\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1주차 소프트웨어 생명(수명) 주기

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-소프트웨어 공학(SE)

1. 소프트웨어 위기를 극복하기 위한 품질과 생산성을 향상시킬 목적

2. IEEE 소프트웨어 공학 표준 용어사전, Fairly, Boehm

- IEEE 소프트웨어 공학 표준 용어사전

체계적 접근 방안

- Fairly

지정된 비용, 기간 내 생산, 유지보수 기술

= Boehm

문서작성과정

-소프트웨어 생명 주기

1. 요구사항 분석,설계,개발,관리하는 과정,

2. 소프트웨어 개발방법론의 바탕이 되는 것

3. 폭포수, 프로토타입, 나선형, 애자일

- 폭포수(waterfall)

1. 타당성검토 -> 계획 -> 요구분석 -> 설계 -> 구현(코딩) -> 시험(검사) -> 유지보수

2. 이전단계로 못돌아감, 각 단계를 확실히 매듭지음

3. 고전적 생명 주기 모형, 오래되고 폭넓음

4. 선형 순차적 모형

5. 계획, 문서(매뉴얼) 중심으로 만듦

- 프로토타입(원형 모형)

1. 요구수집 -> 빠른설계 -> 프로토타입구축 -> 고객평가 -> 프로토타입조정 -> 구현 -> 요구 수집 -> ...

2. 견본을 만들어 최종 결과물 예측

3. 사용자와 시스템 사이의 인터페이스에 중점

- 나선형(Spiral Model)

1. 계획수립 -> 위험분석 -> 개발및검증 -> 고객평가

2. 폭포수,프로토타입 장점에 위험 분석 기능 추가

3. 점진적 모형, 완벽한 최종 소프트웨어를 개발, 위험 최소화

4. 유지보수 필요없음

- 애자일

1. 개발 -> 설계 -> 테스트

2. 고객 요구사항 변화에 유연하게 대응할 수 있도록 일정주기 반복

3. 주기 = 스프린트 = 이터레이션

4. 고객 평가와 요구 적극 수용

5. 몇주단위, 방대한 문서보다는 실행되는 SW중심

6. 개발진척도 확인 1차 기준은 소프트웨어, 단순화 추구

7. 스크럼, XP, 칸반, Lean, 크리스탈, ASD, 기능중심개발(FDD), DSDM, DAD

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2주차 애자일 -> 스크럼

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-스크럼

1. 팀이 중심이 되어 개발하는 것

2. 스스로

3. 제품 책임자, 스크럼 마스터, 개발팀

- 제품 책임자(PO : Product Owner)

1) 이해관계자들 중 요구사항 책임지고 의사 결정할 사람

2) 요구 사항이 담긴 백로그 작성, 우선순위 지정

3) 팀원이 백로그에 스토리 추가만 가능

- 스크럼 마스터(SM : Scrum Master)

1) 팀에게 객관적인 시각에서 조언해주는 가이드 역할

2) 팀을 통제하는 것이 아님

3) 일일 스크럼 회의 주관 -> 과정 검토, 장애 요소 공론화하여 처리

- 개발팀(DT : Development Team)

1) 위 두개를 제외한 모든 팀원, 7~8명

4. 제품 백로그 -> 스프린트 계획 회의 -> 스프린트 -> 일일 스크럼 회의

-> 스프린트 검토 회의 -> 스프린트 회고

- 제품 백로그

1) 제품 책임자(PO)가 우선순위에 따라 모든 요구사항 나열한 목록

2) 지속적으로 업데이트 됨

3) 백로그에 작성된 사용자 스토리 기반으로 전체 일정 계획인 릴리즈 계획 수립

- 스프린트 계획 회의

1) 스크럼마스터(SM) 주관으로 단기 일정 수립

2) 태스크 단위로 분할 후 개발자별로 수행할 작업 목록인 스프린트 백로그 작성

- 스프린트

1) 실제 개발 과정 2~4주

2) 속도(=한번의 스프린트에서 팀이 감당할 백로그의 양)를 추정하여 개발자에게 할당

3) 태스크는 3개로 나눠짐 - 할일(To Do), 진행중(In Progress), 완료(Done)

- 일일 스크럼 회의

1) 모든 팀원이 매일 15분 정도 진행 상황 점검

2) 소멸차트

3) 개발 과정 중 장애 요소 해결

4) 스크럼마스터(SM) 주관

- 스프린트 검토 회의

1) 사용자가 포함된 참석자 앞에서 테스팅

2) 제품 책임자(PO) 가 피드백 정리 후 제품 백로그 업데이트

3) 스프린트의 1주당 1시간 내 진행

- 스프린트 회고

1) 스프린트 주기를 되돌아보며 규칙 준수했는지 개선할 점 없는지 확인, 기록

2) 해당 스프린트가 끝난 시점에서 수행, 일정 주기로 수행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3주차 애자일 -> XP

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-XP(eXtreme Programming)

1. 반복을 극대화하여 개발 생산성 향상

2. 짧고 반복적인 개발 주기, 단순한 설계, 고객 적극 참여, 소규모

3. 빠르게 개발하는 것이 목표

4. 릴리즈(=부분적 기능이 완료된 것) 를 짧게 반복해서 요구사항에 대한 가시성(요구사항이 잘 반영됐는지) 높임

5. 릴리즈 테스트 고객 적극 참여

6. 핵심 5가지 가치 : 피드백, 존중, 의사소통, 용기, 단순성 (= 피존의 용기는 단순하다)

사용자 스토리

1) 고객의 요구사항을 간단한 시나리오로 표현

2) 기능 단위로 구성, 간단한 테스트 사항 기재

릴리즈 계획 수립

1) 몇 개의 스토리가 적용되어 부분적 기능 완료된 제품

2) 부분 또는 전체 개발 완료 시점에 대한 일정을 수립

스파이크

1) 요구사항의 신뢰성 높이고 별도로 만드는 간단한 프로그램

2) 다른 조건 모두 무시하고 작성

이터레이션

1) 하나의 릴리즈를 더 세분화한 단위

2) 1~3주, 기간 중 새로운 스토리 작성 가능

승인 검사

1) 인수 테스트

2) 릴리즈 단위의 부분 완료 제품이 구현되면 수행

3) 완료되면 다음 이터레이션 진행

소규모 릴리즈

1) 고객의 반응을 기능별로 확인 가능해서 요구사항에 좀 더 유연하게 대응

-xp 주요 실천 방법

Pair Programming(짝 프로그래밍)

