**1장 요구사항 확인**

**소프트웨어 생명주기(SDLC)**

시스템의 전 공정을 체계화한 절차

**SDLC 모델 종류**

폭포수 모델 : 선형 순차적 모형 ( 고전적 생명주기 모형)

프로토타이핑 모델 : 프로토타입을 구현해, 고객의 피드백을 반영하며 만듬

나선형 모델 : 위험을 최소화하기 위해 점진적으로 개발

반복적 모델 : 구축 대상을 나누어 병렬적으로 개발 후 통합하거나 반복적으로 개발

**소프트웨어 개발방법론**

구조적 방법론 : 기능에 따라 나누어 개발하여 통합(하향식 방법론)

나씨-슈나이더만 차트를 사용

정보공학 방법론 : 정보시스템 개발에 필요한 관리 절차와 작업 기반을 체계화

객체지향 방법론 : 복잡한 현실 세계를 사람이 이해하는 방식으로 시스템에 적용

컴포넌트 기반 방법론 : 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(피존의용단)

12가지 기본 원리

-짝프로그래밍(Pair Programming) : 개발자 둘이서 짝 코딩

-공동 코드 소유(Collective Ownership) : 시스템 코드는 누구든지 언제라도 수정가능

-지속적인 통합(Continuous Integration) : 매일 여러번씩 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 체계화 시스템 서술서를 통해 고객과 개발자간의 의사소통을 원활하게 함

간단한 디자인(Simple Design) : 요구사항에 적합한 가장 단순한 시스템 설계

테스트 기반 개발(Test Driven Develop)

리팩토링(Refactoring) : 프로그램의 기능은 바꾸지 않고 중복제거, 단순화등을 위해 시스템을 재구성하는 것

40시간 작업 : 40시간 이상 일X

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, 지식창출, 늦은 확정, 빠른인도, 사람존중, 최적화

**비용산정 모형**

하향식

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

상향식

LoC(Lind of Code) : 원시코드 라인수의 낙관치, 중간치, 비관치를 측정해 예측치를 구해 비용을 산정하는 방식

Man Month : 한 사람이 1개월 동안 할 수 있는 일의양을 기준으로 프로젝트 비용 산정하는 방식

COCOMO : 보헴이 제안, 프로그램 규모에 따른 비용 산정

조직형(Organic Mode) : 5만라인 이하

반 분리형(Semi-Detached Mode) : 30만 라인 이하

임베디드형(Embedded Mode) : 30만 라인 이상

푸트남 : 개발 주기의 단계별로 요구할 인력의 분포를 가정하는 방식, 생명주기 예측 모형, Rayleigh-Nordan 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 수렴기법, 비관치, 중간치, 낙관치의 3점 추정방식 이용

주 공정 연쇄법(CCPM) : 자원제약사항을 고려해 일정 작성

**현행시스템 파악**

구성/기능/인터페이스 파악 -> 아키텍처 및 소프트웨어 구성 파악 -> 하드웨어 및 네트워크 구성 파악

**소프트웨어 아키텍처**

여러가지 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 접근방법 (배구프논유)

-유스케이스 뷰 : 유스케이스, 아키텍처 도출, 다른 뷰 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 마주보는 두 개의 계층 사이에서만 상호작용

클라이언트-서버패턴 : 하나의 서버와 다수의 클라이언트

파이프-필터 패턴 : 데이터 스트림을 생성하고 처리하는 시스템에서 사용

브로커 패턴 : 분리된 컴포넌트들로 이루어진 분산 시스템에서 사용되고, 원격 서비스 실행을 통해 상호작용이 가능

MVC패턴 : 모델, 뷰, 컨트롤러

-모델 : 핵심기능, 데이터보관

-뷰 : 사용자에게 정보 표시

-컨트롤러 : 사용자로부터 요청 입력받아 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 품질요소

