SQL** 활용하기

절차형 SQL: 일반적인 개발 언어처럼 SQL 언어에서도 절차 지향적인 프로그램이 가능하도록 하는 트랜잭션 언어

절차형 SQL 종류: 프로시저, 사용자 정의함수, 트리거

프로시저: 일련의 쿼리들을 하나의 함수처럼 실행하기 위한 쿼리의 집합

사용자 정의함수: SQL 처리를 수행 후 수행 결과를 단일 값으로 반환할 수 있는

절차형 SQL

트리거: DB시스템에서 삽입, 삭제, 갱신등의 이벤트가 발생할 때 관련 작업이 자동적으로 수행되는 절차형 SQL

트리거의 종류: 행 트리거, 문장 트리거

행 트리거: 데이터의 변화가 생길 때마다 실행

문장 트리거: 트리거에 의해 단 한 번 실행

트리거 구성: 선언부, 이벤트부, 시작/종료부, 제어부, SQL, 예외부

선언부(DECLARE): 트리거의 명칭을 정의하는 부분

이벤트부(EVENT): 트리거가 실행되는 타이밍, 이벤트를 명시하는 부분

시작/종료부(BEGIN/END): 트리거의 시작과 종료를 표현하는데 필수적

제어부(CONTROL): 비교 조건에 따라 블록 또는 문장을 실행

SQL: DML을 주로 사용

예외부: BEGIN~END 절에서 SQL문이 실행될 때 예외 발생 시 예외 처리 방법을 정의

7-4. 데이터 조작 프로시저 최적화

쿼리 성능 개선: 데이터베이스에서 프로시저에 있는 SQL 실행 계획을 분석, 수정을 통해 최소의 시간으로 원하는 결과를 얻도록 프로시저를 수정하는 작업

옵티マイ저: SQL을 가장 빠르고 효율적으로 수행할 최적의 처리경로를 생성해주는 DBMS 내부의 핵심 엔진

규칙기반 옵티マイ저: 통계 정보가 없는 상태에서 사전 등록된 규칙에 따라 질의 실행 계획을 선택하는 옵티マイ저

비용기반 옵티マイ저: 통계 정보로부터 모든 접근 경로를 고려한 질의 실행 계획을 선택하는 옵티マイ저

8. 서버 프로그램 구현 (중요도: ★★)

8-1. 개발환경 구축

개발 도구의 분류: 빌드 도구, 구현 도구, 테스트 도구, 형상 관리 도구 (빌구테형)

빌드 도구: 작성한 코드의 빌드 및 배포를 수행하는 도구

구현 도구: 개발자의 코드 작성과 디버깅, 수정 등과 같은 작업을 지원하는 도구

테스트 도구: 코드의 기능 검증과 전체의 품질을 높이기 위해 사용하는 도구

형상관리 도구: 개발자들이 작성한 코드와 리소스등 산출물에 대한 버전 관리를 위한 도구

서버 하드웨어 개발환경: 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버

웹 서버: HTTP를 이용한 요청/응답을 처리, 웹 상의 정적 콘텐츠를 처리

웹 애플리케이션 서버: 동적 콘텐츠(JSP)를 처리하기 위해 사용

데이터베이스 서버: 데이터의 수집, 저장을 위한 용도로 사용

파일 서버: 파일 저장 하드웨어로 물리 저장장치를 활용한 서버

미들웨어: 컴퓨터와 컴퓨터 간의 연결을 쉽고 안전하게 할 수 있도록 해주고 이에 대한 관리를 도와주는 소프트웨어. 자바 기반 환경에서 JVM을 설치하여 컨테이너로 주로 이용

형상 관리: 소프트웨어 개발을 위한 전체 과정에서 발생하는 모든 항목의 변경 사항을 관리하기 위한 활동

형상 관리 절차: 형상 식별, 형상 통제, 형상 감사, 형상 기록

형상 식별: 형상 관리 대상을 정의 및 식별하는 활동

형상 통제: 항목의 변경요구 관리, 변경제어

형상 감사: 소프트웨어 베이스라인의 무결성 평가

형상 기록: 소프트웨어 형상 및 변경관리에 대한 각종 수행결과를 기록

소프트웨어 형상 관리 도구 유형: 공유 폴더 방식, 클라이언트/서버 방식, 분산 저장소 방식

공유 폴더 방식: 매일 개발이 완료된 파일은 약속된 위치의 공유 폴더에 복사하는 방식

클라이언트/서버 방식(CVS, SVN): 중앙에 버전 관리 시스템을 항상 동작시키는 방식

분산 저장소 방식(Git): 로컬 저장소와 원격 저장소로 분리되어 분산 저장하는 방식

CVS(Concurrent Versions System): 서버와 클라이언트로 구성되어있고, 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): CVS와 달리 소스 파일의 수정을 한 사람으로 제한하여 파일 잠금 방식으로 형상을 관리하는 도구

BitKeeper: SVN과 비슷한 중앙 통제 방식으로 대규모 프로젝트에서 빠른 속도를 내도록 개발된 형상 관리 도구

Git: 분산형 버전 관리 시스템이며, 대형 프로젝트에서 효과적이고 유용

Clear Case: 복수 서버, 복수 클라이언트 구조이며 서버가 부족할 때 필요한 서버를 하나씩 추가하여 확장성을 기할 수 있음

8-2. 공통 모듈 구현

모듈: 그 자체로 하나의 완전한 기능을 수행할 수 있는 독립된 실체

모듈화 기법: 루틴, 메인 루틴, 서브 루틴

루틴: 소프트웨어에서 특정 동작을 수행하는 일련의 코드로 기능을 가진 명령들의 모임

메인 루틴: 프로그램의 주요한 부분이며, 전체의 대략적인 동작 절차를 표시하도록 만들어진 루틴

서브 루틴: 메인 루틴에 의해 필요할 때마다 호출되는 루틴

공통 모듈: 전체 프로그램의 기능 중 특정 기능을 처리할 수 있는 실행 코드. 자체적으로 컴파일이 가능하고 다른 프로그램에서 재사용이 가능

응집도: 모듈의 독립성을 나타내는 정도로, 모듈 내부 구성요소간 연관 정도

응집도의 유형: 우연적, 논리적, 시간적, 절차적, 통신적, 순차적, 기능적 (우논시절통순기)

우연적 응집도 (응집도 낮음, 나쁜 품질) ~ 기능적 응집도 (응집도 높음, 좋은 품질)

우연적 응집도: 모듈 내부의 각 구성요소가 연관이 없을 경우의 응집도

논리적 응집도: 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우의 응집도

시간적 응집도: 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우의 응집도

절차적 응집도: 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성요소들이 그 기능을 순차적으로 수행 할 경우의 응집도

통신적 응집도: 동일한 입력과 출력을 사용하여 다른 기능을 수행하는 활동들이 모여있을 경우의 응집도

순차적 응집도: 모듈 내에서 한 활동으로부터 나온 출력값을 다른 활동이 사용할 경우의 응집도

기능적 응집도: 모듈 내부의 모든 기능이 단일한 목적을 위해 수행되는 경우의 응집도

결합도: 모듈 내부가 아닌 외부의 모듈과의 연관도 또는 모듈 간의 상호의존성

결합도의 유형: 내용, 공통, 외부, 제어, 스템프, 자료 결합도

내용 결합도(결합도 높음, 낮은 품질) ~ 자료 결합도(결합도 낮음, 좋은 품질)

내용 결합도: 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우의

결합도

공통 결합도: 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 전역 변수를 갱신하는 식으로 상호작용하는 경우의 결합도

외부 결합도: 두 개의 모듈이 외부에서 도입된 데이터 포맷, 통신 프로토콜, 또는 디바이스 인터페이스를 공유할 경우의 결합도

제어 결합도: 단순 처리할 대상인 값만 전달되는 것이 아닌 어떻게 처리를 해야 한다는 제어 요소가 전달되는 경우의 결합도

스탬프 결합도: 모듈 간의 인터페이스로 배열이나 객체, 구조 등이 전달되는 경우의 결합도

자료 결합도: 모듈 간의 인터페이스로 전달되는 파라미터를 통해서만 모듈 간의 상호 작용이 일어나는 경우의 결합도

공통 모듈 테스트의 종류: 화이트박스, 메서드 기반, 화면 기반 테스트, 테스트 드라이버, 스텁

화이트박스 테스트: 응용 프로그램 내부 구조와 동작을 검사하는 SW 테스트 방식

메서드 기반 테스트: 공통 모듈의 외부에 공개된 메서드 기반의 테스트

화면 기반 테스트: 각각의 화면단위로 단위모듈을 개발 후 화면에 직접 데이터를 입력

테스트 드라이버: 하위 모듈은 있지만 상위 모듈이 없는 경우 사용

테스트 스텁: 상위 모듈은 있지만 하위 모듈이 없는 경우 사용

(상드 하스)