1) 다른 사람과 같이 프로그래밍 수행

Collective Ownership(공동 코드 소유)

1) 개발 코드에 대한 권한 책임을 공동으로 소유

Test-Driven Development(테스트 주도 개발)

1) 테스트 케이스 먼저 작성 후 그것에 맞춰 개발

2) 자동화된 테스팅 도구를 사용해 지속적으로 테스트 진행

Whole Team(전체 팀)

1) 모든 구성원들이 역할을 가지고 책임을 가짐

Continuous Intergration(계속적인 통합)

1) 모듈 단위로 나눠 코드 작업이 마무리 될때마다 지속적으로 통합

Design Improvement/Refactoring(디자인 개선/리팩토링)

1) 프로그램 기능 변화 없이 단순화 유연성 강화를 통해 시스템 재구성

Small Releases(소규모 릴리즈)

1) 릴리즈 기간을 짧게 반복해 요구 변화에 신속히 대응

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4주차 현행 시스템 파악

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-현행 시스템 파악

1. 개발 범위를 명확히 설정

2. 구기인 이라는 친구가 / 아라소 / 하네

1단계

- 시스템 구성 파악

- 시스템 기능 파악

- 시스템 인터페이스 파악

2단계

- 아키텍처 구성 파악

- 소프트웨어 구성 파악

3단계

- 하드웨어 구성 파악

- 네트워크 구성 파악

시스템 구성 파악

1) 주요 업무(기간 업무)와 지원 업무로 구분

2) 모든 정보 시스템의 현황을 파악할 수 있도록 업무 명칭, 주요 기능 명시

시스템 기능 파악(계층)

1) 주요 기능, 하부 기능, 세부 기능을 계층형으로 표시

시스템 인터페이스 파악

1) 단위 업무 시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 등을 명시

2) 데이터,통신규약, 등

아키텍처 구성 파악(계층)

1) 업무 수행에 어떠한 기술 요소들이 사용되는지 최상위 수준에서 계층별로 표현

2) 아키텍처 구성도로 작성

소프트웨어 구성 파악

1) 소프트웨어의 제품명, 용도, 라이선스 수 명시

2) 시스템 구축비용에서 적지않은 비용 차지하므로 정확한 파악이 중요

하드웨어 구성 파악

1) 서버의 주요 사양과 수량, 이중화(운용 서버의 장애 대비 예비 서버) 적용 여부 명시

2) 이중화로 인한 비용증가, 구축난이도 증가 고려

네트워크 구성 파악

1) 네트워크 구성 파악, 서버의 위치 네트워크 연결 방식

2) 보안 취약성 분석

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5주차 개발 기술 환경 파악 가성기주구(가용성,성능,기술지원,주변기기,구축비용)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-개발 기술 환경

1) 운영체제,데이터베이스관리 시스템, 미들웨어(운영체제와 응용프로그램 사이에서 별도 서비스 제공)

선정할 때 고려해야 할 사항 기술

2) 오픈 소스 사용 시 주의 사항 기술

운영체제(OS)

1) 사용자가 컴퓨터를 편하고 효율적으로 사용할 수 있도록 제공하는 소프트웨어

2) 시스템 소프트웨어의 일종으로 다른 응용 프로그램이 작용하도록 환경 제공

3) Windows ,UNIX, Linux, Mac OS, iOS, Android

4) 가용성, 성능, 기술 자원, 주변 기기, 구축 비용

-가용성

1. 장시간 운영으로 인한 운영체제 고유의 장애 발생 가능성

2. 메모리 누수로 인한 성능 저하 및 재가동

3. 패치 설치로 인한 재가동

-성능

1. 대규모 동시 사용자 요청에 대한 처리

2. 대규모 파일 작업에 대한 처리

3. 지원 가능한 메모리 크기

-기술 자원

1. 기술 지원 및 정보 공유

2. 오픈소스 여부

-주변 기기

1. 설치 가능한 하드웨어

2. 주변 기기 지원 여부

-구축 비용

1. 하드웨어 비용, 응용 프로그램 라이선스 정책 및 비용

2. 유지관리, 총 소유 비용(TCO)

데이터베이스 관리 시스템(DBMS)

1) 정보 생성 및 관리 시스템

2) 데이터 종속성, 중복성 해결, 모든 응용 프로그램이 공용사용

3) Oracle, IBM DB2, Microsoft SQL Server, MySQL, SQLite, MongoDB, Redis

4) 가용성, 성능, 기술 자원, 상호 호환성, 구축 비용

-가용성

1. 장시간 운영으로 인한 운영체제 고유의 장애 발생 가능성

2. 백업이나 복구의 편의성, DBMS 이중화 및 복제 지원

3. 패치 설치로 인한 재가동

-성능

1. 대규모 데이터 처리 성능

2. 대용량 트랜잭션 처리 성능

3. 튜닝 옵션 지원, 질의 최적화 지원

-기술 자원

1. 기술 지원 및 정보 공유

2. 오픈소스 여부

-상호 호환성

1. 설치 가능한 운영체제 종류

2. JDBC(자바에서 DB 접근), ODBC(응용 프로그램에서 DB 접근)와 호환 여부

-구축 비용

1. 라이선스 정책 및 비용

2. 유지관리, 총 소유 비용(TCO)

웹 애플리케이션 서버(WAS:Web Application Server)

1) 사용자의 요구에 따라 변하는 동적인 콘텐츠 처리하기 위해 사용되는 미들웨어

2) 데이터 접근, 세션 관리, 트랜잭션 관리를 위한 라이브러리 제공

3) 주로 데이터베이스 서버와 연동해서 사용

4) Tomcat, GlassFish, JBoss, Jetty, JEUS, Resin, WebLogic, WebSphere

5) 가용성, 성능, 기술 지원, 구축 비용

-가용성

1. 장시간 운영으로 인한 장애 발생 가능성

2. 안정적인 트랜잭션 처리, WAS 이중화 지원

3. 패치 설치로 인한 재가동

-성능

1. 다양한 설정 옵션 지원

2. 대용량 트랜잭션 처리 성능

3. 가비지 컬렉션(GC)의 다양한 옵션

4. 실제로 사용되지 않으면서 가용 공간 리스트에 반환되지 않는 메모리 공간을

강제로 해제하여 사용할 수 있도록 하는 메모리 관리 기법

-기술 자원

1. 기술 지원 및 정보 공유

2. 오픈소스 여부

-구축 비용

1. 라이선스 정책 및 비용

2. 유지관리, 총 소유 비용(TCO)

