1. 요구사항 확인

소프트웨어 생명주기(SDLC)시스템의 전 공정을 체계화한 절차

SDLC 모델 종류

- 폭포수 모델 : 각 단계를 확실히 마무리 지은 후에 다음 단계로 넘어감, 선형 순차적 모형(고전적 생명주기 모형)
- 프로토타이핑 모델 : 프로토타입을 구현해, 고객의 피드백을 반영하며 만들어 간다
- 나선형 모델 : 위험을 최소화하기 위해 점진적으로 개발
- 반복적 모델 : 구축 대상을 나누어 병렬적으로 개발 후 통합하거나, 반복적으로 개발(SDLC 모델)

소프트웨어 개발방법론

- 구조적 방법론 : 전체 시스템을 기능에 따라 나누어 개발하고, 이를 통합.(하향식 방법론) 나씨-슈나이더만 차트 사용(도형식, 제어 논리 구조, 명확한 식별)
- 정보공학 방법론 : 정보시스템 개발에 필요한 관리 절차와 작업 기반을 체계화
- 객체지향 방법론 : 복잡한 현실 세계를 사람이 이해하는 방식으로 시스템에 적용
- 컴포넌트 기반 방법론(CBD) : 컴포넌트를 조립해 하나의 새로운 응용 프로그램 작성(생산성, 확장성, 재사용)
- 애자일 방법론 : 절차보다는 사람이 중심, 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템 개발
- 제품 계열 방법론 : 특정 제품에 적용하고 싶은 공통된 기능을 정의해 개발, 임베디드에 유용

애자일

- XP : 의사소통 개선과 즉각적 피드백

    - 5가지 가치 : 용기, 단순성, 의사소통, 존중, 피드백
    - 12가지 기본 원리
        - 짝 프로그래밍(Pair Programming) : 개발자 둘이서 짝 코딩
        - 공동 코드 소유(Collective Ownership) : 시스템 코드는 누구든지 언제라도 수정 가능
        - 지속적인 통합(Continuous Integration) : 매일 여러 번씩 SW통합, 빌드 해야함
        - 계획 세우기(Planning Process) :개발자가 필요한 것은 무엇이며 어떤 부분에서 지연될 수 있는지를 알려줘야함
        - 작은 릴리즈(Small Release) : 작은 시스템 먼저 만들고 짧은 단위 업데이트
        - 메타포어(Metathor) : 공통적인 이름 체계와 시스템 서술서를 통해 고객과 개발자 간의 의사소통을 원활하게
        - 간단한 디자인(Simple Design): 요구사항에 적합한 가장 단순한 시스템 설계
        - 테스트 기반 개발(Test Driven Develop)
        - 리팩토링(Refactoring) : 프로그램의 기능은 바꾸지 않고 중복제거, 단순화 등을 위해 시스템 재구성
        - 40시간 작업 : 40시간 이상 일 X
        - 고객 상주(On Site Customer) : 개발자들의 질문에 대답할 수 있는 고객 풀타임 상주
        - 코드 표준(Coding Standard) : 효과적인 공동 작업을 위해 코딩 표준을 정의
- SCRUM : 매일 정해진 시간, 장소에서 짧은 시간의 개발
    - 백로그(Backlog) : 제품과 프로젝트에 대한 요구사항
    - 스프린트(Sprint) : 2~4주의 짧은 개발 기간 반복적 수행
    - 스크럼 미팅(Scrum Meeting) : 매일 15분 정도 미팅
    - 스크럼 마스터(Scrum Master) : 프로젝트 리더
    - 스프린트 회고(Sprint Retrospective) : 스프린트 주기 되돌아보며 규칙 준수 여부, 개선점 확인
    - 번 다운 차트(Burn Down Chart)

- LEAN : 도요타, 낭비 요소를 제거하여 품질 향상
    - 낭비제거, 품질 내재화, 지식 창출, 늦은 확정, 빠른 인도, 사람 존중, 전체 최적화

비용산정 모형

- 하향식
    - 델파이 기법 : 전문가의 경험적 지식을 통한 문제 해결 및 미래예측을 위한 기법
- 상향식
    - LoC(Lind of Code) : 원시 코드 라인 수의 낙관치, 중간치, 비관치를 측정해 **예측치**를 구해 비용을 산정하는 방식
    - Man Month : 한 사람이 1개월 동안 할 수 있는 일의 양을 기준으로 프로젝트 비용 산정하는 방식
        - 프로젝트 기간 = man month(LoC/프로그래머 월간 생산성)/프로젝트 인력
    - COCOMO : 보헴이 제안, 프로그램 규모에 따른 비용 산정
        - 조직형(Organic Mode) : 5만 라인 이하
        - 반 분리형(Semi-Detached Mode) : 30만 라인 이하
        - 임베디드형(Embedded Mode) : 30만 라인 이상
    - 푸트남 : 개발주기의 단계별로 요구할 인력의 분포를 가정하는 방식, 생명주기 예측 모형, Rayleigh-Norden 곡선
    - 기능점수(FP) : 요구 기능에 따른 가중치 부여

일정관리 모델

- 주 공정법(CPM) : 여러 작업의 수행 순서가 얽혀 있는 프로젝트의 일정 계산(임계 경로는 가장 오래 걸리는 경로)
- PERT : 일의 순서를 계획적으로 정리하기 위한 수렴 기법, 비관치, 중간치, 낙관치의 **3점 추정방식** 이용
- 주 공정 연쇄법(CCPM) : 자원제약사항을 고려해 일정 작성

현행 시스템 파악

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 구초체

소프트웨어 아키텍처 4+1 뷰

고객의 요구사항을 정리해 놓은 시나리오를 4개의 관점에서 바라보는 소프트웨어적인 접근방법

- 유스케이스 뷰 : 유스케이스 또는 아키텍처 도출, 다른 뷰를 검증하는데 사용
- 논리 뷰 : 시스템의 기능적인 요구사항
- 프로세스 뷰 : 시스템의 비기능적 요구사항
- 구현 뷰 : 모듈의 구성을 보여줌
- 배포 뷰 : 어떻게 배치되는가

소프트웨어 아키텍처 패턴 유형

- 계층화 패턴(Layered) : 서로 마주 보는 두 개의 계층 사이에서만 상호작용
- 클라이언트-서버 패턴 : 하나의 서버와 다수의 클라이언트
- 파이프-필터 패턴 : 데이터 스트림을 생성하고 처리하는 시스템에서 사용
- 브로커패턴 : 분리된 컴포넌트들로 이루어진 분산 시스템에서 사용되고, 원격 서비스 실행을 통해 상호작용이 가능
- MVC패턴
    - 모델 : 핵심 기능, 데이터 보관
    - 뷰 : 사용자에게 정보 표시
    - 컨트롤러 : 사용자로부터 요청 입력 받아 처리

소프트웨어 아키텍처 비용 평가 모델 종류

- SAAM : 변경 용이성, 기능성에 집중
- ATAM : 아키텍처 품질 속성을 만족시키는지 판단
- CBAM : 경제적 의사결정에 대한 요구를 충족하는지
- ADR : 응집도 평가 모델
- ARID : 특정 부분 품질 요소

디자인 패턴 ⭐⭐⭐⭐⭐

SW설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계 방법을 정리한 패턴

- 생성 bprofas 비프로파스
  - builder : 복잡한 인스턴스를 조립해 만드는 구조
  - prototype : 처음부터 일반적인 원형을 만들어 놓고, 그것을 복사한 후 필요한 부분만 수정해 사용하는 패턴
  - factory method : 상위 클래스에서 인터페이스 정의, 하위클래스에서 인스턴스 생성
  - abstract factory : 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공
  - singleton : 전역 변수 사용하지 않고 객체 하나만 생성, 그 객체는 어디서든지 참조할 수 있음
- 구조 abcdffp
  - adapter : 기존에 생성된 클래스를 재사용할 수 있도록 중간에서 맞춰주는 역할
  - bridge : 기능 계층과 구현 계층을 연결, 구현부에서 추상 계층 분리
  - composite : 객체들의 관계를 트리 구조로 구성
  - decorator : 기존에 구현되어 있는 클래스에 필요한 기능 추가해 나감
  - facade : 복잡한 시스템에 대해 단순한 인터페이스 제공, 시스템 구조에 대한 파악 쉽게
  - flyweight : 메모리 절약, '클래스의 경량화'목적
  - proxy : 실체 객체에 대한 대리 객체

- 행위
  - Mediator : 중간에 통제, 중재자
  - Interpreter : 언어의 다양한 해석, 구문의 해석을 맡는 클래스 각각 작성
  - Iterator : 컬렉션 구현 방법 노출시키지 않으면서도 그 집합체 안에 들어있는 모든 항목에 접근할 방법을 제공
  - Template Method : 상위 클래스-추상, 하위 클래스-구체
  - Observer : 한 객체의 상태가 바뀌면 그 객체에 의존하는 다른 객체들에 연락
  - State : 상태에 따라 다르게 처리할 수 있도록 행위 내용 변경
  - Visitor : 클래스의 메서드가 각 클래스를 돌아다니며 특정 작업 수행
  - Command : 명령이 들어오면 그에 맞는 서브 클래스 선택되어 실행
  - Strategy : 알고리즘 군 정의, 행위를 클래스로 캡슐화해 동적으로 행위 자유롭게 변환
  - Memento : Undo(작업취소) 기능 개발
  - Chain of Responsibility : 정적으로 어떤 기능에 대한 처리의 연결이 하드 코딩 되어 있을 때, 이를 동적으로 연결되어 있는 경우에 따라 다르게 처리될 수 있도록 연결한 디자인

운영체제

컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스 담당

- 윈도즈 : 중/소규모 서버, 일반 pc
- 유닉스 : 대용량 처리, 엔터프라이즈 급 서버
- 리눅스 : 중/대규모 서버 대상, 높은 보안성, 비용 가장 적음
- 안드로이드 : 리눅스 위에서 구동, 자바와 코틀린으로 작성
- IOS : 높은 보안성, 고성능

운영체제 현행 시스템 분석 고려사항(신성기주구)