8-3. 서버 프로그램 구현

백엔드: 사용자와 만나지 않고 프론트엔드와 연동하여 핵심 로직을 처리하는 영역

프론트엔드: 사용자의 화면에 나타나는 웹 화면 영역으로 웹 페이지를 그리는 기술을 활용

DAO(Data Access Object): 특정 타입의 데이터베이스에 추상 인터페이스를 제공하는 객체로 세부내용 노출 없이 데이터 조작

DTO(Data Transfer Object): 프로세스 사이에서 데이터를 전송하는 객체로 데이터 저장, 회수 외에 다른 기능 없음

VO(Value Object): 간단한 엔티티를 의미하는 작은 객체 가변 클래스인 DTO와 달리 고정 클래스를 가짐

8-4. 배치 프로그램 구현

배치 프로그램: 사용자와의 상호작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법

배치 프로그램의 유형: 이벤트, 온디맨드, 정기 배치 (이온정)

이벤트 배치: 사전에 정의해 둔 조건 충족 시 자동으로 실행

온디맨드 배치: 사용자의 명시적 요구가 있을 때마다 실행

정기 배치: 정해진 시점에 정기적으로 실행

배치 스케줄러: 일괄 처리를 위해 주기적으로 발생하는 작업을 지원하는 도구

배치 스케줄러의 종류: 스프링 배치, 쿼츠 스케줄러

스프링 배치: 스프링 프레임워크의 3대 요소를 모두 사용할 수 있는 대용량 처리를 제공하는 스케줄러 배치 애플리케이션

쿼츠 스케줄러: 스프링 프레임워크에 플러그인되어 수행하는 작업과 실행 스케줄을 정의하는 트리거를 분리하여 유연성을 제공하는 오픈 소스 기반 스케줄러

9. 소프트웨어 개발 보안 구축 (중요도: ★★)

9-1. 소프트웨어 개발 보안 설계

소프트웨어 개발 보안의 3대 요소: 기밀성, 무결성, 가용성 (기무가)

기밀성: 인가되지 않은 개인 혹은 시스템 접근에 따른 정보 공개 및 노출을 차단하는 특성

무결성: 정당한 방법을 따르지 않고서는 데이터가 변경될 수 없으며, 데이터의 정확성 및 안정성과 악의로 변경되거나 훼손되지 않음을 보장하는 특성

가용성: 권한을 가진 사용자나 애플리케이션이 원하는 서비스를 지속해서 사용할 수 있도록 보장하는 특성

소프트웨어 개발 보안 용어: 자산, 위협, 취약점, 위험 (자위취위)

자산: 조직의 데이터 또는 조직의 소유자가 가치를 부여한 대상

위협: 조직의 자산에 악영향을 끼칠 수 있는 사건이나 행위

취약점: 위협이 발생하기 위한 사전 조건으로 시스템의 정보 보증을 낮추는 데 사용되는 약점

위험: 위협이 취약점을 이용하여 조직의 자산 손실 피해를 가져올 가능성

DoS(Denial of Service): 시스템을 악의적으로 공격해서 해당 시스템의 자원을 부족하게 하여 원래 의도된 용도로 사용하지 못하게 하는 공격

DoS 공격의 종류: SYN 플러딩, UDP 플러딩, 스머프, 죽음의 핑, 랜드 어택, 티어드롭, 봉크, 보잉크

SYN 플러딩(SYN Flooding): 서버의 동시 가용 사용자 수를 SYN 패킷만 보내 점유하여 다른 사용자가 서버를 사용 불가능하게 하는 공격

UDP 플러딩(UDP Flooding): 대량의 UDP 패킷을 만들어 임의의 포트로 전송하여 응답 메시지를 생성하게 하여 지속해서 자원을 고갈 시키는 공격

스머프/스머핑(Smurf/Smurfing): 출발지 주소를 공격 대상의 IP로 설정하여 네트워크 전체에게 ICMP Echo 패킷을 직접 브로드캐스팅하여 마비시키는 공격

죽음의 핑(Ping of Death): ICMP 패킷을 정상적인 크기보다 크게 만들어 전송하여

다수의 IP 단편화가 발생하고, 단편화된 패킷을 처리하는 과정에서 많은 부하가 발생하거나 버퍼의 오버플로우가 발생하여 정상적인 서비스를 하지 못하도록 하는 공격기법

랜드 어택(Land Attach): 출발지 IP와 목적지 IP를 같은 패킷 주소로 만들어 보냄으로써 수신자가 자기 자신에게 응답을 보내게 하여 시스템의 가용성을 침해하는 공격 기법

티어 드롭(Tear Drop): IP 패킷의 재조합 과정에서 잘못된 Fragment Offset 정보로 인해 수신시스템이 문제를 발생하도록 만드는 DoS 공격

봉크(Bonk)/보잉크(Boink): 프로토콜의 오류 제어를 이용한 공격기법으로서 시스템의 패킷 재전송과 재조립이 과부하를 유발. 봉크: 같은 시퀀스 번호를 계속 보냄/보잉크: 일정한 간격으로 시퀀스 번호에 빈 공간 생성

DDoS(Distributed DoS): DoS의 또 다른 형태로 여러 대의 공격자를 분산 배치하여 동시에 동작하게 함으로써 특정 사이트를 공격하는 기법

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

애플리케이션 공격기법

HTTP GET 플러딩(Flooding): 과도한 Get 메시지를 이용하여 웹 서버의 과부하를 유발시킴

Slowloris: HTTP GET 메서드를 사용하여 헤더의 최종 끝을 알리는 개행 문자열을 다르게 전송하여 대상 웹 서버와 연결상태를 장기간 지속시키고 연결 자원을 모두 소진시키는 공격

RUDY: 요청 헤더의 Content-Length를 비정상적으로 크게 설정하여 메시지 바디 부분을 매우 소량으로 보내 계속 연결상태를 유지시키는 공격

Slow HTTP Read DoS: TCP 윈도 크기와 데이터 처리율을 감소시킨 상태에서 다수 HTTP 패킷을 지속적으로 전송하여 대상 웹 서버의 연결상태가 장시간 지속, 연결자원을 소진시킴

Hulk DoS: 공격자가 공격대상 웹 사이트의 주소(URL)을 지속적으로 변경하면서 다량으로 GET 요청을 발생시키는 서비스 거부 공격

Hash DoS: 조작된 많은 수의 파라미터를 POST 방식으로 웹 서버로 전달하여 다수의 해시 충돌을 발생시켜서 자원을 소모시키는 공격

네트워크 공격기법

스니핑(Sniffing): 공격자에게 직접 공격을 하지 않고 데이터만 몰래 들여다보는 수동적 공격

네트워크 스캐너(Scanner)/스니퍼(Sniffer): 네트워크 하드웨어 및 소프트웨어 구성의 취약점 파악을 위해 공격자가 취약점을 탐색하는 공격 도구

패스워드 크래킹>Password Cracking: 사전 트래킹 공격, 무차별 크래킹 공격, 패스워드 하이브리드 공격, 레인보우 테이블 공격 활용

IP 스폐핑(IP Spoofing): 침입자가 인증된 컴퓨팅 시스템인 것처럼 속여서 타깃 시스템의 정보를 빼내기 위해서 본인의 패킷 헤더를 인증된 호스트의 IP 주소로 위조하여 타깃에 전송

ARP 스폐핑(ARP Spoofing): 공격자가 특정 호스트의 MAC 주소를 자신의 MAC 주소로 위조한 ARP Reply를 만들어 희생자에게 지속적으로 전송하여 희생자의 ARP 캐시 테이블에 특정 호스트의 MAC 정보를 공격자의 MAC 정보로 변경, 희생자로부터 특정 호스트로 나가는 패킷을 공격자가 스니핑 하는 공격

ICMP Redirect 공격: 3계층에서 스니핑 시스템을 네트워크에 존재하는 또 다른 라우터라고 알림으로써 패킷의 흐름을 바꾸는 공격

트로이 목마: 악성 루틴이 숨어 있는 프로그램으로 실행하면 악성 코드를 실행하는 프로그램

버퍼 오버플로우 공격: 메모리에 할당된 버퍼 크기를 초과하는 양의 데이터를 입력하여 이로 인해 프로세스의 흐름을 변경시켜서 악성 코드를 실행시키는 공격기법

버퍼 오버플로우 공격 유형: 스택 버퍼, 힙 버퍼 오버플로우 공격

스택 버퍼 오버플로우 공격: 스택 영역에 할당된 버퍼 크기를 초과하는 양의 데이터를 입력하여 복귀 주소를 변경하고 공격자가 원하는 임의의 코드를 실행하는 공격기법