오픈 소스 사용

1) 누구나 사용할 수 있도록 소스 코드를 공개한 것

2) 사용시에는 라이선스의 종류, 사용자 수, 기술의 지속 가능성 고려해야 됨

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6주차 요구사항 정의

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-요구사항

1) 서비스에 대한 설명, 제약조건, 이해관계자들 간의 의사소통을 도움

2) 기능 요구사항, 비기능 요구사항, 사용자 요구사항, 시스템 요구사항

기능 요구사항

1. 시스템이 무엇을 하는지, 어떤 기능을 하는지에 대한 사항

2. 시스템의 입력, 출력, 반드시 수행해야 하는 기능

3. 사용자가 시스템을 통해 제공받기를 원하는 기능

비기능 요구사항

1. 하드웨어, 소프트웨어, 네트워크 등의 시스템 장비 구성에 대한 요구사항, 제약사항

-성능 요구사항

1) 처리 속도 및 시간, 가용성 등 성능

-인터페이스 요구사항

1) 사용자 인터페이스에 대한 요구사항 다른 시스템과의 정보 교환에 사용되는 프로토콜

-데이터 요구사항

1) 데이터를 구축하기 위해 필요한 요구사항

-테스트 요구사항

1) 성능 테스트(BMT) 등을 테스트

-보안 요구사항

1) 시스템에 접근 통제

-품질 요구사항

1) 품질을 가용성, 정합성, 상호 호환성, 대응성, 신뢰성, 사용성, 유지\*관리성, 이식성,

확장성,보안성 등으로 구분하여 기술

-제약사항

1) 기술, 표준 업무, 법\*제도 등의 제약조건

-프로젝트 관리 요구사항

-프로젝트 지원 요구사항

사용자 요구사항

1. 사용자 관점에서 시스템이 제공해야 할 요구사항

2. USER-FRIENDLY

시스템 요구사항(=소프트웨어 요구사항)

1. 개발자 관점에서 사용자와 다른 시스템에 제공해야 할 요구사항

2. 전문적 기술적 용어

요구사항 개발 프로세스

1. 도출(Elicitation) -> 분석(Analysis) -> 명세(Specification)-> 확인(Validation)

2. 진행되기 전에 타당성 조사가 선행되어야 한다.

3. 요구공학의 한 요소(품질을 개선하여 실패를 최소화)

-요구사항 도출(수집)

1) 관계가 만들어지고 이해관계자 식별됨

2) 소프트웨어 개발 생명 주기(SDLC) 동안 주기적으로 반복

3) 청취, 인터뷰, 설문, 브레인스토밍(3인이상 의견 교환), 워크샵,

프로토타이핑(견본품, 단순한 설명을 대충 보여줌)

유스케이스(사용자 요구사항을 기능 단위로 표현)

-요구사항 분석

1) 명확하지 않고 모호한 요구사항 걸러내는 과정

2) 요구사항의 타당성 조사 및 비용과 일정에 대한 제약 설정

3) 자료 흐름도(DFD), 자료 사전(DD) 사용

4) 개발의 실질적인 첫 단계

-요구사항 명세

1) 모델 작성 및 문서화

2) 기능 요구사항은 빠짐없이 기술, 비기능 요구사항은 필요한 것만 기술

3) 소단위 명세서 사용될 수 있음

4) 소프트웨어 요구사항 명세서(SRS) - 정형 명세 기법, 비정형 명세 기법

정형 명세 기법 - 수학적, 간결, 어려움, VDM, Z, Petri-net, CSP

비정형 명세 기법 - 객체 중심, 자연어, 쉬움, FSM, Decision Table, ER 모델링, STATE Chart(SADT)

-요구사항 확인(검증)

1) 요구사항 명세서가 정확하게 작성되었는지 검토

2) 이해관계자들이 검토, 모든 문제 확인 불가

3) 형상 관리(SCM, 프로그램, 프로그램을 설명하는 문서,데이터 등을 관리) 수행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7주차 요구사항 분석

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-요구상항 분석

1. 개발의 실질적인 첫 단계, 요구사항 이해 및 문서화(명세화)

2. 타당성 조사, 비용과 일정에 대한 제약 설정

3. 소프트웨어 분석가에 의해 수행

4. UML, 자료흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec), 개체 관계도(ERD)

상태 전이도(STD), 제어 명세서

구조적 분석 기법 (자료흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec), 개체 관계도(ERD)

상태 전이도(STD), 제어 명세서)

1. 도형 중심 분석, 하향식 방법

자료 흐름도(DFD)

1. 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술

2. 자료 흐름 그래프, 버블 차트

3. 프로세스, 자료 흐름, 자료 저장소, 단말

-프로세스(Process)

1) 처리, 기능, 변환, 버블

2) 둥글게 표현

-자료 흐름(Data Flow)

1) 흐름이나 연관관계

2) 화살표 위에 자료 이름 기입

-자료 저장소(Data Store)

1) 파일, 데이터베이스를 나타낸다.

2) 도형 안에 자료 저장소 이름 기입

-단말(Terminator)

1) 시스템과 교신하는 외부 개체, 입력 출력

2) 도형 안에 이름 기입

자료 사전(DD)

1. 자료 흐름도에 있는 자료를 더 자세히 정의한 것

2. 데이터의 데이터, 메타 데이터라고 함

= : ~로 구성되있다, 자료 정의

+ : 그리고

() : 생략 가능

[|] : 또는

{} : 반복, n번 이상 m번 이하

\*\* : 주석

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8주차 요구사항 분석 CASE와 HIPO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-요구사항 분석을 위한 CASE(자동화 도구)

1. 요구사항을 자동으로 분석 및 명세서 기술

2. 표준화를 통한 문서화 품질 개선

3. 데이터베이스가 모두에게 이용 가능

4. SADT, SREM, PSL/PSA, TAGs, EPOS

- SADT(Structured Analysis and Design Technique)