**디자인 패턴★**

SW설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계방법을 정리한 패턴

**생성 (bprofas)**

-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 어디서든 참조할 수 있음

**구조 (abcdffp)**

-adapter : 기존에 생성된 클래스를 재사용할 수 있도록 중간에서 맞춰주는 역할

-bridge : 기능 계층과 구현 계층을 연결, 구현부에서 추상계층분리

-composite : 객체들의 관계를 트리 구조로 구성

-decorator : 기존에 구현되어 있는 클래스에 필요한 기능 추가함

-facade : 복잡한 시스템에 대해 단순한 인터페이스 제공, 시스템 구조에 대한 파악을 쉽게함

-flyweight : 메모리 절약, 클래스의 경량화가 목적

-proxy : 실체 객체에 대한 대리 객체

**행위**

-Mediator : 중간에 통제, 통제자

-Interpreter : 언어의 다양한 해석, 구문에 해석을 맡는 클래스 각각 작성

-Iterator : 컬렉션 구현 방법 노출시키지 않으면서도 그 집합체 안에 들어있는 모든 항목에 접근할 방법을 제공

-Template Method : 상위 클래스 - 추상적, 하위클래스 - 구체적

-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 있을때, 이를 동적으로 연결되어 있는 경우에 따라 다르게 처리될 수 있도록 연결한 디자인

**운영체제**

컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스 담당

-윈도우즈 : 중/소규모 서버, 일반PC

-유닉스 : 대용량 처리, 엔터프라이즈 급 서버

-리눅스 : 중/대규모 서버 대상 , 높은 보안성, 비용가장적음

-안드로이드 : 리눅스 위에서 구동, 자바와 코틀린으로 작성

-IOS : 높은 보안성 , 고성능

**운영체제 현행 시스템 분석 고려사항(구신성기주)**

신뢰도, 성능, 기술지원, 주변기기, 구축비용

**미들웨어**

응용 프로그램과 환경간에 원만한 통신이 이루어질 수 있도록 제어해주는 SW

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

**요구공학** **(출석명확)**

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

**요구사항**

기능적 요구사항 : 시스템이 제공하는 기능, 서비스에 대한 요구사항(UI)

비기능적 요구사항 : 시스템이 수행하는 기능 이외의 사항(백엔드)

**요구사항 개발 단계**

도출

-인터뷰 : 이해 관계자와 직접 대화

-설문조사 : 설문지, 여론조사

-브레인스토밍 : 말을 꺼내기 쉬운 분위기로 만들어 비판없이 수용할 수 있도록 하는 회의

-델파이 기법 : 전문가의 경험적 지식을 통한 문제해결방법

-롤 플레잉 : 여러 사람이 각자가 맡은 역을 연기

워크숍 : 단기간에 다양하고 전문적인 정보를 획득하고 공유

분석

-청취기술

-인터뷰와 질문기술

명세

-비정형 명세기법

-자연어 기반

-사용자와 개발자 이해 용이

-명확성 및 검증문제

-정형 명세기법

-수학적인 원리와 표기법, Z-스키마,Petri Nets

-표현간결, 명확성 및 검증 용이

-기법 이해 어려움

확인 및 검증

정형 기술 검토

-동료 검토 : 2~3명이 진행, 작성자가 명세서 설명하고 이해관계자들이 설명들으면서 결함발견

-워크 스루 : 검토자료를 회의전에 배포해서 사전검토한 후 짧은 시간동안 회의 진행

-인스펙션 : 저작자 외의 다른 전문가 또는 팀이 검사하여 오류를 찾아내는 공식적 검토 방법

**2장 화면설계**

**UI 유형**

CLI(Command Line Interface) : 명령어를 텍스트로

GUI(Grapci User Interface) : 마우스, 전자펜

NUI(Natural User Interface) : 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) : 정확,완벽하게 사용자의 목표가 달성 될 수 있도록 제작

학습성(Learnability) : 초보와 숙련자 모두 쉽게 배우고 사용가능하게 제작

유연성(Flexibility) : 사용자의 요구사항을 최대한 수용하고 실수를 방지할 수 있도록 제작

**UI 품질 요구사항 (신기이유사효)**

기능성(Functionality) : 실제 사용시 정확하지 않은 결과가 발생할 확률과 시스템 동작 관찰 - 적절성, 정밀성, 상호운용성, 보안성, 호환성

신뢰성(Realiability) : 일정한 시간, 작동되는 시간동안 의도하는 기능을 수행함을 보증 - 성숙성, 고장허용성, 회복성

사용성(Usablity) : 어떠한 행위를 정확하고 쉽게 인지할 수 있어야 함

- 이해성, 학습성, 운용성

효율성(Efficiency) : 할당된 시간에 한정된 자원으로 얼마나 빨리 처리 할 수 있는가 - 시간효율성, 자원효율성

이식성(Portability) : 다른 운영체제에서도 얼마나 쉽게 적용이 가능한가

-적용성, 설치성, 대체성

**UI개발을 위한 주요 기법**

3C 분석 : 고객(Customer), 자사(Company), 경쟁사(Competitor) 비교 분석

SWOT 분석 : 기업 내부 환경과 외부환경을 분석해 Strength, Weakness, Opportunity, Threat요인을 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 특정문제나 과제에 대한 새로운 지식, 기술, 아이디어들을 서로 교환하고 검토하는 세미나