힙 버퍼 오버플로우 공격: 프로그램 실행 시 동적으로 할당되는 힙 영역에 할당된 버퍼 크기를 초과하는 데이터를 입력하여 메모리의 데이터 등을 변경, 공격자가 원하는 임의의 코드를 실행하는 공격 기법

버퍼 오버플로우 공격 대응 방안

스택가드 활용: 카나리라고 불리는 무결성 체크용 값을 복귀 주소와 변수 사이에 삽입해두고, 버퍼 오버플로우 발생 시 카나리 값을 체크, 변할 경우 복귀 주소를 호출하지 않는 방식

스택쉴드: 함수 시작 시 복귀 주소를 Global RET이라는 특수 스택에 저장해두고, 함수 종료 시 저장된 값과 스택의 RET 값을 비교해서 다를 경우 오버플로우로 간주하고 실행을 중단

ASLR(Address Space Layout Randomization): 메모리 공격을 방어하기 위해 주소 공간 배치를 난수화하고, 실행 시마다 메모리 주소를 변경시켜 버퍼 오버플로우를

통한 특정 주소 호출을 차단

주요 시스템 보안 공격기법

포맷 스트링 공격(Format String Attack): 포맷 스트링을 인자로 하는 함수의 취약점을 이용한 공격

레이스 컨디션 공격(Race Condition Attack): 실행되는 프로세스가 임시파일을 만드는 경우 악의적 프로그램을 통적 실행 중에 끼어 들어 임시파일을 심볼릭 링크하여 악의적인 행위를 수행하게 하는 공격

키로거 공격(Key Logger Attack): 사용자의 키보드 움직임을 탐지해서 저장하고 개인정보를 빼가는 해킹 공격

루트킷(Rootkit): 시스템 침입 후 침입 사실을 숨기고 차후의 침입을 위한 백도어, 원격 접근, 관리자 권한 획득 등 불법적인 해킹에 사용되는 기능을 제공하는 프로그램의 모음

보안 관련 용어

스피어피싱(Spear Phishing): 일반적인 이메일로 위장한 메일을 지속적으로 발송하여 본문 링크나 첨부된 파일을 클릭시 사용자의 개인정보를 탈취하는 공격

스미싱(Smishing): 문자메시지를 이용하여 개인정보를 요구, 소액 결제를 유도하는 피싱공격

큐싱(Qshing): QT 코드를 통해 악성 앱을 내려받도록 유도하여 금융 정보등을 빼내는 공격

봇넷(Botnet): 악성 프로그램에 감염되어 다수의 컴퓨터들이 네트워크로 연결된 형태

APT 공격(Advanced Persistent Threat): 특수목적의 조직이 하나의 표적에 대해 다양한 IT 기술을 이용하여 정보를 수집, 취약점 분석을 하여 피해를 주는 공격

공급망 공격(Supply Chain Attack): 소프트웨어 개발사의 네트워크에 침투하여 악의적인 코드를 삽입하거나 배포 서버에 접근하여 악의적인 파일을 변경하는 방식을 통해 사용자가 소프트웨어를 설치, 업데이트 시에 자동적으로 감염되게 하는 공격

제로데이 공격(Zero Day Attack): 보안 취약점이 발견되어 널리 공표되기 전에 해당 취약점을 악용하여 이루어지는 보안 공격

웜(Worm): 스스로를 복제하여 네트워크 등의 연결을 통하여 전파하는 악성 프로그램

악성 봇(Malicious Bot): 스스로 실행되지 못하고 해커의 명령에 의해 원격에서 제어 또는 실행이 가능한 프로그램 혹은 코드

사이버 킬체인(Cyber Kill Chain): 공격형 방위 시스템으로 지능적, 지속적 사이버공격에 대해 7단계 프로세스별 공격분석 및 대응을 체계화한 APT 공격 방어

분석 모델

랜섬웨어(Ransomware): 파일들을 암호화하여 암호화된 파일의 몸값을 요구하는 소프트웨어

이블 트윈 공격(Evil Twin): 무선 Wifi 피싱 기법으로 핫스팟에 연결한 무선 사용자들의 정보를 탈취하는 공격

난독화(Obfuscation): 코드의 가독성을 낮춰 역공학에 대한 대비를 하기 위해서 프로그램 코드의 일부 또는 전체를 변경하는 기법

Tripwire: 크래커가 침입하여 시스템에 백도어를 만들어 놓거나 설정 파일을 변경해 놓았을 때 이러한 사실을 알 수 있게 분석하는 도구

Ping: 인터넷으로 접속하려는 원격 호스트가 정상적으로 운영되고 있는지를 확인하는 진단 목적으로 사용하는 명령어

Tcpdump: 네트워크 인터페이스를 거치는 패킷의 내용을 출력해주는 프로그램

인증 기술의 유형: 지식기반, 소지기반, 생체기반, 특징기반 (지소생특)

서버 접근 통제 유형: DAC, MAC, RBAC

임의적 접근 통제(DAC; Discretionary Access Control): 시스템에 대한 접근을 사용자/그룹의 신분 기반으로 제한하는 방법

강제적 접근 통제(MAC; Mandatory Access Control): 시스템 정보의 허용등급을 기준으로 사용자가 갖는 접근 허가 권한에 근거하여 시스템에 대한 접근을 제한하는 방법

역할기반 접근 통제(RBAC; Role Based Access Control): 중앙 관리자가 사용자와 시스템의 상호관계를 통제하며 조직 내 맡은 역할에 기초하여 자원에 대한 접근을 제한하는 방법

벨-라파둘라 모델: 미 국방부 지원 보안 모델로 보안 요소 중 기밀성을 강조하며 강제적 정책에 의해 접근 통제하는 모델 (No Read Up, No Write Down)

비바 모델: 벨-라파둘라 모델의 단점을 보완한 무결성을 보장하는 최초의 모델 (No Read Down, No Write Up) (벨기비무)

대칭 키 암호 방식: 암호화와 복호화에 같은 암호 키를 쓰는 알고리즘 (블록 암호 방식, 스트림 암호 방식)

비대칭 키 암호 방식: 사전에 개인 키를 나눠 가지지 않은 사용자들이 안전하게 통신하는 방식. 공개키는 누구나 알 수 있지만, 비밀키는 키의 소유자만이 알 수 있어야 함

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

일방향 암호 종류: MAC, MDC

MAC(Message Authentication Code): 키를 사용하는 메시지 인증 코드

MDC(Modification Detection Code): 키를 사용하지 않는 변경 감지 코드

대칭키 암호화 알고리즘

DES(Data Encryption Standard): 1975년 미국 연방 표준국(NIST)에서 발표한 대칭 키 기반의 블록 암호화 알고리즘. 블록 크기는 64bit, 페이스텔 구조

SEED: 1999년 국내 한국인터넷진흥원(KISA)이 개발한 블록 암호화 알고리즘

AES(Advanced Encryption Standard): 2001년 NIST에서 발표한 블록 암호화 알고리즘

ARIA(Academy, Research Institute, Agency): 2004년 국가정보원과 산학연구협회가 개발한 블록 암호화 알고리즘. 대부분의 연산은 XOR과 같은 단순한 바이트 단위 연산으로 구성

IDEA(International Data Encryption Algorithm): DES를 대체하기 위해 스위스 연방기술 기관에서 개발한 블록 암호화 알고리즘

LFSR(Linear Feedback Shift Register): 시프트 레지스터의 일종으로, 레지스터에 입력되는 값이 이전 상태 값들의 선형 함수로 계산되는 구조로 되어 있는 스트림 암호화 알고리즘

비대칭키 암호화 알고리즘

디피-헬만: 최초의 공개키 알고리즘으로 디피와 헬만이 1976년에 고안한 알고리즘. 유한 필드 내에서 이산대수의 계산이 어려운 문제를 기본원리로 하고 있음

RSA: 1977년 3명의 MIT 수학 교수가 고안한 큰 인수의 곱을 소인수 분해하는 수학적 알고리즘을 이용하는 공개키 암호화 알고리즘

ECC: 1985년 코블리치와 밀러가 RSA 암호 방식에 대한 대안으로 처음 제안. 타원 곡선 암호는 유한체 위에서 정의된 타원곡선 군에서의 이산대수 문제에 기초한 알고리즘

ElGamal: T.Gamal이 1984년에 제안한 공개키 알고리즘. 이산대수의 계산이 어려운 문제를 기본 원리로 하고 있고 전자서명과 데이터 암, 복호화에 함께 사용 가능