1) SoftTech 사에서 개발

2) 구조적 요구 분석을 위한 블록 다이어그램

- SREM(Software Requirements Engineering Methodology) = RSL/REVS

1) TRW 사가 우주국방 시스템에서 실시간 처리 소프트웨어

2) RSL(요구사항 기술 언어) - 요소, 속성, 관계, 구조

요소 - 개체와 개념

속성 - 요소를 수정하거나 수식

관계 - 개체들 간의 관계

구조 - 정보 흐름을 묘사

3) REVS - RSL로 기술된 요구사항 자동 분석 및 명세서 출력

- PSL/PSA

1) 미시간 대학에서 개발

2) PSL - 요구사항 기술 언어

3) PSA - 요구사항 분석기

- TAGs

1) 시스템 공학 방법 응용에 대한 자동 접근 방법 개발 주기 전과정에 이용

2) 통합 자동화 도구

3) 구성 - IORL(언어), 분석기, 기초 TAGs 방법론

-HIPO(Hierarchy Input Process Output)

1. 시스템 분석 및 설계나 문서화할 떄 사용되는 기법

2. 입력 처리 출력 기능 나타냄

3. 하향식 소프트웨어 개발을 위한 문서화 도구

4. HIPO Chart - 기능을 여러개의 고유 모듈들로 분할하여 인터페이스 계층 구조로 표현

Visual Table of Contents, Overview Diagram, Detail Diagram

가시적 도표(도식목차) : 전체적인 기능과 흐름 보여주는 계층(Tree) 구조도

총체적 도표(총괄도표, 개요 도표) : 기능 보여주는 입력,처리,출력에 대한 정보 제공

세부적 도표(상세 도표) : 총체적 도표에 표시된 기능을 구성하는 기본요소들을 자세히 기술

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9주차 UML(Unified Modeling Language)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-UML

1. 대표적 객체 지향 모델링 언어

2. 고객과 개발자 상호간의 의사소통 원활

3. 국제 표준화기구인 OMG(Object Management Group) 에서 표준으로 지정

4. 6개 구조 다이어그램, 7개 행위 다이어그램

5. 사물(Things), 관계(Relationships), 다이어그램(Diagram)

사물(Things)

1) 가장 중요한 기본 요소

2) 구조사물, 행동사물, 그룹사물, 주해사물

구조사물 - 개념적,물리적 요소, (클래스, 유스케이스, 컴포턴트, 노드)

행동사물 - 시간,공간에 따른 요소들의 행위, (상호작용,상태 머신)

그룹사물 - 요소들을 그룹으로 묶음, (패키지)

주해사물 - 부가적인 설명 제약조건 표현, (노트)

관계(Relationships)

1) 사물과 사물 사이 연관성 표현

2) 연관관계, 집합관계, 포함관계, 일반화관계, 표현관계, 일반화관계, 의존관계, 실체화관계

연관관계

1. 두개 이상의 사물이 서로 관련

2. 실선과 화살표

3. 다중도 표기

사람 -> 집 : 사람은 집을 소유한다.

선생님 - 학생 : 선생님은 학생을 가르친다, 학생은 선생님에게 가르침 받는다.

집합관계

1. 포함하는 쪽과 포함되는 쪽은 서로 독립적이다.

2. 포함되는 쪽에서 포함하는 쪽으로 속이 빈 마름모를 연결한다.

포함관계

1. 집합관계의 특수한 형태, 포함하는 사물의 변화가 사물에게 영향

2. 서로 생명주기를 함께함

3. 포함되는 쪽에서 포함하는 쪽으로 속이 채워진 마름모 연결

일반화관계

1. 부모와 자식으로 표현

2. 자식에서 부모에게 속이 빈 화살표 연결

에스프레소 -> 커피 <- 아메리카노

의존관계

1. 필요에 의해 서로 영향을 주는 짧은 시간동안 관계

2. 영향을 주는 사물에서 영향을 받는 사물쪽으로 점선 화살표 연결

등급 ----> 할인율 : 등급이 올라가면 할인율이 올라감

실체화관계

1. 사물이 할 수 있거나 해야 하는 기능, 그룹화

2. 사물에서 기능 쪽으로 빈 점선 화살표

비행기 ---> 날 수 있는 <---- 새

다이어그램(Diagram)

1) 사물과 관계를 도형으로 표현한 것

2) 구조적 다이어그램, 행위 다이어그램

-구조적 다이어그램 종류 (정적)

클래스 다이어그램

1. 클래스로 표현

2. 시스템 구조 파악, 구조상의 문제점 도출

객체 다이어그램

1. 클래스에 속한 객체, 객체 사이의 관계

2. 럼바우 객체지향 분석 기법에서 객체 모델링에 활용

컴포넌트 다이어그램

1. 실제 구현 모듈인 컴포넌트 간의 관계나 인터페이스 표현

2. 구현 단계에서 사용

배치 다이어그램

1. 물리적 요소들의 위치 표현

2. 노드와 의사소통 경로로 표현

3. 구현 단계에서 사용

복합체 구조 다이어그램

1. 클래스나 컴포넌트가 복합 구조를 갖는 경우 내부 구조 표현

패키지 다이어그램

1. 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계 표현

-행위 다이어그램 종류 (동적)

유스케이스 다이어그램

1. 사용자의 요구 분석, 기능 모델링 작업에 수행

2. 사용자와 사용사례로 구성

순차 다이어그램(Sequence Diagram)

1. 상호작용하는 시스템이나 객체들이 주고받는 메세지 표현

커뮤니케이션 다이어그램

1. 메세지뿐만 아니라 객체들 간의 연관까지 표현

상태 다이어그램

1. 하나의 객체가 자신이 속한 클래스의 상태 변화를 표현

2. 럼바우 객체지향 분석 기법에서 동적 모델링에 활용

활동 다이어그램

1. 시스템이 어떤 기능을 수행하는지 객체 처리 로직, 처리 흐름을 순서에 따라 표현

상호작용 개요 다이어그램

1. 상호작용 다이어그램 간의 제어 흐름을 표현

타이밍 다이어그램

1. 객체 상태 변화와 시간 제약을 표현

-스테레오 타입

1) 길러멧<<>> 사용

2) 기본 기능 외의 추가적 기능 표현

<<include>> : 연결된 다른 UML요소에 대해 포함 관계에 있는 경우

<<extend>> : 연결된 다른 UML요소에 대해 확장 관계에 있는 경우

<<interface>> : 인테페이스 정의

<<exception>> : 예외 정의

<<constructor>> : 생성자 역할 수행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10주차 주요 UML다이어그램

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-유스케이스 다이어그램 (연포확일)