**UI 화면 설계**

스토리보드 : 정책, 프로세스, 와이어 프레임, 기능정의, 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 할때 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

**3장 데이터 입출력**

**[1] 논리데이터**

**데이터 모델**

현실세계의 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 (튜플?)

- 프로젝트(π) : R에서 주어진 속성들의 값으로만 구성됨

- 조인(⋈) : 공통 속성을 이용

- 디비전(÷) : 릴레이션 S의 모든 튜플과 관련있는 R의 튜플 반환

**관계 해석**

튜플 관계해석과 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**

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

(도부이결다조 = 두부이걸다줘)

1NF : 도메인이 원자값

2NF : 부분함수 종속 제거

3NF : 이행함수 종속 제거(A->B , B->C이면 A->C)

BCNF : 결정자 후보키가 아닌 함수 종속 제거

4NF : 다치(다중 값) 종속 제거

5NF : 조인 종속 제거

**이상현상(Anomaly)**

데이터의 중복성으로 인해 릴레이션을 조작할 때 발생하는 비합리적 현상

- 삽입 이상, 삭제이상, 갱신이상

**반정규화(De-Normalization)**

정규화 된 엔티티,속성, 관계에 대해 성능향상과 개발 운영의 단순화를 위해 중복,통합, 분리등을 수행하는 과정

**[2]물리데이터**

**물리 데이터 모델링**

논리 모델을 적용하고자 하는 기술에 맞도록 상세화해가는 과정

**참조무결성 제약조건**

참조하는 외래키의 값은 항상 참조되는 릴레이션에 기본키로 존재해야한다.

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

**인덱스**

전체 데이터 검색 없이 필요한 정보에 대해 신속한 조회가능

**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 저장될 데이터에 대한 명시적 제어가능

컴포지트(Composite) 파티셔닝 : 레인지, 해시, 리스트중에 2개이상의 결합

**[3] 데이터베이스**

**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

**데이터베이스 특성**

실시간 접근성, 계속적인 변화, 동시 공용, 내용 참조

**DBMS**

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

**DBMS 유형**

- 키-값 DBMS

- 컬럼기반 데이터 저장(Column Faimily Data Store)

- 문서 저장(Document Store)

- 그래프(Graph Store) : 시맨틱 웹과 온톨로지 분야

**빅데이터**

시스템, 서비스, 조직등에서 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

**NoSQL의 특성(BASE)**

Bascally 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(ClassFication) : 과거 데이터로부터 특성을 찾아내어 분류모형을 만들어 결과값 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

**4장 통합구현**

**연계 메커니즘**

응용 소프트웨어와 연계 대상 모듈간의 데이터 연계시 요구사항을 고려한 연계 방법과 주기를 설계하기 위한 메커니즘

**주요 연계기술**

직접 연계

-DB링크, DB연결, API, JDBC, 하이퍼링크

간접 연계

-연계 솔루션(EAI) : 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보전달, 연계, 통합을 가능하게 해주는 솔루션, 어댑터 이용

-Web Servic/ESB : WSDL과 SOAP프로토콜을 이용한 시스템 간 연계

-Socket : 소켓을 생성하여 포트를 할당하고, 클라이언트의 요청을 연결하여통신

**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 입출력도구

-브로커 : 데이터 포맷과 코드를 변환하는 솔루션

-메시지 큐 : 비동기 메시지를 사용하는 다른 응용 프로그램 사이에서 데이터를 송수신 하는 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 집중형 방식

-메시지버스 : 미들웨어를 두어 연계하는 통합 방식

-하이브리드 : 그룹내는 허브 앤 스포크, 그룹간에는 메시지 버스

**ESB(Enterprise Service Bus)**

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간에 하나의 시스템으로 관리 운영할 수 있도록 서비스 중심의 통합을 지향하는 아키텍처, 느슨한 결합방식