해시 암호화 알고리즘

MD5: 1991년 R.rivest가 MD4를 개선한 암호화 알고리즘으로 프로그램이나 파일의 무결성 검사에 사용

SHA-1(Secure Hash Algorithm): 1993년 NSA에서 미 정부 표준으로 지정되었고, DSA에서 사용. 160비트의 해시값을 생성하는 해시 알고리즘

SHA-256/384/512: 256비트의 해시값을 생성하는 해시 함수

HAS-160: 국내 표준 서명 알고리즘을 위하여 개발된 해시함수. MD5와 SHA1의 장점을 취하여 개발된 해시 알고리즘

IPSec(Internet Protocol Security): IP 계층(3계층)에서 무결성과 인증을 보장하는 인증 헤더와 기밀성을 보장하는 암호화를 이용한 IP 보안 프로토콜

SSL(Secure Socket Layer)/TLS(Transport Layer Security): 전송계층(4계층)과 응용계층(7계층) 사이에서 클라이언트와 서버 간의 웹 데이터 암호화, 상호 인증 및 전송 시 데이터 무결성을 보장하는 보안 프로토콜

S-HTTP(Secure Hypertext Transfer Protocol): 웹 상에서 네트워크 트래픽을 암호화하는 주요 방법 중 하나로 클라이언트와 서버 간에 전송되는 모든 메시지를 각각 암호화하여 전송하는 기술

9-2. 소프트웨어 개발 보안 구현

시큐어 코딩 가이드: 설계 및 구현 단계에서 해킹 등의 공격을 유발할 가능성이 있는 잠재적인 보안 취약점을 사전에 제거하고, 외부 공격으로부터 안전한 SW를 개발하는 기법

시큐어 코딩 가이드 항목: 입력데이터 검증 및 표현, 보안 기능, 시간 및 상태, 에러 처리, 코드 오류, 캡슐화 API 오용 (입보시 예코캡아)

입력 데이터 검증 및 표현 취약점

XSS(Cross Site Script): 검증되지 않은 외부 입력 데이터가 포함된 웹페이지가 전송되는 경우, 사용자가 열람함으로써 웹페이지에 포함된 부적절한 스크립트가 실행되는 공격

사이트 간 요청 위조(CSRF; Cross-Site Request Forgery): 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 공격

SQL 삽입(Injection): 응용 프로그램의 보안 취약점을 이용해서 악의적인 SQL 구문을 삽입, 실행 시켜서 데이터베이스의 접근을 통해 정보를 탈취, 조작하는 공격

Stored XSS: 웹 페이지 읽기 / **Reflected XSS:** 이메일 URL 클릭 / **DOM XSS:** DOM기반 취약점이 있는 브라우저

네트워크 보안 솔루션

방화벽(Firewall): 기업 내부, 외부 간 트래픽을 모니터링하여 시스템의 접근을 허용하거나 차단하는 시스템

웹 방화벽(WAF, Web Application Firewall): 일반적인 네트워크 방화벽과는 달리 웹 애플리케이션 보안에 특화된 보안장비

네트워크 접근 제어(NAC; Network Access Control): 단말기가 내부 네트워크 접속을 시도할 때 이를 제어하고 통제하는 기능을 제공하는 솔루션

침입 탐지 시스템(IDS; Intrusion Detection System): 네트워크에서 발생하는 이벤트를 모니터링하고 비인가 사용자에 의한 자원접근과 보안정책 위반행위를 실시간으로 탐지하는 시스템

침입 방지 시스템(IPS; Intrusion Prevention System): 네트워크에 대한 공격을 실시간으로 차단하고, 유해트래픽에 대한 조치를 능동적으로 처리하는 시스템

무선 침입 방지 시스템(WIPS; Wireless Intrusion Prevention System): 인가되지 않은 무선 단말기의 접속을 자동적으로 탐지, 차단하고 보안에 취약한 무선공유기를 탐지하는 시스템

통합 보안 시스템(UTM; Unified Threat Management): 방화벽, IDS, IPS등 다양한 보안 장비의 기능을 하나의 장비로 통합하여 제공하는 시스템

가상사설망(VPN; Virtual Private Network): 인터넷과 같은 공중망에 인증, 암호화, 터널링 기술을 활용하여 전용망을 사용하는 효과를 가진 보안 솔루션

시스템 보안 솔루션

스팸 차단 솔루션(Anti-Spam Solution): 메일서버 앞단에 위치하여 프록시 메일서버로 동작

보안 운영체제(Secure OS): 컴퓨터 운영체제의 커널에 보안 기능을 추가한 솔루션

콘텐츠 유출 방지 솔루션

보안 USB: 정보 유출 방지등의 보안 기능을 갖춘 USB 메모리

데이터 유출 방지(DLP; Data Loss Prevention): 조직 내부의 중요 자료가 외부로 빠져나가는 것을 탐지하고 차단하는 솔루션

디지털 저작권 관리(DRM; Digital Right Management): MP3, E-Book과 같은 디지털 저작물에 대한 보호와 관리를 위한 솔루션

비즈니스 연속성 계획(BCP; Business Continuity Plan): 각종 재해, 장애, 재난으로부터 위기관리를 기반으로 재해복구, 업무복구 및 재개등을 통해 비즈니스 연속성을 보장하는 체계

BIA(Business Impact Analysis): 시간 흐름에 따른 영향도 및 손실평가를 조사하는 BCP를 구축하기 위한 비즈니스 영향 분석

RTO(Recovery Time Objective): 업무중단 시점부터 업무가 복구되어 다시 가동될 때까지의 시간

RPO(Recovery Point Objective): 업무중단 시점부터 데이터가 복구되어 정상가동될 때 데이터의 손실 허용 시점

DRP(Disaster Recovery Plan): 재난으로 장기간에 걸쳐 시설 운영이 불가능한 경우를 대비한 재난 복구 계획

DRS(Disaster Recovery System): 재해복구 계획의 원활한 수행을 지원하기 위하여 평상시에 확보하여 두는 인적, 물적 자원 및 이들에 대한 지속적인 관리체계가 통합된 재해복구센터

10. 애플리케이션 테스트 관리 (중요도: ★★)

10-1. 애플리케이션 테스트 케이스 설계

소프트웨어 테스트: 개발된 응용 애플리케이션이나 시스템이 사용자가 요구하는 기능과 성능, 사용성, 안정성 등을 만족하는지 확인하고 노출되지 않도록 숨어있는 소프트웨어의 결함을 찾아내는 활동

결함집중: 파레토 법칙인 오류의 80%는 전체 모듈의 20% 내에서 발견된다는 것이 적용

살충제 패러독스: 동일한 테스트 케이스에 의한 반복적 테스트는 새로운 버그를 찾지

못함

오류-부재의 케번: 요구사항을 충족시켜주지 못한다면, 결함이 없다고 해도 품질링 높다고 볼 수 없음

소프트웨어 테스트 산출물

테스트 계획서: 테스트 목적, 테스트 수행 절차등 테스트 수행을 계획한 문서

테스트 베이시스: 분석, 설계 단계의 논리적인 케이스로 테스트 설계를 위한 기준이 되는 문서

테스트 케이스: 응용 소프트웨어가 사용자의 요구사항을 준수하는지 확인하기 위해 설계된 입력값, 실행 조건, 기대 결과로 구성된 테스트 항목의 명세서

테스트 슈트: 테스트 케이스를 실행환경에 따라 구분해 놓은 테스트 케이스의 집합

테스트 시나리오: 애플리케이션의 테스트 되어야 할 기능 및 특징, 테스트가 필요한 상황을 작성한 문서

테스트 스크립트: 테스트 케이스의 실행 순서를 작성한 문서 (테스트 스텝/절차서라고도 함)

테스트 결과서: 테스트 결과를 정리한 문서로 테스트 프로세스를 리뷰하고 테스트 결과를 평가하고 리포팅하는 문서

정적 테스트: 테스트 대상을 실행하지 않고 구조를 분석하여 논리성을 검증하는 테스트

동적 테스트: 소프트웨어를 실행하는 방식으로 테스트를 수행하여 결함을 검출하는 테스트

화이트박스 테스트: 각 응용 프로그램의 내부 구조와 동작을 검사하는 소프트웨어

[화이트박스 테스트 유형](#) (구결과 조변다 기제데)

구문 커버리지: 프로그램 내의 모든 명령문을 적어도 한 번 수행하는 커버리지

결정 커버리지: 각 분기의 결정 포인트 내의 전체 조건식이 적어도 한 번은 참과 거짓의 결과를 수행하는 커버리지

[조건 커버리지](#): 각 분기의 결정 포인트 내의 각 개별 조건식이 적어도 한 번은 참과 거짓의 결과가 되도록 수행하는 커버리지

조건/결정 커버리지: 전체 조건식뿐만 아니라 개별 조건식도 참 한 번, 거짓 한 번 결과가 되도록 수행하는 테스트 커버리지

변경 조건/결정 커버리지: 개별 조건식이 다른 개별 조건식에 영향을 받지 않고 전체 조건식에 독립적으로 영향을 주도록 함으로써 조건/결정 커버리지를 향상시킨