1) 기능을 사용자 관점에서 표현

2) 외부요소와 시스템 간의 상호 작용을 확인 가능

3) 요구사항 분석 도구, 시스템 범위 파악

4) 시스템 범위, 액터, 유스케이스, 관계

시스템 범위

1. 시스템 내부 기능과 외부 시스템을 구분하기 위해 내부 유스케이스들을 사각형으로 묶음

액터

1. 시스템과 상호작용하는 모든 외부 요소, 사람이나 외부시스템을 의미

2. 주액터, 부액터

주액터 : 시스템을 사용함으로써 이득을 얻는 대상, 주로 사람

부액터 : 주액터의 목적 달성을 위해 시스템에 서비스를 제공하는 외부 시스템, 조직 기관

유스케이스

1. 사용자 관점에서 시스템이 액터에게 제공하는 서비스 또는 기능

관계

1. 액터와 유스케이스, 유스케이스끼리의 관계

2. 연관 관계, 포함 관계, 확장 관계, 일반화 관계

-클래스 다이어그램(연집포일의)

1) 클래스, 속성, 오퍼레이션, 제약조건, 관계 등이 있음

2) 구조적 다이어그램

3) 시스템 구성 요소를 문서화하는 데 사용

4) 시스템을 모델링하는 데 사용

5) 클래스, 제약조건, 관계

클래스

1. 각각의 객체들이 갖는 속성과 오퍼레이션(동작) 표현

2. 3개의 구획으로 나눠 클래스의 이름, 속성, 오퍼레이션 표기

속성 : 클래스의 상태나 정보

오퍼레이션 : 클래스가 수행할 수 있는 동작, 함수(메소드)

제약조건

1. 속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건

관계

1. 클래스와 클래스 사이의 연관성

2. 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계

-접근 제어자

public : + 모두 접근 가능

private : - 클래스 내부에서만 접근 가능

protected : # 동일 패키지 내의 클래스 또는 해당 클래스를 상속 받은 외부 패키지의 클래스에서 접근 가능

package : ~ 동일 패키지 내부에 있는 클래스에서만 접근 가능

-순차(시퀀스) 다이어그램

1) 시스템이나 객체들이 메세지를 주고 받으며 시간의 흐름에 따라 상호작용 표현

2) 수행기간, 상호작용 확인 가능

3) 액터, 객체, 생명선, 실행 상자, 메시지

액터

1. 서비스 요청하는 외부 요소, 사람이나 외부 시스템

객체

1. 메시지를 주고받는 객체

생명선

1. 객체가 메모리에 존재하는 기간, 객체 아래에 점선을 그어 표현

실행 상자

1. 객체가 메시지를 주고받으며 구동되고 있음을 표현

메시지

1. 객체가 상호 작용을 위해 주고받는 메시지

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11주차 - 사용자 인터페이스

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 사용자 인터페이스(UI)

1) 개요

- 사용자와 시스템 간의 상호작용이 원활하게 이뤄지도록 도와주는 장치, 소프트웨어

- 세가지 분야

- 정보 제공, 전달을 위한 물리적 제어에 관한 분야

- 콘텐츠의 상세적인 표현과 전체적인 구성에 관한 분야

- 모든 사용자가 간편하게 사용하도록 하는 기능에 관한 분야

2) 특징

- 사용자 만족도에 가장 큰 영향, 변경이 가장 많이 발생

- 편리성, 가독성 높임

- 최소한의 노력으로 원하는 결과 얻음

- 사용자 중심

3) 구분

- CLI(Command Line Interface) : 명령과 출력이 텍스트 형태

- GUI(Graphical User Interface) : 아이콘 등 그래픽 환경

- NUI(Natural User Interface) : 사용자의 말이나 행동, (Tap, Double Tap, Drag, Pan, Press, Flick, Pinch)

- VUI(Voice User Interface) : 음성

- OUI(Organic User Interface) : 모든 사물과 사용자 간의 상호작용, ex)가상현실,증강현실

4) 기본 원칙(직유학유)

- 직관성 : 누구나 쉽게 이해, 사용

- 유효성 : 사용자 목적 정확히 이해, 달성

- 학습성 : 누구나 쉽게 배움

- 유연성 : 사용자 요구사항 최대한 수용, 실수 최소화

5) 설계 지침

- 사용자 중심 : 사용자가 쉽게 이해, 사용

- 사용성 : 사용자가 얼마나 빨리 쉽게 이해, 효율적, 가장 우선적 고려

- 일관성 : 일관성 있게 쉽게 기억하게

- 단순성 : 조작 방법 단순화 인지적 부담 최소화

- 결과 예측 가능 : 기능만 보고 결과 예측

- 가시성 : 메인 화면에 주요 기능 노출시켜 최대한 조작 쉽게

- 심미성 : 디자인적, 가독성 높임

- 표준화 : 한 번 익히면 쉽게 사용

- 접근성 : 다양한 계층이 사용할 수 있게

- 명확성 : 개념적으로 쉽게 인지

- 오류 발생 해결 : 오류가 나면 사용자가 쉽게 인지적

6) 개발 시스템 기능

- 사용자 입력 검증

- 에러 처리, 에러 메세지 표시

- 도움과 프롬프트 제공

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13주차 - UI 설계 도구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. UI 설계 도구