신뢰도, 성능, 기술 지원, 주변 기기 , 구축 비용

미들웨어

응용 프로그램과 환경 간에 원만한 통신이 이루어질 수 있도록 제어해주는 SW

- WAS : 서버계층에서 애플리케이션이 동작할 수 있는 환경 제공, 트랜잭션 처리, 이기종 시스템 연동

요구공학

사용자 요구사항에 대한 도출, 분석, 명세 , 확인 및 검증하는 구조화된 활동

요구사항

- 기능적 요구사항 : 시스템이 제공하는 기능, 서비스에 대한 요구사항(사용자UI)
- 비기능적 요구사항 : 시스템이 수행하는 기능 이외의 사항(백엔드)

요구사항 개발 단계(도분명확)

- 도출
  - 인터뷰 : 이해 관계자와 직접 대화
  - 설문조사 : 설문지, 여론조사
  - 브레인스토밍 : 말을 꺼내기 쉬운 분위기로 만들어 비판 없이 수용할 수 있도록 하는 회의
  - 델파이 기법 : 전문가의 경험적 지식을 통한 문제 해결 방법
  - 롤 플레잉 : 여러 사람이 각자가 맡은 역을 연기
  - 워크숍 : 단기간에 다양하고 전문적인 정보를 획득하고 공유
- 분석
  - 청취 기술
  - 인터뷰와 질문 기술
- 명세
  - 비정형 명세 기법

        - 자연어 기반
        - 사용자와 개발자 이해 용이
        - 명확성 및 검증 문제
    - 정형 명세 기법
        - 수학적인 원리와 표기법, Z-스키마, Petri Nets
        - 표현 간결, 명확성 및 검증 용이
        - 기법 이해 어려움
- 확인 및 검증
    - 정형 기술 검토
        - 동료 검토 : 2~3명이 진행, 작성자가 명세서 설명하고 이해관계자들이 설명을 들으면서 결함 발견
        - 워크스루 : 검토자료를 회의 전에 배포해서 사전검토한 후 짧은 시간동안 회의 진행
        - 인스펙션 : 저작자 외의 다른 전문가 또는 팀이 검사하여 오류를 찾아내는 공식적 검토 방법

2. 화면설계

UI 유형

- CLI (Command Line Interface): 명령어를 텍스트로
- GUI(Grapic User Interface) : 마우스, 전자펜
- NUI :터치, 음성
- OUI(Organic User Interface) : 현실에 존재하는 모든 사물

UI 설계 원칙 / 직유학유

- 직관성(Intuitiveness) : 누구나 쉽게 이해하고, 쉽게 사용할 수 있어야 함
- 유효성(Efficiency) : 정확하고 완벽하게 사용자의 목표가 달성 될 수 있도록 제작
- 학습성(Learnability) : 초보와 숙련자 모두가 쉽게 배우고 사용할 수 있게 제작
- 유연성(Flexibility) : 사용자의 요구사항을 최대한 수용하고, 실수를 방지할 수 있도록 제작

UI 품질 요구사항 / 기신사효유이

- 기능성(Functionality) : 실제 사용시 정확하지 않은 결과가 발생할 확률과 시스템의 동작 관찰
  - 적절성, 정밀성, 상호 운용성, 보안성, 호환성
- 신뢰성(Realiability): 일정한 시간, 작동되는 시간동안 의도하는 기능을 수행함을 보증
  - 성숙성, 고장 허용성, 회복성
- 사용성(Usablity) : 어떠한 행위를 정확하고 쉽게 인지할 수 있는
  - 이해성, 학습성, 운용성
- 효율성(Efficiency) : 할당된 시간에 한정된 자원으로 얼마나 빨리 처리할 수 있는가
  - 시간 효율성, 자원 효율성
- 유지보수성(Maintainability) : 요구사항 개선, 확장에 있어 얼마나 용이한가

    - 분석성, 변경성, 안정성, 시험성
- 이식성(Portability) : 다른 운영체제에서도 얼마나 쉽게 적용이 가능한가
    - 적용성, 설치성, 대체성

UI 개발을 위한 주요 기법

- 3C 분석 : 고객(Customer), 자사(Company), 경쟁사(Competitor) 비교하고 분석
- SWOT 분석 : 기업 내부 환경과 외부환경을 분석해 Strength, Weakness, Opportunity, Threat 요인을 규정하고 이를 토대로 경영 전략 수립
- 시나리오 플래닝 : 상황 변화를 사전에 예측하고 다양한 시나리오를 설계하는 방법
- 사용성 테스트 : 사용자가 직접 제품을 사용하면서 미리 작성된 시나리오에 맞추어 과제를 수행 한 후, 질문에 답하도록 하는 테스트
- 워크숍 : 소집단 인원으로 특정 문제나 과제에 대한 새로운 지식, 기술, 아이디어들을 서로 교환하고 검토하는 세미나

UI 화면 설계

- 스토리보드 : 정책, 프로세스, 와이어 프레임, 기능 정의, 데이터베이스 연동 등 서비스 구축을 위한 정보가 수록된 문서, 디자이너와 개발자가 최종적으로 참고하는 산출 문서
- 와이어 프레임 : 화면 단위의 레이아웃을 설계하는 작업
- 프로토타입 : 정적인 화면(와이어 프레임, 스토리보드)에 동적 효과를 적용해 실제 구현된 것처럼 시뮬레이션 할 수 있는 모형

UML(Unified Modeling Language)

객체지향 소프트웨어 개발 과정에서 산출물을 명세화, 시각화, 문서화 할 때 사용되는 모델링 기술과 방법론을 통합해서 만든 표준화된 범용 모델링 언어

UML구성요소

사물, 관계, 다이어그램

UML 다이어그램

- 구조적(Structural) 다이어그램 / 정적(Static) 다이어그램
  - 클래스(Class) : 클래스 간 정적인 관계를 표현
  - 객체(Object): 클래스에 속한 사물, 인스턴스
  - 컴포넌트(Component) : 컴포넌트와 그들 사이 의존 관계
  - 배치(Deployment) : 컴포넌트 사이의 종속성, 물리적 요소들의 위치
  - 복합체 구조(Composite Structure) : 클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현
  - 패키지(Package) : 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현
- 행위적(Behavioral) 다이어그램 / 동적(Dynamic) 다이어그램
  - 유스케이스 : 시스템 외부 요소를 사용자의 관점에서 표현
  - 시퀀스 : 시간적 개념을 중심으로 메시지 흐름으로 표현
  - 커뮤니케이션 : 객체들이 주고받는 메시지를 표현하고 객체 간의 연관까지 표현
  - 상태 : 상태가 어떻게 변화하는지 표현
  - 활동 : 어떤 기능을 수행하는지, 처리 로직이나 처리 흐름
  - 타이밍 : 객체 상태 변화와 시간 제약을 명시적으로 표현

UML 확장 모델의 스테레오 타입

'<< >>'(길러멧; Guillemet) 기호를 사용하여 표현

클래스 다이어그램

- 접근제어자
  - public + : 클래스 외부 접근 허용
  - private - : 클래스 내부 접근 허용
  - protected # : 동일 패키지/파생 클래스에서 접근

    - default ~ : 동일 패키지 클래스에서 접근
- 클래스 간의 관계
    - 연관 : 실선, 2개 이상의 사물이 서로 관련되어 있는 상태
    - 집합 : 속이 빈 마름모 (차/엔진, 바퀴, 운전대), 하나의 객체에 여러 개의 독립적인 객체들이 구성
    - 복합(=포함) : 속이 채워진 마름모, 집합 보다 더 강한 관계
    - 일반화 : 부모-자식, 속이 빈 화살표 (차/버스,택시,자가용), 상속 관계
    - 의존 : 점선 화살표, 서로 연관은 있으나 필요에 따라 짧은 시간동안만 연관을 유지

UI 시나리오 문서의 작성 요건(완일이가 추수)

완전성, 일관성, 이해성, 가독성, 추적 용이성, 수정 용이성

3. 데이터 입출력

1) 논리데이터

데이터 모델

현실 세계의 정보를 인간과 컴퓨터가 이해할 수 있도록 추상화하여 표현한 모델

데이터 모델 절차

요구사항 분석 > 개념적 > 논리적(정규화) > 물리적(반정규화)

논리 데이터 모델 종류

- 관계 데이터 모델 : 1:1, 테이블
- 계층 데이터 모델 : 1:N, 트리
- 네트워크 데이터 모델 : N:M, 그래프

논리 데이터 모델링 속성

개체(entity), 속성(attribute), 관계(relationship)

관계 데이터 모델

- 튜플(tuple), 행(row), 카디널리티(cardinality)
- 속성(attribute), 열(column), 차수(degree)

관계 대수

절차적 언어

- 일반 집합 연산자 : 합집합(∪), 교집합(∩), 차집합(–), 카티션 프로덕트(X)
- 순수 관계 연산자 :
  - 셀렉트(σ) : R에서 조건을 만족하는
  - 프로젝트(π) : R에서 주어진 속성들의 값으로만 구성된
  - 조인(⋈) : 공통 속성을 이용
  - 디비전(÷) : 릴레이션 S의 모든 튜플과 관련 있는 R의 튜플 반환

관계 해석

튜플 관계 해석과 도메인 관계해석을 하는 비절차적 언어

개체-관계(E-R) 모델

현실 세계에 존재하는 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 명확하게 표현하기 위해 사용되는 모델

- 개체 □ 관계 ◇ 속성 ○ 다중 값 속성 ◉ 관계-속성 ─

정규화(Nomalization)

데이터의 중복성을 제거해 이상현상을 방지하고, 데이터의 일관성과 정확성을 유지하기 위해 무손실 분해하는 과정

- 1NF : 도메인이 원자값
- 2NF : 부분함수 종속 제거
- 3NF : 이행함수 종속 제거(A→ B , B→C 이면 A→C)
- BCNF : 결정자 후보 키가 아닌 함수 종속 제거
- 4NF : 다치(다중 값) 종속 제거
- 5NF : 조인 종속 제거

이상 현상(Anomaly)

데이터의 중복성으로 인해 릴레이션을 조작할 때 발생하는 비합리적인 현상

- 삽입 이상, 삭제 이상, 갱신 이상

반 정규화(De-Normalization)

정규화 된 엔티티, 속성, 관계에 대해 성능 향상과 개발 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 과정

2) 물리 데이터

물리 데이터 모델링

논리모델을 적용하고자 하는 기술에 맞도록 상세화해가는 과정

참조무결성 제약조건

참조하는 외래키의 값은 항상 참조되는 릴레이션에 기본키로 존재해야한다.