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 데이터 사용현황을 추가한 그래프를 통해 테스트하는 기법

블랙박스 테스트: 프로그램 외부 사용자의 요구사항 명세를 보면서 수행하는 테스트

블랙박스 테스트 유형 (동경결상 유분폐원비)

동등분할 테스트: 입력 데이터의 영역을 유사한 도메인별로 유효값/무효값을 그룹핑하여 대푯값 테스트 케이스를 도출하여 테스트하는 기법 (**2020년 동등분할테스트**)

경곗값 분석 테스트: 등가 분할 후 경곗값 부분에서 오류 발생 확률이 높기 때문에 경곗값을 포함하여 테스트 케이스를 설계하는 테스트하는 기법

결정 테이블 테스트: 요구사항의 논리와 발생조건을 테이블 형태로 나열하여, 조건과 행위를 모두 조합하여 테스트하는 기법

상태 전이 테스트: 테스트 대상, 시스템이나 객체의 상태를 구분하고 이벤트에 의해 어느 한 상태에서 다른 상태로 전이되는 경우의 수를 수행하는 테스트 기법

유스케이스 테스트: 시스템이 실제 사용되는 유스케이스로 모델링 되어 있을 때 프로세스 흐름을 기반으로 테스트케이스를 명세화하여 수행하는 테스트 기법

분류 트리 테스트: SW의 일부 또는 전체를 트리 구조로 분석 및 표현하여 테스트 케이스를 설계하여 테스트하는 기법

페어와이즈 테스트: 테스트 데이터값들 간에 최소한 한 번씩을 조합하는 방식

원인-결과 그래프 테스트: 그래프를 활용하여 입력 데이터간의 관계 및 출력에 미치는 영향을 분석하여 효용성이 높은 테스트 케이스를 선정하여 테스트하는 기법

비교 테스트: 여러 버전의 프로그램에 같은 입력값을 넣어서 동일한 결과 데이터가 나오는지 비교해보는 테스트 기법

테스트 시간에 따른 분류

검증: 소프트웨어 개발과정을 테스트, 올바른 제품을 생산하고 있는지 검증

확인: 소프트웨어 결과를 테스트, 만들어진 제품이 제대로 동작하는지 확인

테스트 목적에 따른 분류 (회안성 구회병)

회복 테스트: 시스템에 고의로 실패를 유도하고, 시스템의 정상적 복귀 여부를 테스트하는 기법

안전 테스트: 불법적인 소프트웨어가 접근하여 시스템을 파괴하지 못하도록 소스코드 내의 보안적인 결함을 미리 점검하는 테스트 기법

성능 테스트: 사용자의 이벤트에 시스템이 응답하는 시간, 특정 시간 내에 처리하는 업무량, 사용자 요구에 시스템이 반응하는 속도 등을 측정하는 테스트 기법

구조 테스트: 시스템의 내부 논리 경로, 소스 코드의 복잡도를 평가하는 테스트 기법

회귀 테스트: 오류를 제거하거나 수정한 시스템에서 오류 제거와 수정에 의해 새로이 유입된 오류가 없는지 확인하는 일종의 반복 테스트 기법

병행 테스트: 변경된 시스템과 기존 시스템에 동일한 데이터를 입력 후 결과를 비교하는 테스트 기법

성능 테스트의 상세 유형 (부스스내)

부하 테스트: 시스템에 부하를 계속 증가시키면서 시스템의 임계점을 찾는 테스트

스트레스 테스트: 시스템 처리 능력 이상의 부하, 즉 임계점 이상의 부하를 가하여 비정상적인 상황에서의 처리를 테스트

스파이크 테스트: 짧은 시간에 사용자가 몰릴 때 시스템의 반응 측정 테스트

내구성 테스트: 오랜 시간동안 시스템에 높은 부하를 가하는 시스템 반응 테스트

테스트 종류에 따른 분류 (명구경)

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 기법

경험 기반 테스트(블랙박스 테스트): 유사 소프트웨어나 유사 기술 평가에서 테스터의 경험을 토대로 한, 직관과 기술 능력을 기반으로 수행하는 테스트 기법

리뷰의 유형

관리 리뷰: 프로젝트 진행 상황에 대한 전반적인 검토를 바탕으로 범위, 일정, 인력등에 대한 통제 및 의사 결정을 지원하는 리뷰

기술 리뷰: 정의된 계획 및 명세를 준수하고 있는지에 대한 검토를 수행하는 리뷰

인스펙션: 소프트웨어 요구, 설계, 원시 코드 등의 저작자 외의 다른 전문가 또는 팀이 검사하여 문제를 식별하고 문제에 대한 올바른 해결을 찾아내는 형식적인 검토 기법

워크스루: 검토 자료를 회의 전에 배포해서 사전 검토 한 후 짧은 시간동안 회의를 진행하는 비형식적인 검토 기법

감사: 소프트웨어 제품 및 프로세스가 규제, 표준, 가이드라인, 계획, 절차를 준수하고 있는지를 독립적으로 평가하는 기법

정적분석 유형

코딩 표준: 개발자가 프로그램 작성 시 지켜야 할 코딩 표준 및 코딩 지침에 대한 준수 여부 검사

복잡도 측정: 신뢰할 수 있는 척도를 사용하여 프로그램의 복잡도 측정 및 분석 검사

자료 흐름 분석: 프로그램의 자료 흐름에 이상 존재 여부에 대한 분석 검사

동적 테스트-화이트박스 테스트(구조 기반 테스트)-테스트 커버리지 유형 (기라코)

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

라인 커버리지: 애플리케이션 전체 소스코드의 라인 수를 모수로 테스트 시나리오가 수행한 소스코드의 라인 수를 측정하는 방법

코드 커버리지: 소스코드의 구문, 조건, 결정등의 구조 코드 자체가 얼마나 테스트되었는지를 측정하는 방법

경험 기반 테스트 유형 (탐오체특)

탐색적 테스트: 테스트 스크립트 또는 테스트 케이스를 문서로 작성하지 않고 경험에 바탕을 두고 탐색적으로 기능을 수행해보면서 테스트하는 기법

오류 추정: 개발자가 범할 수 있는 실수를 추정하고 이에 따른 결함이 검출되도록 테스트케이스를 설계하여 테스트하는 기법

체크리스트: 테스트하고 평가해야 할 내용과 경험을 분류하여 나열한 이후 하나씩 확인하는 테스트 기법

특성테스트: 국제표준인 ISO/IEC 9126등의 품질모델에 있는 품질 특성을 염두에 두고 이를 근간으로 경험적으로 테스트케이스를 설계하고 테스트하는 기법

테스트 오라클: 테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된

참값을 입력하여 비교하는 기법

테스트 오라클 종류 (참샘휴일)

참 오라클: 모든 입력값에 대하여 기대하는 결과를 생성함으로써 발생된 오류를 모두 검출할 수 있는 오라클

샘플링 오라클: 특정한 몇 개의 입력값에 대해서만 기대하는 결과를 제공해주는 오라클

휴리스틱 오라클: 샘플링 오라클을 개선한 오라클로, 특정 입력값에 대해 올바른 결과를 제공하고 나머지 값들에 대해서는 추정으로 처리하는 오라클

일관성 검사 오라클: 애플리케이션 변경시 수행 전, 후의 결괏값이 동일한지 확인하는 오라클

테스트 레벨: 함께 편성되고 관리되는 테스트 활동의 그룹

테스트 레벨 종류 (단통시인)

단위 테스트: 사용자 요구사항에 대한 단위 모듈, 서브루틴 등을 테스트하는 단계

통합 테스트: 단위 테스트를 통과한 모듈 사이의 인터페이스, 통합된 컴포넌트 간의 상호작용을 검증하는 테스트 단계

시스템 테스트: 통합된 단위 시스템의 기능이 시스템에서 정상적으로 수행되는지를 검증하는 테스트 단계

인수 테스트: 계약상의 요구사항이 만족되었는지 확인하기 위한 테스트 단계

인수 테스트 종류

사용자 인수 테스트: 비즈니스 사용자가 시스템 사용의 적절성 여부 등을 확인하는 테스트

운영상의 인수 테스트: 백업/복원 시스템, 재해 복구, 사용자 관리, 정기 점검 등을 확인하는 테스트

계약 인수 테스트: 계약상의 인수/검수 조건을 준수하는지 여부 등을 확인하는 테스트

규정 인수 테스트: 정부 지침, 법규, 규정 등이 규정에 맞게 개발되었는지 확인하는 테스트

알파 테스트: 선택된 사용자가 개발자 환경에서 통제된 상태로 개발자와 함께 수행하는 인수 테스트

베타 테스트: 실제 환경에서 일정 수의 사용자에게 대상 소프트웨어를 사용하게 하고 피드백을 받는 인수 테스트

10-2. 애플리케이션 통합 테스트

목 객체: 스텝의 객체지향 버전

목 객체 유형 (더스드 스가)