1) 개요

- 사용자 요구사항에 맞게 UI 화면 구조나 배치 등을 설계할 때 사용하는 도구

2) 도구

1) 와이어 프레임

- 기획 단계 초기에 제작

- UI 요소 등에 대한 뼈대 설계

- 와이어 프레임 툴 : 손그림, 파워포인트, 키노트, 스케치, 일러스트, 포토샵

2) 목업

- 디자인, 사용 방법 설명, 평가 등을 위해 와이어프레임보다 좀 더 실제 화면과 유사하게 만듦

- 정적인 형태의 모형

- 목업 툴 : 파워 목업, 발사믹 목업

3) 스토리보드

- 와이어프레임 콘텐츠에 대한 설명, 페이지 간 이동 흐름 추가

- 최종적으로 참고하는 작업 지침서, 서비스 구축을 위한 모든 정보가 들어 있음

- 상단이나 우측에 제목, 작성자, 좌측에는 UI화면, 우측에는 디스크립션 기입

- 디스크립션은 화면에 대한 설명, 전반적인 로직 처리 등을 명확하고 세부적으로 작성

- 스토리보드 툴 : 파워포인트, 키노트, 스케치, Axure

4) 프로토타입

- 와이어프레임이나 스토리보드에 인터랙션을 적용

- 실제 구현된 것처럼 테스트가 가능한 동적인 형태 모형

- 사용성 테스트나 서비스 이해를 위한 샘플

- 작성 방법에 따라 페이퍼 프로토타입, 디지털 프로토타입

- 프로토타입 툴 : HTML/CSS, Axure, Flinto, 네이버 프로토나우, 카카오 오븐

5) 유스케이스

- 사용자 측면의 요구사항을 달성하기 위해 수행할 내용 기술

- 요구사항 빠르게 파악함으로써 프로젝트 초기에 기능적 요구 결정 및 문서화

- 자연어로 작성된 요구사항을 구조적으로 표현한 것, 다이어그램 형식

- 각각의 유스케이스에 대해 유스케이스 명세서 작성

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

15주차 - 품질 요구사항

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 개요

1) 소프트웨어의 기능, 성능, 만족도 등 소프트웨어에 대한 요구사항이 얼마나 충족하는가

2) ISO/IEC 9126

- 평가를 위한 표준 지침

- 품질에 대한 요구사항을 기술하거나 개발중, 개발 완료된 소프트웨어의 품질 평가에 사용

- 2011년에 호환성과 보안성 강화하여 ISO/IEC 25010으로 개정

- ISO/IEC 12119 테스트 절차가 포함된 표준

2. 기능 (기신사효유이) 귀신사요유이

1) 기능성

- 요구사항을 정확하게 만족하는 기능을 제공하는지

- 적절성/적합성 : 적절한 기능 제공

- 정밀성/정확성 : 결과 정확하게 산출

- 상호 운용성 : 다른 시스템과 어울려 작업

- 보안성 : 접근을 권한에 따라 허용, 차단

- 준수성 : 기능과 관련된 표준, 규정을 준수

2) 신뢰성

- 요구사항을 정확하고 일관되게 오류 없이 수행한는지

- 성숙성 : 고장을 피해갈 수 있는 능력

- 고장 허용성 : 고장 시에도 규정된 성능 수준 유지

- 회복성 : 고장 시 다시 회복하고 복구

3) 사용성

- 사용자가 쉽게 배우고 사용하고 다시 사용하고 싶은지

- 이해성 : 사용방법 이해 능력

- 학습성 : 학습 능력

- 운용성 : 사용자가 운용 제어 능력

- 친밀성 : 다시 사용하고 싶어 하도록 하는 능력

4) 효율성

- 할당된 시간동안 한정된 자원으로 얼마나 빨리 처리하는지

- 시간 효율성 : 반응 시간, 처리 시간

- 자원 효율성 : 적절한 자원의 양, 종류 제공

5) 유지 보수성

- 환경의 변화나 새로운 요구사항이 발생했을 때 개선하거나 확장할 수 있는지

- 분석성 : 결함 원인 식별

- 변경성 : 결함 제거 수정 쉽게 구현

- 안정성 : 변경으로 인한 예상치 못한 결과 최소화

- 시험성 : 소프트웨어 변경 검증

6) 이식성

- 다른 환경에서도 얼마나 쉽게 적용할 수 있는지

- 적용성 : 원래의 목적 외에 다른 환경으로 변경될 수 있는 능력

- 설치성 : 임의의 환경에 설치될 수 있는 능력

- 대체성 : 동일 환경에서 동일 목적을 위해 다른 소프트웨어를 대신하여 사용 될 수 있는 능력

- 공존성 : 다른 소프트웨어와 공존할 수 있는 능력

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

19주차 - UI 상세 설계

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. UI 시나리오 문서 개요

1) UI 설계서를 바탕으로 실제 설계 및 구현을 위해 모든 화면에 대해 자세한 설계를 진행

2) 반드시 시나리오 작성해야 함

3) 최종 목표를 달성하기 위한 방법을 순차적으로 묘사

4) UI설계자 또는 인터랙션 디자이너가 UI시나리오 문서 작성 후 그래픽 디자이너가 디자인 하고 개발자가 UI 구현

2. UI 시나리오 문서 작성 원칙

1) 전체적인 기능과 작동 방식을 알 수 있게 구체적으로 작성

- 주로 트리(보통 계층)구조, 플로차트 표기법으로 작성

2) 모든 기능에 공통적으로 적용될 UI 요소와 인터랙션을 일반 규칙으로 정의

3) 대표 화면의 레이아웃과 속할 기능 정의

4) 예외 상황 대비 다양한 케이스 정의

5) 일반 규칙을 지키며 기능별 상세 기능 시나리오 정의

6) UI 시나리오 규칙 정의

3. UI 시나리오 문성 작성을 위한 일반 규칙

1) 주요 키의 위치와 기능

- 모든 화면에 공통적으로 배치되는 주요 키의 위치와 기능

- 여러 화면 간의 일관성 보장

2) 공통 UI 요소

- 체크 박스, 라디오 버튼 UI 요소를 언제 어떤 형태로 사용할 지 정의 후 조작했을 때 반응 흐름 설명

- 체크 박스 : 여러 선택 중 1개 이상

- 라디오 버튼 : 여러 선택 중 하나만

- 텍스트 박스 : 데이터 입력과 수정 가능