- 제한(RESTRICT), 연쇄(CASCADE), 널 값(SET NULL)

인덱스

전체 데이터 검색 없이 필요한 정보에 대해 신속한 조회 가능

뷰

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

클러스터

데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법

파티션(Partition)의 종류

- 레인지(Range) 파티셔닝 : 연속적인 숫자나 날짜 기준

- 해시(Hash) 파티셔닝 : 파티션 키의 해시 함수 값
- 리스트(List) 파티셔닝 : 특정 파티션에 저장 될 데이터에 대한 명시적 제어 가능
- 컴포지트(Composite) 파티셔닝 : 레인지, 해시, 리스트 중 2개 이상의 파티셔닝 결합

3) 데이터베이스

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, 계속적인 변화, 동시 공용, 내용 참조

DBMS

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

DBMS 유형

- 키-값 DBMS
- 컬럼 기반 데이터 저장(Column Family Data Store)
- 문서 저장(Document Store)
- 그래프(Graph Store) : 시맨틱 웹과 온톨로지 분야

빅데이터

시스템, 서비스, 조직 등에서 주어진 비용, 시간 내에 처리가 가능한 수십 페타바이트 크기의 비정형 데이터

- HDFS : 대용량의 데이터의 집합을 처리하는 응용 프로그램에 적합하도록 설계된 하둡 분산 파일 시스템
- 맵 리듀스(Map Reduce) : 구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅 처리하기 위한 목적으로 제작해 2004년에 발표한 소프트 프레임 워크

NoSQL

전통적인 RDBMS와 다른 DBMS를 지칭하기 위한 용어, 데이터 저장에 고정된 테이블 스키마가 필요하지 않고 조인 연산을 사용할 수 없으며, 수평적으로 확장이 가능한 DBMS

NoSQL의 특성(BASE)

- Basically Available : 언제든지 데이터는 접근할 수 있어야 하는 속성
- Soft-State : 노드의 상태는 외부에서 전송된 정보를 통해 결정되는 속성
- Eventually Consistency : 일정 시간이 지나면 데이터의 일관성이 유지

시맨틱 웹(Semantic Web)

기계가 이해할 수 있는 온톨로지 형태로 표현하고 자동화된 기계가 처리하도록 하는 지능형 웹

온톨로지(Ontology)

실세계에 존재하는 모든 개념들과 개념들의 속성, 개념들 간의 관계 정보를 컴퓨터가 이해할 수 있도록 서술해 놓은 지식베이스

데이터 마이닝(Data Minning)

대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아내는 기술

데이터 마이닝 주요기법

- 분류 규칙(Classification) : 과거 데이터로부터 특성을 찾아내어 분류모형을 만들어 결과 값 예측
- 연관 규칙(Association) : 데이터 안에 존재하는 항목들 간의 종속관계를 찾아내는 기법
- 연속 규칙(Sequence) : 연관 규칙에 시간 관련 정보가 포함된 형태의 기법
- 데이터 군집화(Clustering) : 대상 레코드들을 유사한 특성을 지는 몇 개의 소그룹으로 분할하는 작업

4. 통합 구현

연계 메커니즘

응용 소프트웨어와 연계 대상 모듈 간의 데이터 연계 시 요구사항을 고려한 연계 방법과 주기를 설계하기 위한 메커니즘

주요 연계 기술

- 직접 연계
  - DB 링크, DB 연결, API, JDBC, 하이퍼 링크
- 간접 연계
  - 연계 솔루션(EAI) : 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해주는 솔루션, 어댑터 이용
  - Web Service/ESB : WSDL과 SOAP프로토콜을 이용한 시스템 간 연계
  - Socket : 소켓을 생성하여 포트를 할당하고, 클라이언트의 요청을 연결하여 통신

EAI(Enterprise Application Integration)

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해주는 솔루션

- 구성요소
  - EAI 플랫폼 : 이기종 시스템 간 애플리케이션 상호 운영
  - 어댑터 : 기업에서 자체적으로 개발한 애플리케이션을 연결하는 EAI의 핵심장치로 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
- 구축유형
    - 포인트 투 포인트 : 1:1 단순 통합 방법
    - 허브앤 스포크 : 단일한 접점의 허브 시스템을 통하여 데이터를 전송하는 중앙 집중식 방식
    - 메시지 버스 : 미들웨어를 두어 연계하는 통합 방식
    - 하이브리드 : 그룹 내는 허브 앤 스포크, 그룹 간에는 메시지 버스

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

- 느슨한 결합(Loosely Coupled) : 특정 서비스를 변경하더라도 연결된 다른 서비스에는 영향을 주지 않는 유연한 구조

웹 서비스 유형

- 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 정보 등 웹 서비스에 대한 상세정보가 기술된 XML 형식의 언어
- UDDI(Universal Description, Discovery and Integration) : WSDL을 등록하고 검색하기 위한 저장소로 공개적으로 접근, 검색이 가능한 레지스트리이자 표준

5. 인터페이스 구현

JSON(Javascript Object Notation)

속성-값 쌍 또는 “키-값 쌍”으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷,

AJAX에서 많이 사용되고 XML을 대체하는 주요 데이터 포맷

XML(Extensible Markup Language)

HTML의 단점을 보완한 인터넷 언어, SGML의 복잡한 단점을 개선한 특수한 목적을 갖는 마크업 언어

AJAX(Asynchronous Javascript And XML)

자바스크립트를 사용하여 웹 서버와 클라이언트 간 비동기적으로 XML 데이터를 교환하고 조작하기 위한 웹 기술, XMLHttpRequest 객체를 이용해 전체 페이지를 새로 로드하지 않고 필요한 부분만 로드 한다.

REST(Representational State Transfer)

- 웹과 같은 분산 하이퍼미디어 환경에서 자원의 존재/상태 정보를 표준화된 HTTP 메서드로 주고받는 웹 아키텍처 (리소스, 메서드, 메시지)
- HTTP URI를 통해 자원을 명시하고, HTTP 메서드(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 생성, 조회, 갱신, 삭제 등의 명령을 적용할 수 있는 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처이다.

데이터베이스 암호화 기법 (애플티하)

- API 방식 : 암호모듈(API)을 적용하는 애플리케이션 수정 방식
- Plug-In 방식 : 암·복호화 모듈이 DB 서버에 설치된 방식
- TDE 방식 : DBMS 커널이 자체적으로 암·복호화 기능 수행

- Hybrid 방식 : API + Plug-In

인터페이스 구현 검증 도구

- xUnit : 자바, C++, .Net 등 다양한 언어를 지원하는 단위테스트 프레임워크
- STAF : 서비스 호출, 컴포넌트 재사용 등 다양한 환경 지원하는 테스트 프레임워크
- FitNesse : 웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원
- NTAF : FitNess + STAF를 통합한 NHN(Naver)의 테스트 자동화 프레임워크
- Selenium : 다양한 브라우저 지원 및 개발언어를 지원, 테스트 스크립트 언어 학습할 필요 없음, 웹 애플리케이션 테스트 프레임워크
- watir : 루비 기반 웹 애플리케이션 테스트 프레임워크, 모든 언어 기반 웹/브라우저 호환성 테스팅 가능

인터페이스 감시 도구(APM; 성능 모니터링 도구)

- 스카우터(SCOUTER) : 애플리케이션에 대한 모니터링 및 DB Agent를 통해 오픈 소스 DB모니터링 기능, 인터페이스 감시 기능 제공
- 제니퍼(Jennifer) : 애플리케이션 개발부터 테스트, 오픈, 운영, 안정화까지 전 생애주기 단계 동안 성능을 모니터링하고 분석해주는 APM 소프트웨어

8. 서버프로그램 구현

개발 도구(빌구테형)

- 빌드 도구
- 구현 도구
- 테스트 도구
- 형상 관리 도구

서버 하드웨어 개발 환경

- 웹 서버 : 정적 콘텐츠(CSS, Javascript, Image)처리 / Apache 웹 서버
- 웹 애플리케이션 서버 : 동적 콘텐츠(Servlet, JSP) 처리 / Tomcat
- 데이터베이스 서버 : MySQL, Oracle
- 파일 서버 : HDD, SSD

소프트웨어 개발 환경

- 운영체제 : 하드웨어를 사용자가 편리하고 유용하게 사용하기 위한 소프트웨어
- 미들웨어 : 컴퓨터와 컴퓨터 간의 연결을 쉽고 안전하게 할 수 있게
- DBMS : 데이터베이스 관리

형상 관리

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

- 절차 (식통감기) //약술형
  - 형상 식별 : 형상 관리 대상 정의 및 식별
  - 형상 통제 : 형상 항목 버전 관리를 위해 변경 여부와 변경 활동 통제

    - 형상 감사 : 소프트웨어 베이스라인의 무결성 평가, 베이스라인 변경 시 요구사항과 일치하는지 검토
        - 베이스 라인 : 개발과정의 각 단계별 산출물에 대한 변화를 통제하는 시점의 기준
    - 형상 기록 : 형상 및 변경관리에 대한 각종 수행결과 기록

소프트웨어 형상 관리 도구

- 공유 폴더 방식 : 매일 개발이 완료된 파일은 약속된 위치의 공유 폴더에 복사
    - RCS : 소스 파일의 수정을 한 사람만으로 제한
    - SCCS
- 클라이언트 / 서버 방식 : 중앙에 버전 관리 시스템을 항시 동작
    - CVS : 다수 인원 동시에 운영체제 접근 가능
    - SVN : 하나의 서버에서 소스를 쉽고 유용하게 관리할 수 있게 해줌
    - Bitkeeper : SVN과 비슷, 대규모 프로젝트에서 빠른 속도 내도록 개발된 형상 관리 도구
- 분산 저장소 방식 : 로컬/원격 저장소로 분리되어 분산 저장
    - Git : commit, push

모듈

하나의 완전한 기능을 수행할 수 있는 독립된 실체

모듈화

소프트웨어의 성능을 향상시키거나 프로그램을 효율적으로 관리할 수 있도록 시스템을 분해하고 추상화하는 기법

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

응집도(Cohesion)

모듈 내부 구성요소간 연관 정도 (낮음[나쁜 품질]) → 높음[높은 품질])

- 우연적(Coincidental) : 모듈 내부의 각 구성 요소가 연관이 없을 경우
- 논리적(Logical) : 유사한 성격, 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우
- 시간적(Temporal) : 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우
- 절차적(Procedural) : 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우
- 통신적(Communication) : 동일한 입출력을 사용하여 다른 기능을 수행하는 활동들이 모여 있는 경우
- 순차적(Sequential) : 한 활동으로부터 나온 출력 값을 다른 활동이 사용할 경우
- 기능적(Functional) : 모듈 내부 모든 기능이 단일한 목적을 위해 수행되는 경우