더미 객체: 테스트할 때 객체만 필요하고 해당 객체의 기능까지는 필요하지 않은 경우에 사용

테스트 스텝: 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로 더미 객체에서의 단순 기능에 특정 상태를 가정해서 특정한 값을 리턴하거나 특정 메시지를 출력

테스트 드라이버: 테스트 대상 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출

테스트 스파이: 테스트 대상 클래스와 협력하는 클래스로 가는 출력을 검증하는 데 사용

가짜 객체: 실제 협력 클래스의 기능을 대체해야 할 경우에 사용

하향식 통합: 메인 제어 모듈로부터 아래 방향으로 제어의 경로를 따라 이동하면서 하향식으로 통합하면서 테스트를 진행

상향식 통합: 애플리케이션 구조에서 최하위 레벨의 모듈 또는 컴포넌트로부터 위쪽방향으로 제어의 경로를 따라 이동하면서 구축과 테스트를 수행

샌드위치 통합: 상향식, 하향식 통합 테스트 방식을 결합한 테스트 방식

테스트 자동화 도구 유형 (정실성통)

정적 분석 도구: 만들어진 애플리케이션을 실행하지 않고 분석하는 도구

테스트 실행 도구: 테스트를 위해 작성된 스크립트를 실행하고 작성된 스크립트는 각 스크립트마다 특정 데이터와 테스트 수행 방법을 포함.

성능 테스트 도구: 애플리케이션의 처리량, 응답시간, 경과시간, 자원 사용률에 대해 가상의 사용자를 생성하고 테스트를 수행함으로써 성능 목표를 달성했는지 확인하는 도구

테스트 통제 도구: 테스트 관리 도구, 형상관리 도구, 결함 추적 관리 도구 등이 포함

테스트 하네스: 애플리케이션 컴포넌트 및 모듈을 테스트하는 환경의 일부분으로 테스트를 지원하기 위한 코드와 데이터

테스트 하네스 구성요소 (드 스슈케 스목)

테스트 드라이버: 테스트 대상 하위 모듈을 호출, 파라미터 전달, 모듈 테스트 수행 후의 결과를 도출하는 등 상향식 테스트에 필요

테스트 스크립트: 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로 하향식 테스트에 필요

테스트 슈트: 테스트 케이스의 집합

테스트 케이스: 입력값, 실행 조건, 기대 결과등의 집합

테스트 스크립트: 자동화된 테스트 실행 절차에 대한 명세

목 오브젝트(액체): 사용자의 행위를 조건부로 사전에 입력해두면 그 상황에 예정된 행위를 수행하는 객체

10-3. 애플리케이션 성능 개선

Bad Code: 다른 개발자가 로직(Logic)을 이해하기 어렵게 작성된 코드

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

스파게티 코드: 컴퓨터 프로그램의 소스코드가 복잡하게 얹힌 모습을 스파게티에 비유한 표현

Clean Code: 잘 작성되어 가독성이 높고, 단순하며 중복을 최소화하여 깔끔하게 정리된 코드

Clean Code 작성 원칙 (가단의중추)

가독성: 이해하기 쉬운 용어를 사용

단순성: 한 번에 한 가지 처리만 수행

의존성 최소: 영향도를 최소화, 코드의 변경이 다른 부분에 영향이 없게 작성

중복성 제거: 중복된 코드를 제거, 공통된 코드를 사용

추상화: 클래드/메서드/함수에 대해 동일한 수준의 추상화 구현, 상세 내용은 하위 클래스/메서드/함수에서 구현

느슨한 결합: 클래스의 자료 구조, 메서드를 추상화할 수 있는 인터페이스 클래스를 이용하여 클래스 간의 결합도 최소화

소스코드 품질 정적 분석 도구

pmd: 자바 및 타 언어 소스 코드에 대한 버그, 데드코드 분석

cppcheck: C/C++ 코드에 대한 메모리 누수, 오버플로우 등 문제 분석

SonarQube: 소스 코드 품질 통합 플랫폼, 플러그인 확장 가능

checkstyle: 자바 코드에 대한 코딩 표준 검사 도구

ccm: 다양한 언어의 코드 복잡도 분석 도구

cobertura: jcoverage 기반의 테스트 커버리지 측정 도구

소스코드 품질 동적 분석 도구

Avalanche: Valgrind 프레임워크 및 STP 기반 소프트웨어 에러 및 취약점 동적 분석 도구

Valgrind: 자동화된 메모리 및 스레드 결함 발견 분석 도구

11. 응용 SW 기초 기술 활용 (중요도: ★★★)

11-1. 운영체제의 특징

운영체제: 사용자가 컴퓨터의 하드웨어를 쉽게 사용할 수 있도록 인터페이스를 제공해주는 소프트웨어

커널: 운영체제의 핵심이 되는 기능들이 모여있는 컴퓨터 프로그램

윈도즈 운영체제: MS-DOS의 멀티태스킹 기능과 GUI 환경을 제공하는 마이크로소프트사가 개발한 운영체제

유닉스 계열 운영체제: 교육 및 연구 기관에서 사용되는 범용 다중 사용자 방식의 시분할 운영체제

CLI(Command Line Interface): 사용자가 직접 명령어를 입력하여 컴퓨터에 명령을 내리는 방식

GUI(Graphic User Interface): 마우스로 화면을 클릭하여 그래픽 위주로 컴퓨터를 제어하는 방식

메모리 관리 기법 (반배할교)

반입 기법: 주 기억장치에 적재할 다음 프로세스의 반입 시기를 결정하는 기법

배치 기법: 디스크에 있는 프로세스를 주 기억장치의 어느 위치에 저장할 것인지를 결정하는 기법

할당 기법: 실행해야 할 프로세스를 주 기억장치에 어떤 방법으로 할당할 것인지를 결정하는 기법

교체 기법: 재배치 기법으로 주 기억장치에 있는 프로세스 중 어떤 프로세스를 제거할 것인지를 결정하는 기법

메모리 배치 기법: 최초 적합, 최적 적합, 최악 적합 (초적악)

프로세스 상태: 생성, 준비, 실행, 대기, 완료 (생준실행완)

프로세스 상태 전이 (디타 블웨)

디스패치: 준비 상태에 있는 여러 프로세스 중 실행될 프로세스를 선정하여 CPU 할당

타이머 런 아웃: CPU를 할당받은 프로세스는 지정된 시간이 초과되면 스케줄러에 의해 PCB 저장, CPU 반납 후 다시 준비 상태로 전이됨

블록: 실행 상태에 있는 프로세스가 지정된 할당시간을 초과하기 전에 입출력이나 기타 사건이 발생(block)하면 CPU를 스스로 반납하고 입출력이 완료될 때 까지 대기 상태로 전이됨

웨이크업: 입출력이 종료되면 대기 상태의 프로세스에게 입출력 종료 사실을 **wait & signal** 등에 의해 알려주고, 준비 상태로 전이됨

프로세스 스케줄링 유형

선점형 스케줄링: 하나의 프로세스가 CPU를 차지하고 있을 때 우선순위가 높은 다른 프로세스가 현재 프로세스를 중단시키고 CPU를 점유하는 스케줄링

비선점형 스케줄링: 한 프로세스가 CPU를 할당받으면 작업 종료 후 CPU 반환 시까지 다른 프로세스는 CPU 점유가 불가능한 스케줄링 방식

선점형 스케줄링 알고리즘의 유형

라운드 로빈: 같은 크기의 CPU 시간을 할당, 프로세스가 할당된 시간 내에 처리 완료를 못하면 준비 큐 리스트의 가장 뒤로 보내지고 CPU는 대기중인 다음 프로세스로 넘어감

SRT(Shortest Remaining Time First): 가장 짧은 시간이 소요되는 프로세스를 먼저 수행하고, 남은 처리 시간이 더 짧은 프로세스가 준비 큐에 생기면 이 프로세스가 선점됨