- 콤보 상자 : 지정된 목록 내용 표시 또는 새로 입력 가능

- 목록 상자 : 목록 표시하지만 내용 입력 불가능

3) 기본 스크린 레이아웃

- 모든 화면에 공통적으로 나타나는 Titles, Ok/Back, Soft Key, Option 등의 위치 속성 정의

4) 기본 인터랙션 규칙

- 터치 제스처 등에 공통적으로 사용되는 조작 방법 실행 등 화면 전환 효과 기술

5) 공통 단위 태스크 흐름

- 공통적으로 사용되는 삭제, 검색, 매너 모드 등에 대해 인터랙션 흐름 설명

6) 케이스 문서

- 다양한 상황에서 공통적으로 적용되는 시스템 동작 설명

4. UI 시나리오 문서의 요건

1) 완전성

- 누락되지 않도록 최대한 상세하게 기술

- 기능보다 사용자의 태스크에 초점

2) 일관성

- 서비스 목표, 요구사항 등이 모두 일관성 유지관리

3) 이해성

- 누구나 이해하기 쉽게 기술

- 불분명하거나 추상적 표현 피하기

4) 가독성

- 표준화된 템플릿 활용하여 문서를 쉽게 읽을 수 있게 함

- v1.0, v2.0 등 문서 인덱스에 대한 규칙 목차 제공

- 들여쓰기 단락 사용

- 여백 빈페이지 일관성 있게 적용

- 문서들이 서로 참조될 수 있도록 지정

5) 수정 용이성

- 시나리오의 수정이나 개선이 쉬워야 됨

6) 추적 용이성

- 변경 사항은 언제, 어떤 부분이, 왜 발생했는지 쉽게 추적 가능해야 됨

5. UI 시나리오 문서로 인한 기대효과

1) 요구사항 의사소통 오류 감소

2) 재작업 감소 혼선 최소화

3) 불필요한 기능 최소화

4) 개발 비용 절감

5) 개발 속도 증가

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

21주차 - 소프트웨어 아키텍처

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소프트웨어 아키텍처의 설계

1) 소프트웨어의 골격이 되는 기본 구조

2) 소프트웨어를 구성하는 요소들 사이 관계를 표현한 시스템 구조 또는 구조체

3) 소프트웨어 개발 시 적용되는 원칙과 지침, 의사소통 도구

4) 비기능적 요구사항 제약 반영, 기능적 요구사항 구현 방법 찾음

5) 애플리케이션 분할 방법과 분할된 모듈에 할당될 기능, 모듈 간 인터페이스 결정

6) 모듈화, 추상화, 단계적 분해, 정보은닉

2. 모듈화

1) 시스템 기능들을 모듈 단위로 나누는 것

2) 재사용성 향상

3) 너무 작게 나누면 통합 비용 증가, 너무 크게 나누면 모듈 하나의 개발 비용 증가

4) 인터페이스가 단순해지고 효율적인 관리와 오류 파급 효과 최소화

3. 추상화 (제과자)

1) 포괄적인 개념 설계 후 차례로 세분화시켜 구체화

2) 완전한 시스템 구축 전 유사한 모델 만들어 테스트

3) 최소 비용으로 실제 상황에 대처, 대략적으로 구조 및 구성 파악

- 과정 추상화 : 전반적 흐름만 파악

- 데이터 추상화 : 데이터 구조를 대표할 수 있는 표현으로 대체

- 제어 추상화 : 이벤트 발생 정확한 절차 방법 정의하지 않고 다른 표현으로 대체

4. 단계적 분해

1) Niklaus Wirth 하향식 설계 전략

2) 추상화의 반복으로 세분화

3) 기능부터 점차적으로 구체화 후 알고리즘 자료구조 등 상세 내역 뒤로 미룸

5. 정보 은닉

1) 정보를 감추어 다른 모듈이 접근하거나 변경 못하게 하는 것

2) 필요한 정보만 인터페이스를 통해 전달

3) 독립적임, 수정, 시험, 유지보수 쉬움

6. 아키텍처의 품질 속성

1) 시스템 측면 ('성능' 좋은 폰으로 '변경'해서 용량이 '사기가 확보'됐다.)

- 성능 : 이벤트 발생 시 적절하고 빠르게 처리

- 보안 : 허용되지 않는 접근 막고 허용된 접근에 서비스 제공

- 가용성 : 장애 없이 정상적으로 서비스 제공

- 기능성 : 요구 기능 만족스럽게 구현

- 사용성 : 헤매지 않고 명확하고 편리하게 구현

- 변경 용이성 : 처음 설계 목표와 다른 하드웨어 등에서 동작하는 것

- 확장성 : 용량, 처리능력 확장시켰을 때 효과적으로 활용하는 것

- 기타 속성 : 테스트 용이성, 배치성, 안정성 등

2) 비즈니스 측면

- 시장 적시성 : 정해진 시간에 맞춰 출시

- 비용과 혜택 : 비용 더 투자해 유연성 높은 아키텍처를 만들 것인지 결정, 유연성 떨어지면 유지보수 비용 많이 듦

- 예상 시스템 수명 : 시스템을 얼마나 오래 사용할 것인지 고려

- 기타 속성 : 목표 시장, 공개 일정, 기존 시스템과의 통합 등

3) 아키텍처 측면

- 개념적 무결성 : 전체 시스템과 시스템을 이루는 구성요소들 간의 일관성 유지

- 정확성, 완결성 : 요구사항과 제약사항들 모두 충족

- 구축 가능성 : 모듈 단위 시스템을 적절하게 분배하여 유연하게 일정 변경

- 기타 속성 : 변경성, 시험성, 적응성, 일치성, 대체성

7. 아키텍처의 설계 과정

1) 설계 목표 설정

- 요구사항 분석하여 설계 목표 설정

2) 시스템 타입 설정

- 시스템과 서브시스템 타입 결정, 아키텍처 패턴 선택

3) 아키텍처 패턴 적용

- 아키텍처 패턴 참조하여 시스템의 표준 아키텍처 설계

4) 서브시스템 구체화

- 서브시스템 기능 및 상호작용을 위한 동작과 인터페이스 정의

5) 검토

8. 시스템 타입

1) 대화형 시스템 : 요구가 발생하면 시스템이 처리, 반응하는 시스템 (온라인 쇼핑몰)

2) 이벤트 중심 시스템 : 외부 상태 변화에 따라 동작하는 시스템, (전화, 비상벨)

3) 변환형 시스템 : 데이터가 입력되면 작업수행하여 결과 출력하는 시스템 (컴파일러)

4) 객체 영속형 시스템 : 데이터베이스를 활용하여 파일을 효과적으로 저장검색갱신할 수 있는 시스템(서버 관리 소프트웨어)