결합도(Coupling)

모듈 내부가 아닌 외부 모듈과의 연관도 또는 모듈 간의 상호 의존성 (높음[나쁜 품질]) → 낮음[높은 품질])

- 내용(Content) : 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우
- 공통(Common) : 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 갱신하는 식으로 상호작용하는 경우
- 외부(External) : 두 개의 모듈이 외부에서 도입된 인터페이스를 공유할 경우
- 제어(Control) : 단순 처리할 대상인 값만 전달되는게 아니라 어떻게 처리를 해야 한다는 제어 요소가 전달되는 경우
- 스탬프(Stamp) : 인터페이스로 배열, 객체, 구조 등이 전달되는 경우
- 자료(Data) : 파라미터를 통해서만 상호작용이 일어나는 경우

공통 모듈 구현 절차

DTO/VO -> SQL -> DAO -> Service -> Controller -> View

- DAO : 특정 타입의 데이터베이스에 추상 인터페이스를 제공하는 객체
- DTO(Data Transfer Object) : 프로세스 사이에서 데이터를 전송하는 객체

- VO : 간단한 엔티티를 의미하는 작은 객체 가변 클래스인 DTO와 달리 고정 클래스를 가지는 객체

팬인(Fan-In) / 팬 아웃(Fan-Out)

시스템 복잡도를 최적화 하기 위해서는 팬인은 높게, 팬 아웃은 낮게

배치 프로그램

일련의 작업들을 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법

- 이벤트 배치 : 사전에 정의해 둔 조건 충족 시
- 온디맨드 배치 : 사용자의 요구가 있을 때마다
- 정기 배치 : 정해진 시점

배치 스케줄러

일괄 처리를 위해 주기적으로 발생하거나 반복적으로 발생하는 작업을 지원하는 도구

- 스프링 배치 : 스프링의 3대요소를 모두 사용할 수 있는 대용량 처리를 제공
- 쿼츠 스케줄러 :

Cron 표현식

- 리눅스/유닉스 : 분시일 월요연
- 쿼츠 : 초분시일 월요연

9. 소프트웨어 개발 보안 구축

SW 개발 보안의 3대 요소

- 기밀성(Confidentiality) : 인가되지 않은 개인 혹은 시스템 접근에 따른 정보 공개 및 노출을 차단하는 특성
- 무결성(Integrity) : 정당한 방법을 따르지 않고서는 데이터가 변경 될 수 없으며, 데이터의 정확성 및 완전성과 고의/악의로 변경되거나 훼손되지 않음을 보장하는 특성
- 가용성(Availability) : 권한을 가진 사용자나 애플리케이션이 원하는 서비스를 지속해서 사용할 수 있도록 보장하는 특성

DoS(Denial of Service) _공격자 컴퓨터 1대, 직접 공격

시스템을 악의적으로 공격해 해당 시스템의 자원을 부족하게 해 사용하지 못하게 하는 공격

DoS 공격 종류

- SYN 플러딩(Flooding) : 서버의 동시 가용 사용자 수를 SYN 패킷만 보내 점유하여 다른 사용자가 서버를 사용하지 못하게 하는 공격
- UDP 플러딩(Flooding) : 대량의 UDP패킷을 만들어 임의의 포트 번호로 전송하여 지속적으로 자원을 고갈시키는 공격
- 스머프(Smurf)/스머핑(Smurfing) : 출발지 주소를 공격 대상의 IP로 설정하여 네트워크 전체에게 ICMP Echo 패킷을 직접 브로드캐스팅하여 마비시킴
- 죽음의 핑(PoD; Ping of Death) : ICMP 패킷(Ping)을 정상적인 크기보다 아주 크게 만들어서 전송
- 랜드어택(Rand Attack) : 출발지 IP와 목적지 IP를 같은 패킷 주소로 만들어 보내서 수신자가 자기 자신에게 응답을 보내게 함
- 티어드롭(Tear Drop) : IP 패킷의 재조합 과정에서 잘못된 Fragment Offset 정보로 인해 수신 시스템이 문제를 발생하도록 만드는 공격
- 봉크(Bonk)/보잉크(Boink) : 시스템의 패킷 재전송과 재조립이 과부하를 유발하게 하는 공격기법

DDos(Distributed DoS)_공격자가 여러 대의 컴퓨터를 감염 시킴, 공격 지시

여러 대의 공격자를 분산 배치하여 동시에 동작하게 함으로써 특정 사이트 공격

DDoS 공격 도구

- Trinoo : 많은 소스로부터 통합된 UDP flood 서비스 거부 공격을 유발하는데 사용
- TFN(Tribe Flood Network) : Trinoo와 비슷한 분산 도구, 많은 소스에서 하나 혹은 여러 개의 목표 시스템에 대해 서비스 거부 공격
- Stacheldraht : 분산 서비스 거부 에이전트 역할

DRDoS(Distributed Refleection DoS)

공격자는 **출발지 IP를 공격대상 IP로 위조**하여 다수의 반사 서버로 요청 정보를 전송,

공격 대상자는 반사 서버로부터 다량의 응답을 받아서 서비스 거부(DoS)가 되는 공격이다.

애플리케이션 공격

- HTTP GET Flooding : 과도한 GET 메시지를 이용해 웹 서버의 과부하를 유발시키는 공격
- Slowloris(Slow HTTP Header DoS) : HTTP GET 메서드를 사용해 헤더의 최종 끝을 알리는 개행 문자열을 전송하지 않음
- RUDY(Slow HTTP POST DoS) : 요청 헤더의 Content-length를 비정상적으로 크게 설정하고 메시지 바디 부분을 매우 소량을 보내 계속 연결상태 유지시키는 공격(999999 설정 이후 1바이트씩 전송)
- Slow HTTP Read DoS : TCP 윈도 크기와 데이터 처리율을 감소시킨 상태에서(Zero Window Packet) 다수 HTTP 패킷을 지속적으로 전송
- Hulk DoS : 공격자가 공격대상 웹사이트 URL을 지속적으로 변경하면서 다량으로 GET 요청을 발생시키는 서비스 거부 공격
- Hash DoS : 조작된 많은 수의 파라미터를 POST방식으로 웹 서버로 전달하여 다수의 해시 충돌 발생시키는 공격

네트워크 공격

- 스니핑(Sniffing) : 직접 공격을 하지 않고 데이터만 몰래 들여다보는 수동적 공격
- 네트워크 스캐너(Scanner), 스니퍼(Sniffer) : 네트워크 하드웨어, 소프트웨어 구성의 취약점을 탐색하는 공격 도구
- 패스워드 크래킹(Password Cracking)
    - 사전 크래킹(Dictionary) : ID와 패스워드가 될 가능성이 있는 단어를 파일로 만들어 놓음
    - 무차별 크래킹(Brute): 패스워드로 사용될 수 있는 글자를 무작위로 패스워드 자리에 대입
    - 패스워드 하이브리드 공격 : 사전 + 무차별
    - 레인보우 테이블 공격 : 패스워드 별로 해시 값을 미리 생성해서 역으로 패스워드를 찾음
- IP 스푸핑 : 침입자가 인증된 컴퓨팅 시스템인 것처럼 속이기 위해서 본인의 패킷 헤더를 인증된 호스트의 IP로 위조하여 타깃에 전송
- ARP 스푸핑 : 공격자가 특정 호스트의 MAC 주소를 자신의 **MAC 주소로 위조**한 ARP Reply를 만들어 특정 호스트의 MAC 정보를 공격자의 MAC정보로 변경
- ICMP Redirect : 스니핑 시스템을 네트워크에 존재하는 또 다른 라우터라고 알림으로써 패킷의 흐름을 바꿈, Redirect 메시지를 공격자가 원하는 형태로 만들어서 공격
- 트로이 목마 : 겉보기에는 정상적인 프로그램으로 보이지만 실행하면 악성 코드를 실행하는 프로그램

시스템 보안 위협

- 버퍼 오버플로우(Buffer Overflow) : 메모리에 할당된 버퍼크기를 초과하는 양의 데이터를 입력해 공격
    - 유형 : 스택 버퍼 오버플로우, 힙 버퍼 오버플로우
    - 대응방안
        - 스택가드(Stack guard) : 버퍼 오버플로우 발생 시 카나리 값을 체크
        - 스택쉴드(Stack Shield) : 함수 시작 시 복귀 주소를 Global RET에 저장해 두고 함수 종료 시 저장된 값과 스택의 RET값을 비교해서 다를 경우 프로그램 중단

        - ASLR(Address Space Layout Randomization) : 주소 공간 배치를 난수화, 리눅스에서 설정 가능
- 백도어 : 어떤 제품이나 컴퓨터 시스템, 암호시스템, 알고리즘에서 정상적인 인증 절차를 우회하는 기법
- 주요 시스템 보안 공격기법
    - 포맷 스트링 공격 : 외부로부터 입력된 값을 검증하지 않고 그대로 사용하는 경우 발생하는 취약점 공격법
    - 레이스 컨디션 공격 : 실행되는 프로세스가 임시파일을 만드는 경우 악의적인 프로그램을 통해 그 프로세스의 실행 중에 끼어들어 임시파일을 심볼릭 링크하는 공격기법
    - 키로거 공격 : 사용자의 키보드 움직임을 탐지해서 개인의 중요한 정보를 몰래 빼가는 해킹공격
    - 루트킷 : 시스템 침입 후 사실을 숨긴 채 차후의 침입을 위해 불법적인 해킹기능을 제공하는 프로그램(트로이 목마, 백도어..)의 모음

보안 관련 용어

- 스피어피싱(Spear Phishing) : 메일을 이용한 공격
- 스미싱(Smishing) : 문자메시지를 이용한 공격
- 큐싱(Qushing) : QR코드
- APT 공격(Advanced Persistent Threat) : 특정 타깃을 목표로 하여 다양한 수단을 통해 지속적이고 지능적인 맞춤형 공격기법
- 공급망 공격(Supply Chain Attack) : 소프트웨어 개발사의 네트워크에 침투하여 소스 코드를 수정하여 악의적인 코드를 삽입해 공격
- 제로데이 공격(Zero Day Attack) : 보안 취약점이 발견되어 널리 공표되기 전에 해당 취약점을 악용하여 공격
- 웜 : 스스로를 복제하여 네트워크로 전파하는 악성 소프트웨어 컴퓨터 프로그램
- 악성 봇(Malicious Bot) : 스스로 실행되지 못하고 해커에 의해 제어, 실행
- 사이버 킬체인 : 7단계 프로세스별 APT 공격 방어 분석모델
- 랜섬웨어 : 몸값을 요구하는 악성 소프트웨어