다단계 큐: 작업들을 여러 종류 그룹으로 분할, 여러 개의 큐를 이용하여 상위단계 작업에 의한 하위단계 작업이 선점당함

다단계 피드백 큐: 입출력 위주와 CPU 위주인 프로세스의 특성에 따라 큐마다 서로 다른 CPU 시간 할당량을 부여함

비선점형 스케줄링 알고리즘의 유형

우선순위: 프로세스별로 우선순위가 주어지고, 우선순위에 따라 CPU를 할당함

기한부: 작업들이 명시된 시간이나 기한 내에 완료되도록 계획

FCFS(First Come First Service): 프로세스가 대기 큐에 도착한 순서에 따라 CPU를 할당함. FIFO 알고리즘이라고도 함

SJF(Shortest Job First): 프로세스가 도착하는 시점에 따라 그 당시 가장 작은 서비스 시간을 갖는 프로세스가 종료시까지 자원 점유. 기아 현상 발생 가능성이 있음

HRN(Highest Response Ratio Next): 대기 중인 프로세스 중 현재 응답률이 가장 높은

것을 선택. SJF의 약점인 기아 현상을 보완한 기법.
 $(\text{응답률} = (\text{대기시간} + \text{서비스시간}) / \text{서비스시간})$

도착 시간 / 서비스 시간 / 종료 시간 / 반환 시간(종료-도착) / 대기 시간(반환-서비스)

가상화: 물리적인 리소스들을 사용자에게 하나로 보기 위해 하거나, 하나의 물리적인 리소스를 여러 개로 보이게 하는 기술

가상화의 종류

플랫폼 가상화: 하드웨어 플랫폼 위에서 실행되는 호스트 프로그램이 게스트 프로그램을 만들어 마치 독립된 환경을 만들어 낸 것처럼 보여주는 기법

리소스 가상화: 게스트 소프트웨어 위에서 사용자는 독립된 하드웨어에서 소프트웨어가 실행되는 것처럼 활용하는 기법

가상화 기술요소

컴퓨팅 가상화: 물리적으로 컴퓨터 리소스를 가상화하여 논리적 단위로 리소스를 활용할 수 있도록 하는 기술

스토리지 가상화: 스토리지와 서버 사이에 소프트웨어/하드웨어 계층을 추가하여 스토리지를 논리적으로 제어 및 활용할 수 있도록 하는 기술

I/O 가상화: 서버와 I/O 디바이스 사이에 위치하는 미들웨어 계층으로 서버의 I/O 자원을 물리적으로 분리하고 케이블과 스위치 구성을 단순화하여 효율적인 연결을 지원하는 기술

컨테이너: 컨테이너화된 애플리케이션들이 단일 운영체제상에서 실행되도록 해주는 기술

분산처리 기술: 여러 대의 컴퓨터 계산 및 저장능력을 이용하여 커다란 계산 문제나 대용량의 데이터를 처리하고 저장하는 기술

네트워크 가상화 기술: 물리적으로 떨어져 있는 다양한 장비들을 연결하기 위한 수단으로 중계장치의 가상화를 통한 가상 네트워크를 지원하는 기술

클라우드 컴퓨팅: 인터넷을 통해 가상화된 컴퓨터 시스템 리스스를 제공하고, 정보를 자신의 컴퓨터가 아닌 클라우드에 연결된 다른 컴퓨터로 처리하는 기술

클라우드 컴퓨팅 분류 (사공하)

사설 클라우드: 기업 또는 조직 내부에서 보유하고 있는 컴퓨팅 자원을 사용하여 내부에 구축되어 운영되는 클라우드

공용 클라우드: 클라우드 서비스 제공업체에서 다중 사용자를 위한 컴퓨팅 자원

서비스를 제공하는 클라우드

하이브리드 클라우드: 기업 또는 조직 내부 자원을 이용한 사설 클라우드와 공용 클라우드를 모두 사용하는 클라우드

클라우드 컴퓨팅 유형 (인플소)

인프라형 서비스(IaaS; Infrastructure as a Service): 서버, 스토리지 같은 시스템 자원을 클라우드로 제공하는 서비스

플랫폼형 서비스(PaaS; Platform as a Service): 인프라를 생성, 관리하는 복잡함 없이 애플리케이션을 개발, 실행, 관리 할 수 있게 하는 플랫폼을 제공하는 서비스

소프트웨어형 서비스(SaaS; Software as a Service): 소프트웨어 및 관련 데이터는 중앙에 호스팅되고 사용자는 웹 브라우저등의 클라이언트를 통해 접속하여 소프트웨어를 서비스 형태로 이용하는 서비스

블록체인형 서비스(BaaS; Blockchain as a Service): 블록체인 개발환경을 클라우드로 서비스하는 개념

메타버스(Metaverse): 가상을 의미하는 메타(meta)와 현실세계를 의미하는 유니버스(universe)의 합성어로 3차원 가상세계를 의미

11-2. 네트워크 기초 활용하기

네트워크: 원하는 정보를 원하는 수신자 또는 기기에 정확하게 전송하기 위한 기반 인프라

광대역 네트워크(WAN): LAN에 비해 전송 거리가 넓고 라우팅 알고리즘 필요

근거리 네트워크(LAN): 한 건물, 작은 지역을 커버하는 네트워크

OSI(Open System Interconnection) 7계층: (A,P,S,T,Ne,Da,Phy 아파서 티내다 피나다)

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

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

세션 계층(Session Layer): 송수신간의 논리적인 연결, 연결 접속, 동기제어

전송 계층(Transport Layer): 송수신 프로세스 간의 연결, 신뢰성 있는 통신 보장, 데이터 분할, 재조립, 흐름 제어, 오류 제어, 혼잡 제어

네트워크 계층(Network Layer): 단말기간 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

네트워크 장비

1계층 장비

허브: 여러 대의 컴퓨터를 연결하여 네트워크로 보내거나, 하나의 네트워크로 수신된 정보를 여러 대의 컴퓨터로 송신하기 위한 장비

리피터: 디지털 신호를 증폭시켜주는 역할을 하며 신호가 약해지지 않고 컴퓨터로 수신되도록 하는 장비

2계층 장비

브리지: 두 개의 근거리 통신망을 서로 연결해주는 통신망 연결 장치

L2 스위치: 브리지, 허브의 단점을 개선하기 위해서 출발지에서 들어온 프레임을 목적지 MAC 주소 기반으로 빠르게 전송시키는 데이터 링크 계층의 통신 장비

NIC: Network Interface Card의 약자. 외부 네트워크와 접속하여 가장 빠른 속도로 데이터를 주고받을 수 있게 컴퓨터 내에 설치되는 장치

스위칭 허브: 스위치 기능을 가진 허브

3계층 장비

라우터: LAN과 LAN을 연결하거나 LAN과 WAN을 연결하기 위한 인터넷 네트워킹 장비. 패킷의 위치를 추출하여 최적의 경로를 지정, 패킷을 다음 장치로 전송시키는 장치

게이트웨이: 프로토콜을 서로 다른 통신망에 접속할 수 있게 해주는 장치

L3 스위치: 3계층에서 네트워크 단위들을 연결하는 통신 장비. IP 레이어에서 스위칭 수행

유무선 인터넷 공유기: 외부로부터 들어오는 인터넷 라인을 연결하여 공유하도록 하는 장비

망 스위치 허브: 광역 네트워크를 커버하는 스위칭 허브

4계층 장비

L4 스위치: 4계층에서 네트워크 단위들을 연계하는 통신 장비. TCP/UDP 등 스위칭 수행

프로토콜: 서로 다른 시스템이나 기기들 간의 데이터 교환을 원활히 하기 위한 표준화된 통신규약

프로토콜의 3요소 (구의타)

구문(Syntax): 시스템 간의 정보 전송을 위한 데이터 형식

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 특징

단편화: 전송이 가능한 작은 블록으로 나누어지는 기법

재조립: 단편화되어 온 조각들을 원래 데이터로 복원하는 기법

캡슐화: 상위 계층의 데이터에 각종 정보를 추가하여 하위 계층으로 보내는 기법

동기화: 송신과 수신 측의 시점을 맞추는 기법

다중화: 하나의 통신 회선에 여러 기기들이 접속할 수 있는 기술

데이터 링크 계층 프로토콜

HDLC(High-level Data Link Control): 점대점 방식이나 다중방식의 통신에 사용되는 ISO에서 표준화한 동기식 비트 중심의 데이터 링크 프로토콜

PPP(Point-to-Point Protocol): 네트워크 분야에서 두 통신 노드 간의 직접적인 연결을 위해 일반적으로 사용되는 데이터 링크 프로토콜

프레임 릴레이: 프로토콜 처리를 간략화하여 단순히 데이터 프레임들의 중계(Relay) 기능과 다중화 기능만 수행함으로써 데이터 처리속도의 향상시킨 고속의 데이터 전송기술