- 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,SMTP등을 사용하여 XML기반의 메시지를 네트워크 상태에서 교환하는 프로토콜

WSDL(Web Service Description Language) : 웹 서비스 명, 제공 위치, 메시지포맷, 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 포멧,

AJAX에서 많이 사용되고 XML을 대체하는 주요 데이터 포멧

**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, XMLHttpRequest객체를 이용해 전체 페이지를 새로 로드하지 않고 필요한 부분만 로드한다.

**REST(Representational State Transfer)**

웹과 같은 분산 하이퍼미디어 환경에서 자원의 존재/상태 정보를 표준화된 HTTP 메서드로 주고받는 웹 아키텍쳐(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)**

API방식 : 암호모듈(API)를 적용하는 애플리케이션 수정방식

Plug-In 방식 : 암-복호화 모듈이 DB서버에 설치된 방식

TDE방식 : DBMS커널이 자체저으로 암-복호화 기능 수행

Hybrid방식 : API + Plug-In

인터페이스 구현 검증 도구

xUnit : 자바, C++, .Net등 다양한 언어를 지원하는 단위테스트 프레임 워크

STAF : 서비스 호출, 컴포넌트 재사용등 다양한 환경 지원하는 테스트 프레임 워크

FitNesse : 웹 기반 테스트 케이스 설계/실행/결과 확인등을 지원

NTAF : FitNess + STAF를 통합한 NHN(Naver)의 테스트 자동화 프레임워크

Selenium : 다양한 브라우저 지원 및 개발언어를 지원, 테스트 스크립트 언어 학습할 필요없음, 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 감시기능 제공

제니퍼(Jennifer) : 애플리케이션 개발부터 테스트, 오픈, 운영, 안정화까지 전 생애주기 단계동안 성능을 모니터링하고 분석해주는 APM 소프트웨어

**[8] 서버프로그램 구현**

**개발 도구(빌구형테)**

빌드 도구

구현 도구

테스트 도구

형상 관리 도구

**서버 하드웨어 개발 환경**

웹 서버 : 정적콘텐츠(CSS, Javascript, Image)처리 /Apache 웹서버

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 컴퓨터간의 연결을 쉽고 안전하게 할수있게 함

DBMS : 데이터베이스 관리

**형상관리**

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

**절차 (통식감기) //약술형**

- 형상식별 : 형상 관리 대상 정의 및 식별

- 형상통제 : 형상 항목 버전 관리를 위해 변경 여부와 변경활동통제

- 형상 감사 : 소프트웨어 베이스라인의 무결성 평가, 베이스라인 변경 시 요구사항과 일치하는지 검토

- 형상 기록 : 형상 및 변경 관리에 대한 각종 수행결과 기록

(베이스 라인 : 개발과정의 각 단계별 산출물에 대한 변화를 통제하는 시점의 기준)

**소프트웨어 형상 관리 도구**

공유 폴더 방식 : 매일 개발이 완료된 파일은 약속된 위치의 공유폴더에 복사

- RCS : 소스파일의 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

분산저장조 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

**응집도(Cohesion) (기순통절시논우)**

모듈 내부 구성요소간 연관정도 (낮음[나쁜품질] -> 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 모여있는 경우

순차적(Sequential) : 한 활동으로부터 나온 출력값을 다른 활동이 사용하는 경우

기능적(Functional) : 모듈 내부 모든 기능이 단일한 목적을 위해 수행되는 경우

**결합도(Coupling) (내공외제스자)**

모듈 내부가 아닌 외부 모듈과의 연관도 또는 모듈 간의 상호 의존성

(높음[나쁜품질] -> 낮음[좋은 품질])

내용(Content) : 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우

공통(Common) : 파라미터가 아닌 모듈 밖에 선언되어 있는 전역변수를 참고하고 갱신하는 식으로 상호 작용하는 경우

외부(External) : 두 개의 모듈이 외부에서 도입된 인터페이스를 공유할경우

제어(Control) : 단순처리할 대상인 값만 전달되는게 아니라 어떻게 처리를 해야한다는 제어요소가 전달되는 경우

스탬프(Stamp) : 인터페이스로 배열, 객체, 구조등이 전달되는 경우

자료(Data) : 파라미터를 통해섬나 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, 팬아웃은 낮게

**배치 프로그램**

일련의 작업들을 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법

- 이벤트 배치 : 사전에 정의해 둔 조건 충족시

- 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