- 이블 트윈 공격 : 무선 Wifi 피싱기법
- 난독화(Obfuscation) : 프로그램 코드의 일부 또는 전체를 변경하여 역공학에 대비
- Tripwire : 크래커가 침입했을 때 알 수 있게 분석하는 도구, 데이터베이스 차이점 체크
- Ping : 원격 호스트가 정상적으로 운영되고 있는지를 확인하는 진단 목적으로 사용하는 명령어
- Tcpdump : 네트워크 인터페이스를 거치는 패킷의 내용을 출력해주는 프로그램, 모든 패킷 내용 도청할 수 있음

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): 신분에 근거하여 객체에 대한 접근을 제한하는 방법
- 강제적 접근 통제(MAC): 주체가 갖는 접근 허가 권한에 근거하여 객체에 대한 접근을 제한하는 방법
- 역할 기반 접근 통제(RBAC): 중앙 관리자가 조직 내 맡은 역할에 기초하여 자원에 대한 접근을 제한하는 방법

인증 기술 유형

- 지식기반 : ID/패스워드
- 소지기반 : 공인인증서
- 생체기반 : 얼굴, 지문
- 특징기반 : 발걸음, 몸짓

접근 통제 보호 모델

- 벨-라파둘라 모델 : 미 국방부 지원 모델, **기**밀성 강조
  - 벨**기**노라다(No Write Down/No Read Up) : 보안수준이 높은 주체는 보안 수준이 낮은 객체에 기록하면 안 됨
- 비바 모델 : **무**결성 보장
  - 비**무**노라업(No Write Up/No Read Down): 낮은 등급의 주체는 상위 등급의 객체를 수정 할 수 없음

암호 알고리즘

암호 알고리즘 방식

알고리즘 방식	알고리즘 방식 종류		기법
양방향 방식	대칭 키 암호 방식	블록 암호 방식	DES, SEED, AES, ARIA, IDEA [2020년 3회]
		스트림 암호 방식	RC4, LFSR
	비대칭 키 암호 방식 (=공개키 암호 방식)		RSA, ECC, ELGamal, 디피-헬만(Diffie-Hellman)
일방향 암호 방식 (해시 암호 방식)	MAC		HMAC, NMAC
	MDC		ND5, SHA

양방향 (대비 비공)

- 대칭키(비공개키)
  - 암호화 = 복호화
  - **블록 암호 방식** : 고정 길이의 블록을 암호화하여 반복하는 알고리즘
    - DES : 블록 크기 64bit, 키 길이 56bit인 페이스텔 구조, 미국 연방 표준국(NIST) 암호화 알고리즘
    - AES : DES를 대체, 3 DES의 성능문제를 극복하기 위해 개발, 미국 표준 기술 연구소(NIST)

        - SEED : 한국인터넷진흥원(KISA) 개발
        - ARIA : 경량 환경 및 하드웨어에서의 효율성 향상을 위해 개발,국가정보원 + 산학연구협회가 개발
        - IDEA : 스위스 연방기술 기관에서 개발
    - **스트림 암호 방식** : 매우 긴 주기의 난수열을 발생시켜 평문과 더불어 암호문을 생성하는 방식
        - LFSR : 선형 되먹임 시프트 레지스터
        - RC4
- 비대칭키(공개키)
    - 암호화 ≠ 복호화
        - 디피-헬만 : 최초의 공개키 알고리즘, 이산 대수
        - RSA : 3명의 MIT 수학교수가 고안, 소인수 분해 수학적 알고리즘
        - ECC : RSA 암호 방식 대안, 타원 곡선 암호(ECC)
        - ElGamal : 이산대수 계산이 어려운 문제를 기본원리로 함

일방향

복호화 불가능

- **해시 암호 방식** : MAC(키 사용), MDC(키 사용X)
    - MD5 : MD4개선, 프로그램이나 파일의 무결성 검사에 사용
    - SHA-1 : NSA에서 미 정부 표준으로 지정, DSA에서 사용
    - SHA-256/384/512 : 256비트의 해시값을 생성하는 해시함수
    - HAS-160 : 국내 표준 서명 알고리즘(KCDSA)를 위해 개발된 해시 함수, MD5장점 +SHA-1장점

IPSec(Internet Protocol Security)

IP계층에서 무결성과 인증을 보장하는 인증 헤더와 기밀성을 보장하는 암호화를 이용한 IP 보안 프로토콜

- 인증, 암호화, 키 관리 프로토콜로 구성

SSL(Secure Socket Layer)/TLS(Transport Layer Security)

전송계층과 응용계층 사이에서 클라이언트와 서버 간의 웹 데이터 암호화, 상호 인증 및 전송 시 데이터 무결성을 보장하는 보안 프로토콜

S-HTTP

웹 상에서 네트워크 트래픽을 암호화하는 주요 방법, 클라이언트와 서버 간 전송되는 모든 메시지를 각각 암호화해 전송하는 기술

개인정보보호 관련 법령

개인정보 보호법, 정보통신망법, 신용정보법

- 민감 정보 : 주체의 사생활을 현저하게 침해할 수 있는 정보(유전자 검사정보)
- 고유 식별정보 : 개인을 고유하게 구별하기 위해 부여된 식별 정보(주민번호)

입력 데이터 검증 및 표현 취약점

- XSS(Cross Site Script) : 검증되지 않은 외부 입력 데이터가 포함된 웹페이지를 사용자가 열람할 때 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) : 악의적인 SQL 구문을 삽입하고 실행시켜 정보를 열람, 조작할 수 있는 취약점 공격법

네트워크 보안 솔루션

- 방화벽(Firewall) : 기업 내부, 외부 간 트래픽을 모니터링 하여 시스템의 접근을 허용하거나 차단하는 시스템
- 웹 방화벽 (WAF; Web Application Firewall): 웹 어플리케이션 보안에 특화된 보안장비
- 네트워크 접근 제어(NAC; Network Access Control) : 단말기가 내부 네트워크에 접속을 시도할 때 이를 제어하고 통제하는 기능을 제공하는 솔루션
- 침입 탐지 시스템(IDS; Intrusion Detection System) : 네트워크에 발생하는 이벤트를 모니터링하고, 비인가 사용자의 침입을 실시간으로 탐지하는 시스템

- 침입 방지 시스템(IPS; Intrusion Prevention System) : 네트워크에 대한 공격이나 침입을 실시간적으로 차단하는 시스템
- 무선 침입 방지 시스템(WIPS; Wireless Intrusion Prevention System) : 인가되지 않은 무선 단말기의 접속을 자동 탐지 및 차단하고 보안에 취약한 무선 공유기를 탐지
- 통합 보안 시스템(UTM; Unified Threat Management) : 다양한 보안 장비의 기능을 하나의 ㅈ ㅏ ㅇ비로 통합하여 제공하는 시스템
- 가상사설망(VPN; Virtual Private Network) : 인터넷과 같은 공중망에 인증, 암호화, 터널링 기술을 활용해 마치 전용망을 사용하는 효과를 가지는 보안 솔루션

시스템 보안 솔루션

- 스팸 차단 솔루션(Anti-Spam Solution) : 메일 서버 앞단에 위치하여 프록시(Proxy) 메일 서버로 동작
- 보안 운영체제(Secure OS) : 컴퓨터 운영체제의 커널에 보안 기능을 추가한 솔루션

콘텐츠 유출 방지 솔루션

- 데이터 유출 방지(DLP; Data Loss Prevention) : 조직 내부의 중요 자료가 외부로 빠져나가는 것을 탐지하고 차단
- 디지털 저작권 관리(DRM; Digital Right Management) : 디지털 저작물에 대한 보호와 관리 솔루션

비즈니스 연속성 계획 (BCP; Busniess Continuity Plan)

각종 재해, 장애, 재난으로부터 위기관리를 기반으로 재해복구, 업무복구 및 재개, 비상계획 등을 통해 비즈니스 연속성을 보장하는 체계

- BIA(Business Impact Analysis) : 장애나 재해로 인한 운영상의 주요 손실을 볼 것을 가정하여 비즈니스 영향 분석
- RTO(Recovery Time Objective) : 업무중단 시점부터 업무가 복구되어 다시 가동될 때까지의 시간
- RPO(Recovery Point Objective) : 업무중단 시점부터 데이터가 복구되어 다시 정상 가동될 때 데이터의 손실 허용 시점

- DRP(Disaster Recovery Plan) : 재난으로 장기간에 걸쳐 시설의 운영이 불가능한 경우를 대비한 재난 복구 계획
- DRS(Disaster Recovery System) : 재해 복구 센터

DRS의 유형

- Mirror Site : 재해 발생 시 복구까지의 소요 시간(RTO)은 즉시
- Hot Site : 4시간 이내
- Warm Site : 수일 ~ 수주
- Cold Site : 수주 ~ 수개월

10. 애플리케이션 테스트

소프트웨어 테스트 원리

- 테스팅은 결함이 존재함을 밝히는것
- 완벽한 테스팅은 불가능
- 개발 초기에 테스팅 시작
    - 요르돈의 법칙(눈덩이 법칙) : 개발 초기에 테스팅 하지 않으면 비용이 커진다.
- 결함 집중
    - 파레토 법칙(Pareto Principle) : 오류의 80%는 전체 모듈의 20% 안에서 발견된다.
- 살충제 패러독스(Pesticide Paradox) : 동일한 테스트 케이스에 의한 반복적 테스트는 새로운 버그를 찾지 못함
- 테스팅은 정황에 의존적 : 소프트웨어의 성격에 맞게 테스트 실시
- 오류-부재의 궤변 : 요구사항을 충족시키지 못하다면, 결함이 없다고해도 품질이 높다고 볼 수 없음

테스트 시각에 따른 분류

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

테스트 목적에 따른 분류(회안성 구회병)