ATM(Asynchronous Transport Mode): 정보전달의 기본단위를 53바이트 셀 단위로 전달하는 비동기식 시분할 다중화 방식의 패킷형 전송 기술

네트워크 계층 프로토콜

IP(Internet Protocol): 송수신 간의 패킷 단위로 데이터를 교환하는 네트워크에서 정보를 주고받는 데 사용하는 통신 프로토콜

ARP(Address Resolution Protocol): IP 네트워크 상에서 IP 주소를 MAC 주소(물리 주소)로 변환하는 프로토콜

RARP(Reverse Address Resolution Protocol): IP 호스트가 자신의 물리 네트워크 주소(MAC)는 알지만 IP 주소를 모르는 경우, 서버로부터 IP 주소를 요청하기 위해서 사용하는 프로토콜

ICMP(Internet Control Message Protocol): IP 패킷을 처리할 때 발생되는 문제를

알려주는 프로토콜 메시지. 형식은 8바이트의 헤더와 가변 길이의 데이터 영역으로 분리됨

IGMP(Internet Group Management Protocol): 호스트 컴퓨터와 인접 라우터가 멀티캐스트 그룹 멤버십을 구성하는 데 사용하는 프로토콜

라우팅 프로토콜: 데이터 전송을 위해 목적지까지 갈 수 있는 여러 경로 중 최적의 경로를 설정해주는 라우터 간의 상호 통신 프로토콜

IPv4: 인터넷에서 사용되는 패킷 교환 네트워크 상에서 데이터를 교환하기 위한 32비트 주소체계를 갖는 네트워크 계층의 프로토콜

IPv6: **IPv4**가 가지고 있는 주소 고갈, 보안성, 이동성 지원등의 문제점을 해결하기 위해서 개발된 **128bit** 주소 체계를 갖는 차세대 인터넷 프로토콜

IPv4 전송 방식: 유니캐스트, 멀티캐스트, 브로드캐스트 (유멀브)

IPv6 전송 방식: 유니캐스트, 멀티캐스트, 애니캐스트 (유멀애)

IPv4에서 IPv6로 전환 방법

듀얼 스택: IP 계층에 두 가지의 프로토콜이 모두 탑재되어 있고 통신 상대방에 따라 해당 IP 스택을 선택하는 방법

터널링: IPv6 망에서 인접한 IPv4 망을 거쳐 다른 IPv6 망으로 통신 할 때 IPv4 망에 터널을 만들고 IPv4에서 사용하는 프로토콜로 캡슐화하여 전송하는 방법

주소변환: IPv4 망과 IPv6 망 사이에 주소 변환기를 사용하여 서로 다른 네트워크 상의 패킷을 변환시키는 방법

멀티캐스트 프로토콜: 인터넷에서 같은 내용의 데이터를 여러 명의 특정한 그룹의 수신자들에게 동시에 전송할 수 있는 프로토콜

유니캐스트 프로토콜: 고유 주소로 식별된 하나의 네트워크 목적지에 1:1로 트래픽 또는 메시지를 전송하는 프로토콜

브로드캐스트 프로토콜: 하나의 송신자가 같은 서브 네트워크 상의 모든 수신자에게 데이터를 전송하는 프로토콜

애니캐스트 프로토콜: 단일 송신자로부터의 데이터그램을 토플로지상의 잠재적인 수신자 그룹 안에서 가장 가까운 노드로 연결시키는 전송 프로토콜

라우팅 프로토콜

RIP(Routing Information Protocol): AS(자율 시스템)내에서 사용하는 거리 벡터 알고리즘에 기초하여 개발된 내부 라우팅 프로토콜. 벨만-포드 알고리즘 사용, 15홉

제한

OSPF(Open Shortest Path First): 자신을 기준으로 링크 상태 알고리즘을 적용하여 최단 경로를 찾는 라우팅 프로토콜. 다익스트라 알고리즘 사용

BGP(Border Gateway Protocol): AS 상호간에 경로 정보를 교환하기 위한 라우팅 프로토콜. 경로 벡터 알고리즘을 사용함. ISP 사업자들간에 주로 사용됨.

라우팅 알고리즘 유형

거리 벡터 알고리즘: 인접 라우터와 정보를 공유하여 목적지까지의 방향을 결정하는 라우팅 프로토콜 알고리즘

링크 상태 알고리즘: 링크 상대 정보를 모든 라우터에 전달하여 최단 경로 트리를 구성하는 라우팅 프로토콜 알고리즘

TCP(Transmission Control Protocol): 전송 계층에 위치하면서 근거리 통신망이나 인트라넷, 인터넷에 연결된 컴퓨터에서 실행되는 프로그램 간에 인련의 옥텟을 안정적으로, 순서대로, 에러 없이 교환 할 수 있게 해주는 프로토콜.

TCP 특징: 신뢰성 보장, 연결 지향적 특징, 흐름 제어, 혼잡 제어

UDP(User Datagram Protocol): 비연결성이고, 신뢰성이 없으며 순서화되지 않은 데이터그램 서비스를 제공하는 전송 계층의 통신 프로토콜

세션 계층 프로토콜

RPC(Remote Procedure Call): 원격 프로시저 호출이라고 불리며, 별도의 원격 제어를 위한 코딩 없이 다른 주소 공간에서 프로시저를 실행할 수 있는 프로세스 간 통신에 사용되는 프로토콜

NetBIOS(Network Basic Input/Output System): 응용 계층의 애플리케이션 프로그램에게 API를 제공하여 상호 통신할 수 있도록 해주는 프로토콜

표현 계층 프로토콜

JPEG: 이미지를 위해 만들어진 표준 규격

MPEG: 멀티미디어를 위해 만들어진 표준 규격

응용 계층 프로토콜

HTTP(HyperText Transfer Protocol): 텍스트 기반의 통신 규약으로 인터넷에서 데이터를 주고받을 수 있는 프로토콜

FTP(File Transfer Protocol): TCP/IP 프로토콜을 가지고 서버와 클라이언트 사이의

파일을 전송하기 위한 프로토콜

SMTP(Simple Mail Transfer Protocol): 인터넷에서 TCP 포트번호 25번을 사용하여 이메일을 보내기 위해 이용되는 프로토콜

POP3(Post Office Protocol Version3): 원격 서버로부터 TCP/IP 연결을 통해 이메일 가져오는 데 사용하는 프로토콜

IMAP(Internet Messaging Access Protocol): 원격 서버로부터 TCP/IP 연결을 통해 이메일을 가져오는 데 사용하는 프로토콜

Telnet: 인터넷이나 로컬 영역에서 네트워크 연결에 사용되는 네트워크 프로토콜

패킷 스위칭: 컴퓨터 네트워크와 통신의 방식 중 하나로 작은 블록의 패킷으로 데이터를 전송하며 데이터를 전송하는 동안만 네트워크 자원을 사용하는 통신 방식

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

서킷 스위칭: 네트워크 리소스를 특정 사용층이 독점하도록 하는 통신 방식

11-3. 기본 개발환경 구축하기

개발환경 인프라 구성 방식

온프레미스(On-Premise) 방식: 외부 인터넷망이 차단된 상태에서 인트라넷망만을 활용하여 개발환경을 구축하는 방식

클라우드 방식: 아마존, 구글 등 클라우드 공급 서비스를 하는 회사들의 서비스를 임대하여 개발환경을 구축하는 방식

하이브리드 방식: 온프레미스와 클라우드 방식을 혼용하는 방식

CDN(Contents Delivery Network): 콘텐츠를 효율적으로 전달하기 위해 여러 노드를 가진 네트워크에 데이터를 저장하여 제공하는 시스템

12. 제품 소프트웨어 패키징 (중요도: ☆)

12-1. 제품 소프트웨어 패키징하기

모듈화: 모듈을 이용하여 소프트웨어의 성능을 향상시키거나 시스템의 디버깅, 시험, 통합 및 수정을 용이하도록 하는 모듈 중심의 소프트웨어 설계법

릴리즈 노트: 최종 사용자인 고객에게 개발 과정에서 정리된 제품의 릴리즈 정보를 제공하는 문서

제품 소프트웨어 패키징 도구: 배포를 위한 패키징 시에 디지털 콘텐츠의 지적 재산권을 보호하고 관리하는 기능을 제공하며, 안전한 유통과 배포를 보장하는 도구

12-2. 제품 소프트웨어 매뉴얼 작성 및 버전 등록

백업의 유형 (전, 차, 증)

전체 백업: 백업받고자하는 데이터 전체에 대해 백업하는 방식

차등 백업: 마지막 전체 백업 이후 변경된 모든 데이터를 백업하는 방식

증분 백업: 정해진 시간을 기준으로 그 이후에 변경된 파일만을 백업하는 방식