- 회복 테스트(Recovery) : 시스템에 고의로 실패를 유도하고, 시스템의 정상적 복귀 여부를 테스트
- 안전 테스트(Security) : 소스 내 보안적인 결함을 미리 점검하는 테스트
- 성능 테스트(Performance) : 응답 시간, 반응 속도, 처리량 등을 측정하는 테스트
- 구조 테스트(Structure) : 시스템의 내부 논리 경로, 소스 코드의 복잡도를 테스트

- 회귀 테스트(Regression) : 오류제거와 수정에 의해 새로 유입된 오류가 없는 지 확인하는 일종의 반복 테스트 기법
- 병행 테스트(Parallel) : 변경된 시스템과 기존 시스템에 동일한 데이터 입력 후 결과 비교

성능 테스트 상세 유형(부스스내)

- 부하(Load) 테스트 : 시스템에 부하를 계속 증가시키면서 시스템의 임계점을 찾음
- 스트레스(Stress) 테스트 : 임계점 이상의 부하를 가해 비정상적인 상황에서의 처리를 테스트
- 스파이크(Spike) 테스트 : 짧은 시간에 사용자가 몰릴 때 시스템의 반응 측정 테스트
- 내구성(Endurance) 테스트 : 오랜 시간 동안 시스템에 높은 부하를 가해 테스트

테스트 종류에 따른 분류

- 명세 기반 테스트
- 구조 기반 테스트
- 경험 기반 테스트

정적 테스트

- 정적 분석 : 자동화된 도구를 이용하여 산출물의 결함을 검출하거나 복잡도를 측정(도구)
- 리뷰 : 소프트웨어의 다양한 산출물에 존재하는 결함을 검출하거나 프로젝트의 진행 상황을 점검하기 위한 활동으로 전문가가 수행(사람)
    - 인스펙션(동료검토)
        - 형식적 검토 기법
        - 저작자 외의 다른 전문가 또는 팀이 검사하여 문제를 식별하고 문제에 대한 올바른 해결을 찾아냄
    - 워크스루
        - 비형식적 검토 기법
        - 검토 자료를 회의전에 배포해서 사전 검토한 후 짧은 시간동안 회의를 진행하는 형태

동적 테스트

화이트박스 테스트(구조 기반 테스트)

각 응용프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트

- 구문(문장) 커버리지(Statement Coverage) : 프로그램 내의 모든 명령문을 적어도 한 번 수행
- 결정(분기) 커버리지(Decision/Branch Coverage) : 결정 포인트 내의 전체 조건식이 적어도 한 번은 참과 거짓의 결과를 수행 (= 선택 커버리지, 분기 커버리지)
- 조건 커버리지(Condition Coverage) : 결정 포인트 내의 각 개별 조건식이 적어도 한 번은 참과 거짓의 결과가 되도록 수행
- 조건/결정 커버리지(Condition/Decision Coverage) : 전체 조건식 + 개별 조건식
- 변경 조건/결정 커버리지(Modified Condition/Decision Coverage) : 개별 조건식이 다른 개별 조건식에 영향을 받지 않고 전체 조건식에 독립적으로 영향을 주도록 함
- 다중 조건 커버리지(Multiple Coverage) : 결정 조건 내 모든 개별 조건식의 모든 가능한 조합을 100% 보장
- 기본 경로 커버리지(Base Path Coverage) : 수행 가능한 모든 경로를 테스트
    - 맥케이브 복잡도 : 간선 수 – 노드 수 + 2
- 제어 흐름 테스트(Control flow) : 프로그램 제어 구조를 그래프 형태로 나타내어 내부 로직 테스트
- 데이터 흐름 테스트(Date flow) : 제어 흐름 그래프에 데이터 사용현황 추가

블랙박스 테스트(명세 기반 테스트)

외부 사용자의 요구사항 명세를 보면서 수행하는 테스트

- 동등분할 테스트(Equivalence Partitioning) : 입력 데이터의 영역을 유사한 도메인별로 유효값/무효값을 그룹핑하여 대푯값 테스트 케이스를 도출하여 테스트
- 경곗값 분석 테스트(Boundary Value Analysis) : 최솟값 바로 위, 최대치 바로 아래 등 입력값의 극한 한계를 테스트하는 기법

- 결정 테이블 테스트(Decision Table) : 요구사항의 논리와 발생조건을 테이블 형태로 나열하여, 조건과 행위를 모두 조합하여 테스트
- 상태 전이 테스트(State transition) : 이벤트에 의해 어느 한 상태에서 다른 상태로 전이되는 경우의 수를 수행하는 테스트
- 유스케이스 테스트(Use Case) : 프로세스 흐름을 기반으로 테스트케이스를 명세화하여 수행하는 테스트
- 분류 트리 테스트(Classification Tree) : SW의 일부 또는 전체를 트리구조로 분석 및 표현하여 테스트 케이스 설계해 테스트
- 페어와이즈 테스트(Pairwise) : 테스트 데이터 값들 간에 최소한 한 번씩을 조합하는 방식
- 원인-결과 그래프 테스트(Cause-Effect Graphing) : 그래프를 활용해 입력 데이터 간의 관계 및 출력에 미치는 영향을 분석
- 비교 테스트(Comparison) : 여러 버전의 프로그램에 같은 입력값을 넣어 비교해 테스트

경험 기반 테스트

- 탐색적 테스트(Exploratory Test): 테스트 스크립트를 문서로 작성하지 않고 경험에 바탕을 두고 탐색적으로 기능을 수행해 보면서 테스트 하는 기법
- 오류 추정(Error Guessing): 개발자가 범할 수 있는 실수를 추정하고 이에 따른 결함이 검출되도록 테스트 케이스를 설계하여 테스트
- 체크리스트(Checklist): 테스트 할 내용과 경험을 분류하여 나열하고 하나씩 확인
- 특성테스트(Characeristics Test): 품질모델에 있는 품질특성을 염두에 두고 이를 근간으로 테스트 케이스 설계하고 테스트(ISO/IEC 9126-2 활용)

테스트 오라클

테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참값을 입력하여 비교하는 기법

- 참(True) 오라클 : 모든 입력값에 대해 기대하는 결과를 생성함으로써 발생된 오류를 모두 검출
- 샘플링(Sampling) 오라클 : 특정한 몇 개의 입력값에 대해서만 기대하는 결과를 제공

- 휴리스틱(Heuristic) 오라클 : 샘플링 오라클을 개선하고 나머지 값들에 대해서는 휴리스틱(추정)으로 처리
- 일관성 검사(Consistent) 오라클 : 애플리케이션 변경이 있을 때, 수행 전과 후의 결괏값이 동일한지 확인

테스트 레벨 종류

- 단위(Unit) 테스트 : 구현이 진행되면서 수행하는 테스트, 모듈 및 컴포넌트 등을 테스트
- 통합(Integration) 테스트 : 모듈 간 인터페이스 관련 테스트
- 시스템(System) 테스트 : 단위 시스템 기능이 시스템에서 정상 수행 되는지를 검증하는 테스트(기능적 요구사항/비기능적 요구사항)
- 인수(Acceptance) 테스트
  - 알파 테스트 : 사용자가 개발자 환경에서 수행하는 테스트
  - 베타 테스트 : 실제 환경에서 일정 사용자에게 소프트웨어를 사용하게하고 피드백을 받는 테스트

단위 테스트

- 목(Mock)객체 : 객체지향 프로그램에서 독립적인 컴포넌트 테스트를 위해서 스텁의 객체지향 버전인 목 객체가 필요함
- 목 객체 유형
  - 더미 개체 : 객체만 필요하고 기능까지는 필요하지 않은 경우
  - 테스트 스텁 : 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구
  - 테스트 드라이버 : 테스트 대상 하위 모듈을 호출, 파라미터 전달, 모듈 테스트 수행 후 결과 도출
  - 테스트 스파이 : 테스트 대상 클래스와 협력하는 클래스
  - 가짜 객체 : 실체 협력 클래스의 기능을 대체해야 할 경우 사용

통합 테스트

- 비점증적인 방식
  - 빅뱅 방식 : 모든 모듈을 동시에 통합 후 테스트
- 점증적인 방식
  - 하향식 테스트 : 테스트 스텁 필요
  - 상향식 테스트 : 테스트 드라이버 필요
  - 샌드위치 테스트 : 상향식 + 하향식 테스트, 병렬 테스트 가능

테스트 자동화 도구 //단답형

- 정적 분석 도구(Static Analysis Tools) : 만들어진 애플리케이션을 실행하지 않고 분석하는 도구, 남은 결함을 발견하기 위하여 사용
- 테스트 실행 도구(Test Execution Tools) : 작성된 스크립트를 실행
- 성능 테스트 도구(Performance Test Tools) : 가상의 사용자를 생성하고 테스트를 수행
- 테스트 통제 도구(Test Control Tools) : 테스트 관리, 형상 관리, 결함 추적/관리 도구

테스트 하네스

애플리케이션 컴포넌트 및 모듈을 테스트하는 환경의 일부분으로, 테스트를 지원하기 위한 코드와 데이터를 말하며, 단위 또는 모듈 테스트에 사용하기 위해 코드 개발자가 작성한다.

- 테스트 드라이버
- 테스트 스텁
- 테스트 슈트 : 테스트 케이스 집합
- 테스트 케이스 : 입력값, 실행 조건, 기대 결과 등의 집합
- 테스트 스크립트 : 자동화된 테스트 실행 절차에 대한 명세
- 목 오브젝트 : 사용자의 행위를 조건부로 사전 입력해 두면, 그 상황에 예정된 행위 수행

결함 분석 방법

- 구체화(Specification) : 결함을 발생시킨 입력값, 테스트 절차, 환경을 명확히 파악

- 고립화(Isolation) : 어떤 요소가 결함 발생에 영향을 미치는지 분석
- 일반화(Generalization) : 결함 발생에 영향을 주는 요소를 최대한 일반화 시키는 방법

결함 심각도(치주 보경단)

- 치명적(Critical) 결함 : 기능이나 제품의 테스트를 완전히 방해, 데이터 손실, 시스템 충돌
- 주요(Major) 결함 : 기능이 기대와 다르게 동작
- 보통(Normal) 결함 : 일부 기능 부자연스러움, 사소한 기능 오작동
- 경미한(Minor) 결함 : 사용상의 불편함 유발, UI 잘림
- 단순(Simple) 결함 : 사소한 버그, 미관상 좋지 않음

결함 우선순위

발생한 결함이 얼마나 빠르게 처리되어야 하는지

- 결정적(Critical) - 높음(High) - 보통(Normal) - 낮음(Low)

애플리케이션 성능 측정 지표

- 처리량(Throughput) : 주어진 시간에 처리할 수 있는 트랜잭션의 수
- 응답 시간(Response Time) : 메뉴 클릭 시 해당 메뉴가 나타나기까지 걸리는 시간
- 경과 시간(Turnaround Time) : 사용자가 요구를 입력한 시점부터 트랜잭션을 처리 후 그 결과의 출력이 완료할 때까지 걸리는 시간
- 자원 사용률(Resource Usage) : CPU 사용량, 메모리 사용량, 네트워크 사용량

데이터베이스 관련 성능 저하 원인

- 데이터베이스 락(DB Lock) : 대량의 데이터 조회, 과도한 업데이트 시 발생하는 현상
- 불필요한 데이터베이스 패치(DB Fetch) : 대량의 데이터 요청이 들어올 경우 응답시간 저하 현상 발생
- 연결 누수 (Connection Leak): DB연결과 관련한 JDBC 객체를 사용 후 종료하지 않을 경우

- 부적절한 커넥션 풀 크기(Connection Pool Size) : 너무 작거나 크게 설정한 경우

베드 코드

다른 개발자가 로직을 이해하기 어렵게 작성된 코드

- 외계인 코드 : 아주 오래되거나 참고문서 또는 개발자가 없어 유지보수 작업이 어려운 코드
- 스파게티 코드 : 스파게티처럼 코드가 복잡하게 얽힘
- 알 수 없는 변수명
- 로직 중복

클린 코드

잘 작성되어 가독성 높고, 단순하며, 의존성을 줄이고, 중복을 최소화해 잘 정리된 코드

- 코드 작성원리 (가단의 중추) : 가독성, 단순성, 의존성 최소, 중복성 제거, 추상화
- 느슨한 결합 : 인터페이스 클래스를 이용하여, 클래스 간의 결합도(의존성) 최소화

소스 코드 품질분석 도구

- 정적 분석도구
  - pmd : 자바 및 타언어 소스 코드에 대한 버그, 데드코드 분석
  - cppcheck : C/C++ 코드에 대한 메모리 누수, 오버플로우 등 문제 분석
  - checkstyle : 자바 코드에 대한 코딩 표준 검사 도구
- 동적 분석도구
  - Avalanche : Valgrind , STP 기반 소프트웨어 에러 및 취약점 동적 분석 도구
  - Valgrind : 자동화된 메모리 및 스레드 결함 발견 분석 도구

리팩토링

기능을 변경하지 않고 복잡한 소스 코드를 수정, 보완하여 가용성 및 가독성을 높이는 기법

- 목적(유연산품)

- 유지보수성 향상
- 유연한 시스템
- 생산성 향상
- 품질향상

11. 응용 SW 기초 기술 활용

운영체제

사용자가 컴퓨터 하드웨어를 쉽게 사용할 수 있도록 인터페이스를 제공해주는 소프트웨어

- 특징 : 편리성 제공, 인터페이스 기능, 스케줄링, 자원 관리, 제어 기능
- 운영체제 = 커널 + 쉘
  - 커널 : 하드웨어 관련 내부적인 역할
  - 쉘 : 운영체계의 가장 바깥부분에서 사용자 명령에 대한 처리
- 종류 : 윈도즈, 유닉스, 리눅스, 맥, 안드로이드
- 윈도즈 특징
  - GUI 제공
  - 선점형 멀티태스킹 방식 제공
  - 자동감지 기능 제공(Plug and Play)
  - OLE 사용
- 유닉스 특징
  - 대화식
  - 다중 작업 기능
  - 다중 사용자 기능
  - 이식성 : 90% 이상 C언어로 구현
  - 계층적 트리 구조 파일 시스템 제공

리눅스/유닉스 기본 명령어

- chmod : 특정 파일 또는 디렉토리의 퍼미션 수정 명령어
  - 기호

        - 대상 : u, g, o, a
        - 연산자 : 추가+,제거 -, 지정=
        - 접근권한 : r, w, x 실행
        - ex) chmod go-w yoom.c : yoom.c의 group,others 에 write권한 제거
    - 숫자
        - r : 4, w: 2, x: 1
        - ex) chmod 641 yoom.c : yoom.c의 user에 rw, group에 r, others에 x
- chown : 파일이나 디렉토리의 소유자, 소유 그룹 명령어

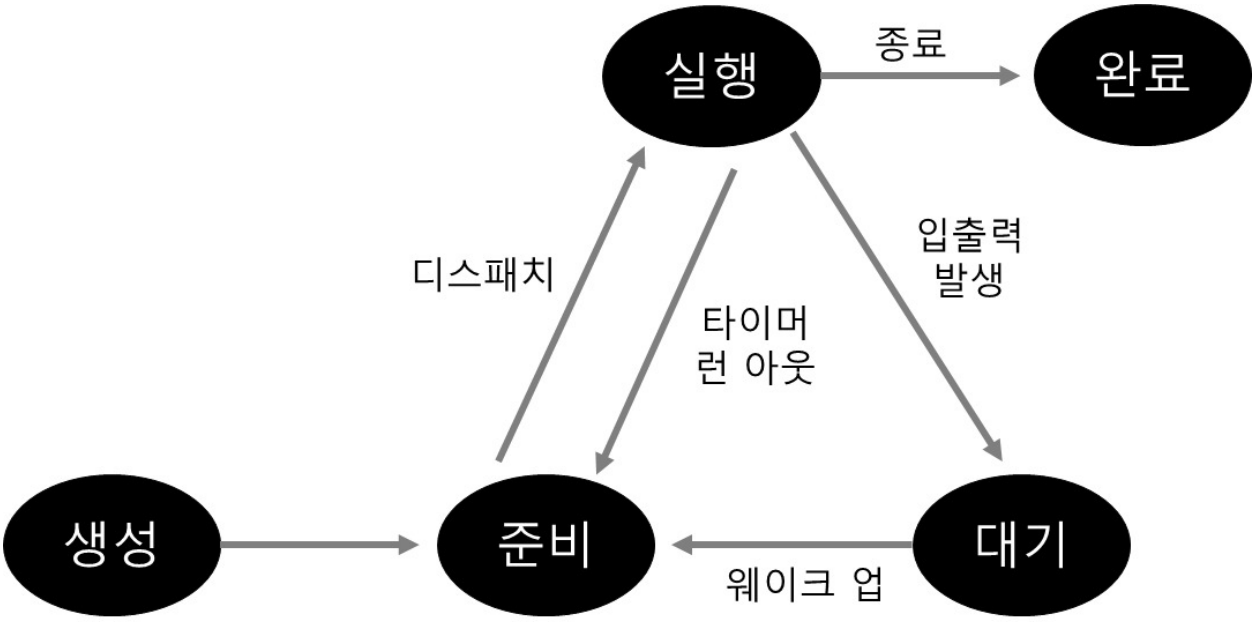
메모리 관리 기법

- 반입 기법 : 메모리로 적재 시기 결정
- 배치 기법 : 메모리로 적재 위치 결정
- 할당 기법 : 메모리로 적재 방법 결정
- 교체 기법 : 메모리 교체 대상 결정

메모리 배치 기법

- 최초 적합(First-fit) : 가용 공간 중 첫 번째 분할에 할당
- 최적 적합(Best-fit) : 가장 크기가 비슷한 공간에 할당
- 최악 적합(Worst-fit) : 가장 큰 공간에 할당

프로세스 상태 전이(생준실대완)



프로세스 스케줄링

- 선점형 : 하나의 프로세스가 CPU를 차지하고 있을 때, 우선순위가 높은 다른 프로세스가 현재 프로세스를 중단시키고 CPU를 점유하는 스케줄링 방식
  - 라운드 로빈(Round Robin) : 같은 크기의 CPU 할당
  - SRT(Shortest Remaining Time First) : 가장 짧은 시간이 소요되는 프로세스를 먼저 수행하고, 남은 처리시간이 더 짧다고 판단되는 프로세스가 준비 큐에 생기면 언제라도 프로세스가 선점됨
  - 다단계 큐(Multi Level Queue) : 여러 개의 큐를 이용하여 상위단계 작업에 의한 하위 단계 작업이 선점
  - 다단계 피드백 큐(Multi Level Feedback Queue): 큐마다 서로 다른 CPU시간 할당량을 부여, FIFO+라운드 로빈 스케줄링 기법 혼합
- 비선점형 : 한 프로세스가 CPU를 할당 받으면, 작업 종료후 다시 반환되기까지 다른 프로세스가 점유 불가능한 방식
  - 우선순위(Priority): 프로세스별 우선순위에 따라 CPU할당
  - 기한부(Deadline) : 작업들이 명시된 기한 내에 완료되도록 계획
  - FCFS(=FIFO) : 프로세스가 대기 큐에 도착한 순서에 따라 CPU 할당

    - SJF(Short Job First) : 프로세스가 도착하는 시점에 따라 그 당시 가장 작은 서비스 시간을 갖는 프로세스가 종료 시까지 자원 점유, 기아 현상 발생
    - HRN(Highest Response Ratio Next) : 대기 중인 프로세스 중 현재 응답률이 가장 높은 것을 선택, 기아현상 최소화 기법
        - 우선순위 = (대기시간+ 서비스시간) / 서비스시간
- 반환시간 = 종료시간 - 도착시간
- 대기시간 = 반환시간 - 서비스 시간

가상화(Virtualization)

물리적인 리소스들을 사용자에게 하나로 또는 여러 개로 보이게 하는 기술

이를 통해 서버의 가동률을 60~70% 이상으로 올릴 수 있다.

클라우드 컴퓨팅

인터넷의 서버를 통해 IT관련 서비스를 한 번에 사용할 수 있는 컴퓨팅 환경

- 인프라형 IaaS : 서버, 스토리지 같은 시스템 자원을 클라우드로 제공하는 서비스
- 플랫폼형 PaaS : 애플리케이션을 개발, 실행, 관리할 수 있게 하는 플랫폼을 제공하는 서비스
- 소프트웨어형 SaaS : 클라이언트를 통해 접속하여 소프트웨어를 서비스 형태로 이용하는 서비스

프로토콜

서로 다른 시스템에 있는 두 개체간의 데이터 교환을 원활히 하기 위한 일련의 통신규약

- 구문 : 데이터 형식, 코딩, 신호 레벨등의 규정
- 의미 : 조정, 에러처리를 위한 규정
- 타이밍 : 속도 조절, 순서 관리 규정

네트워크 프로토콜

컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고 받는 양식과 규칙의 체계

OSI 7계층

- **계층 1 - 물리계층 Physical Layer**
  - 0과 1의 비트 정보를 회선에 보내기 위한 신호 변환
  - 프로토콜
    - RS-232
  - 전송 단위 : 비트
  - 장비 : 허브, 리피터
- **계층 2 - 데이터 링크 계층 Data Link Layer**
  - 링크의 설정, 유지, 종료 담당 및 노드 간의 회선제어, 흐름제어, 오류제어
  - 프로토콜
    - HDLC : 점대점 방식이나 다중방식 통신에 사용
    - PPP : **두 통신 노드 간의 직접적인 연결**
  - 전송 단위 : 프레임
  - 장비 : 스위치, 브리지
- **계층 3 - 네트워크 계층**
  - 다양한 길이의 패킷을 네트워크들을 통해 전달하고 전송 계층이 요구하는 서비스 품질을 위한 수단을 제공하는 계층
  - 프로토콜
    - IP : 송수신 간의 패킷 단위, 정보를 주고받는 데 사용하는 통신 프로토콜
    - ARP : IP네트워크 상에서 MAC 주소를 알기 위해서 사용, IP 주소를 MAC 주소로 변환
    - RARP : MAC 주소는 알지만 IP주소를 모르는 경우 서버로부터 IP주소를 요청하기 위해 사용
    - ICMP: 수신지 도달 불가 메시지를 통지하는 데 사용

        - IGMP: 화상회의, IPTV에서 활용되는 프로토콜
        - 라우팅 프로토콜 : 데이터 전송을 위해 목적지까지 갓 수 있는 여러 경로 중 최적의 경로를 설정해주는 상호 통신 규약
            - RIP : AS(자율 시스템)내에서 사용하는 거리벡터 알고리즘에 기초하여 개발된 내부 라우팅 프로토콜, 최초 라우팅 프로토콜, 벨만-포드 알고리즘, 15홉 제한, IGRP
            - OSPF : RIP의 단점 개선, 최단 경로를 찾는 프로토콜, 다익스트라 알고리즘, 홉 제한 없음, ELGRP
            - BGF : 자치 시스템(AS)간 경로 정보를 교환, 링크 상태 알고리즘 사용
            - 라우팅 알고리즘
                - 거리 벡터 알고리즘: 인접 라우터와 정보를 공유하여 목적지까지의 거라와 방향을 결정하는 알고리즘, 벨만포드 사용
                - 링크 상태 알고리즘: 링크상태 정보를 모든 라우터에 전달하여 최단경로 트리를 구성하는 알고리즘, 다익스트라 알고리즘사용
    - 장비 : 라우터, L3스위치
- **계층 4 - 전송 계층 transport layer**
    - 상위 계층들이 데이터 전달의 유효성이나 효율성을 생각하지 않게 해주면서 종단간의 사용자들에게 신뢰성 있는 데이터를 전달하는 계층, 오류 제어 방식 사용
    - 프로토콜
        - TCP (신연흐혼): 옥텟을 안정적이고, 순서대로 에러없이 교환할 수 있게 해줌
            - 신뢰성, 연결성, 흐름제어, 혼잡제어
        - UDP : 비연결성, 비신뢰성, 순서화되지 않은 데이터그램 서비스 제공
    - 전송 단위 : 세그먼트
    - 장비 : L4스위치
- **계층 5 - 세션 계층**
    - 프로세스들의 논리적인 연결, 응용 프로그램 간의 대화를 유지하기 위한 구조 제공, 연결이 끊어지지 않도록 유지 시켜주는 역할 수행
- **계층 6 - 표현 계층 Presentation Layer**

    - 정보를 통신에 알맞은 형태로 만듦, 하위 계층에서 온 데이터를 사용자가 이해할 수 있는 형태로 만듦. 부호교환, 암복호화
    - 프로토콜
        - JPEG : 이미지
        - MPEG : 멀티미디어
- **계층 7 - 응용 계층 Application Layer**
    - 프로토콜
        - HTTP : 하이퍼텍스트 교환
        - FTP : 서버- 클라이언트 사이의 파일을 전송
        - SMTP : 이메일 보냄
        - POP3 : 원격 서버로부터 이메일 가져옴
        - IMAP :원격 서버로부터 이메일 가져옴
        - Telnet: 인터넷이나 로컬에서 네트워크 연결에 사용
    - 장비 : L7스위치

IPv4

- 32 비트
- 8비트씩 4부분으로 나뉜 10진수
- 유니캐스트, 멀티캐스트, 브로드캐스트

IPv6

- 128 비트
- 16비트씩 8부분으로 나뉜 16진수
- 유니캐스트, 멀티캐스트, 애니캐스트

4 → 6 전환 방법

- 듀얼 스택
- 터널링
- 주소변환

개발환경 인프라 구성 방식

- 온프레미스(On-Pramise)방식 : 외부 인터넷망이 차단된 상태에서 인트라넷 망만을 활용하여 개발환경을 구축하는 방식
- 클라우드(Cloud)방식 : 아마존, 구글, 마이크로소프트 등 클라우드 공급 서비스를 하는 회사들의 서비스를 임대하여 개발환경을 구축하는 방식
- 하이브리드 방식 : 온프레미스 + 클라우드

서킷 스위칭(Circuit Swiching)

네트워크 리소스를 특정 사용 층이 독점하도록 하는 통신 방식

애드 혹 네트워크(Ad-hoc Network)

노드들에 의해 자율적으로 구성되는 기반 구조가 없는 네트워크

패킷 스위칭(Packet Switching)

작은 블록의 패킷으로 데이터를 전송하며, 데이터를 전송하는 동안만 네트워크 자원을 사용하도록 하는 통신 방식

- X.25 : 통신을 원하는 두 단말장치가 패킷 교환망을 통해 패킷을 원활히 전달하기 위한 통신 프로토콜
- 프레임 릴레이 : ISDN을 사용하기 위한 프로토콜, ITU-T에 의해 표준으로 작성됨
- ATM(Asynchronous Transfer Mode) : 비동기 전송모드, 광대역 전송에 쓰이는 스위칭 기법

12. 소프트웨어 패키징

제품 소프트웨어 패키징

개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 포장하는 과정

릴리즈 노트

최종 사용자인 고객에게 개발과정에서 정리된 제품의 릴리즈 정보를 제공하는 문서

- 작성항목
  - 헤더 : 문서 이름, 제품 이름, 버전 번호, 릴리즈 날짜, 참고 날짜 등의 정보
  - 개요 : 제품 및 변경에 대한 간략한 전반적 개요
  - 목적 : 목적에 대한 개요, 버그 수정 및 새로운 기능 기술
  - 이슈 요약 : 버그의 간단한 설명 또는 릴리즈 추가 항목 요약
  - 재현 항목 : 버그 발견에 따른 재현 단계 기술
  - 수정, 개선 내용
  - 사용자 영향도 : 버전 변경에 따른 최종 사용자 기준의 기능 및 응용 프로그램상의 영향도 기술
  - 소프트웨어 지원 영향도
  - 노트 : 소프트웨어 및 하드웨어 설치 항목, 제품, 문서를 포함한 업그레이드 항목 메모
  - 면책 조항 : 회사 및 표준 제품과 관련된 메시지, 프리웨어 및 불법 복제 방지, 중복 등 참조에 대한 고지 사항
  - 연락 정보

디지털 저작권 관리(DRM; Digital Right Management) 구성요소

- 콘텐츠 제공자(Contents Provider) : 콘텐츠를 제공하는 저작권자
- 콘텐츠 분배자(Contents Distributor) : 쇼핑몰 등으로써 암호화된 콘텐츠 제공

- 패키저(Packager) : 콘텐츠를 메타데이터와 함께 배포 가능한 단위로 묶는 기능
- 보안 컨테이너(Security Container) : 원본을 안전하게 유통하기 위한 전자적 보안 장치
- DRM 컨트롤러(DRM Controller) : 배포된 컨텐츠의 이용 권한을 통제
- 클리어링 하우스(Clearing House) : 소비자와 유통업자 사이에 발생하는 거래에 대해 디지털 저작권 라이선싱을 중개하고 라이선스 발급을 수행하는 장소

패키징 도구 구성

- 암호화(Encryption)
  - 공개키 기반구조(PKI) : 전자 서명 인증서를 발급받아 비밀통신 가능하게
  - 전자서명 : 서명자가 해당 전자문서에 서명했다는 사실을 나타내기 위해 논리적으로 결합된 전자적 형태의 정보
- 키 관리(Key Management)
  - 콘텐츠를 암호화한 키에 대한 저장 및 배포 기술
- 식별 기술(Identification)
  - DOI : 디지털 저작물의 저작권 보호를 위해 특정 번호 부여하는 바코드 시스템
  - URI : 인터넷에 있는 자원을 식별할 수 있도록 나타내는 주소
- 저작권 표현(Right Expression)
  - XrML : 디지털 콘텐츠 / 웹 서비스 권리 조건을 표현한 XML기반 마크업 언어
  - MPEG-21 : 멀티미디어 표준 규격
- 암호화 파일 생성(Packager)
  - 콘텐츠의 암호화를 통해 콘텐츠를 보호하는 기술
- 정책관리(Policy Management)
  - XML : 마크업 언어를 만드는데 사용하는 다목적 언어
  - CMS : 콘텐츠 작성, 수집, 관리, 배급부터 활용, 폐기까지 전 공급과정 관리
- 크랙방지(Tamper Resistance)
  - 코드 난독화 : 역공학 공격을 막기 위해 프로그램 소스코드 바꾸는 기술

    - Secure DB : 커널 암호화 방식, 데이터베이스 보안 강화 기술
- 인증(Authentication)
    - SSO : 한 번의 시스템 인증을 통하여 여러 정보시스템에 재인증 절차 없이 접근 가능한 통합 로그인 기술

제품 소프트웨어 사용자 메뉴얼

사용자가 소프트웨어 사용에 필요한 내용, 제반 절차, 환경등의 내용을 포함하는 문서

백업유형

- 전체백업(Full Backup): 데이터 전체에 대한 백업
- 차등백업(Differential Backup): 마지막 전체 백업 이후 변경된 모든 데이터를 백업
- 증분백업(Incremental Backup): 정해진 시간을 기준으로 그 이후에 변경된 파일만을 백업