9. 협약에 의한 설계

1) 선행 조건 : 오퍼레이션 호출 전에 참이 되야 할 조건

2) 결과 조건 : 오퍼레이션 수행된 후에 참이 되야 할 조건

3) 불변 조건 : 오퍼레이션 실행 동안 항상 참이 되야 할 조건

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

22주차 - 아키텍처 패턴

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 아키텍처 패턴의 개요

1) 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제

2) 기본적인 윤곽 제시

3) 서브시스템과 역할이 정의되어 있고 서브 시스템 사이 관계, 규칙 포함

4) 아키텍처 스타일, 표준 아키텍처

5) 아키텍처 패턴의 장점

- 시행착오를 줄여 개발 시간 단축, 고품질

- 검증되었으므로 안정적 개발

- 의사소통 간편

- 구조 이해 쉬움, 시스템 특성 개발 전에 예측 가능

6) 레이어 패턴, 클라이언트-서버 패턴, 파이프-필터 패턴, 모델-뷰-컨트롤러 패턴

2. 레이어 패턴

1) 계층으로 구분하는 고전적 방법

2) 각 서브시스템들이 계층 구조를 이루고 하위 계층은 상위 계층에 대한 서비스 제공자가 됨

3) 상위 계층은 하위 계층의 클라이언트가 됨

4) 서로 마주보는 두 개 계층 사이에서만 상호작용함, 두 개만 영향을 미쳐 변경 작업 쉬움

5) 특정 계층만 교체해 시스템 개선 가능

6) ex) OSI 참조 모델

3. 클라이언트-서버 패턴

1) 하나의 서버 컴포넌트와 다수 클라이언트 컴포넌트로 구성

- 컴포넌트 : 실행 코드 기반 모듈

2) 사용자는 클라이언트와만 의사소통 함

3) 서버는 클라이언트 요청에 대비해 항상 대기 상태 유지

4) 클라이언트나 서버는 요청 응답 시 동기화를 제외하고 독립적

4. 파이프-필터 패턴

1) 데이터 스트림 절차의 각 단계를 필터 컴포넌트로 캡슐화하여 파이프로 데이터 전송

- 데이터 스트림 : 데이터가 송수신되거나 처리되는 일련의 연속적 흐름

2) 재사용성, 추가가 쉬워 확장 용이

3) 필터 컴포넌트 재배치 다양한 파이프라인 구축 가능

4) 데이터 변환, 버퍼링, 동기화에 사용

5) 필터 간 데이터 이동 시 데이터 변환으로 인한 오버헤드 발생

6) ex) UNIX 쉘

5. 모델-뷰-컨트롤러 패턴

1) 서브시스템을 3개 부분으로 구조화

- 모델 : 서브시스템의 핵심 기능과 데이터 보관

- 뷰 : 사용자에게 정보 표시

- 컨트롤러 : 입력된 변경 요청 처리하기 위해 모델에게 명령

2) 각 부분이 별도의 컴포넌트로 분리되어 있음

3) 여러개 뷰 만들 수 있어서 대화형 애플리케이션에 적합

6. 기타

1) 마스터-슬레이브 패턴

- 마스터 컴포넌트는 동일한 구조의 슬레이브 컴포넌트로 작업 분할

- 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받음

- 마스터 컴포넌트 : 주체, 슬레이브 컴포넌트 : 지시를 받아 작업 수행

- 장애 허용 시스템, 병렬 컴퓨팅 시스템

2) 브로커 패턴

- 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커가 맞는 컴포넌트와 유저 연결해줌

- 원격 서비스 호출에 응답하는 컴포넌트가 여러 개 있을 때 적합

- 분산 환경 시스템

3) 피어-투-피어 패턴

- 피어를 하나의 컴포넌트로 간주, 각 피어는 클라이언트, 서버가 될 수 있음

- 멀티 스레딩 방식 : 자원을 공유하며 병렬로 수행하는 기능

4) 이벤트-버스 패턴

- 소스가 특정 채널에 이벤트 메시지를 발행하면 해당 채널을 구독한 리스너들이 메세지를 받아 처리하는 방식

- 소스 : 이벤트 생성

- 리스너 : 이벤트 수행

- 채널 : 이벤트 통로

- 버스 : 채널 관리

5) 블랙보드 패턴

- 모든 컴포넌트가 공유 데이터 저장소와 블랙보드 컴포넌트에 접근 가능

- 컴포넌트는 검색을 통해 블랙보드에서 원하는 데이터를 찾음

- 해결책이 명확하지 않은 문제 처리

- 음성 인식, 차량 식별, 신호 해석

6) 인터프리터 패턴

- 프로그램 코드의 각 라인을 수행하는 방법 지정, 기호마다 클래스를 갖도록 구성

- 특정 언어로 작성된 프로그램 코드를 해석하는 컴포넌트 설계할 때 사용

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

23주차 - 객체 지향

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 객체지향의 개요

1) 현실 세계 개체를 하나의 객체로 만듦

2) 구조적 기법의 문제점으로 인한 소프트웨어 위기의 해결책으로 채택되어 사용됨

- 구조적 기법 : 프로시저에 근간을 두고 작업을 세부적으로 나누고 만들어서 다시 합치는 것

3) 재사용, 확장, 유지보수 쉬움

4) 단계적, 계층적으로 표현

5) 객체, 클래스, 캡슐화, 상속, 다형성, 연관성

2. 객체

1) 모듈

- 데이터

- 객체의 정보, 속성, 상태, 분류

- 함수

- 수행 기능

- 데이터 처리하는 알고리즘

2) 독립적으로 식별 가능한 이름이 있음

3) 객체가 가질 수 있는 조건을 상태라고 함, 일반적으로 시간에 따라 변함

4) 메시지의 집합을 행위라고 함

3. 클래스

1) 공통된 속성과 연산을 갖는 객체의 집합, 일반적인 타입, 틀

2) 데이터를 추상화하는 단위

3) 클래스에 속한 객체를 인스턴스라고 하고 객체를 만드는 것을 인스턴스화라고 함

4) 슈퍼 클래스 : 특정 클래스의 부모 클래스, 서브 클래스 : 특정 클래스의 자식 클래스가

4. 캡슐화

1) 데이터와 데이터를 처리한느 함수를 하나로 묶는 것

2) 정보 은닉, 외부 모듈의 변경으로 인한 파급 효과 적음

3) 재사용 쉬움

4) 내부 내용을 알 필요 없으므로 인터페이스 단순해지고 객체 간 결합도 낮아짐

5. 상속

1) 부모 클래스의 모든 속성과 연산을 물려받는 것

2) 재사용을 높임

3) 다중 상속 : 한 개의 클래스가 두개 이상의 상위 클래스로부터 상속받는 것(부모가 2개이상)

6. 다형성

1) 연산 수행할 때 각각의 클래스가 가지고 있는 고유한 방법으로 응답할 수 있는 능력

2) 동일한 메소드명을 사용하여 같은 의미의 응답을 함

3) 오버로딩 : 메소드 이름 동일, 인수 자료형 개수로 구별

오버라이딩 : 상위 클래스에서 정의한 메소드와 이름은 같지만 실행 코드를 달리해 자식 클래스에서 재정의

7. 연관성

1) 두 개 이상의 객체가 상호 참조하는 관계

- is member of : 연관화(Association) -> 2개 이상의 객체가 상호 관련

- is instance of : 분류화(Classfication) -> 동일한 형의 특성을 갖는 객체들을 모아 구성

- is part of : 집단화(Aggregation) -> 관련 있는 객체 묶어 하나의 상위 객체 구성

- is a : 일반화(Generalization) -> 공통적인 성질들로 추상화한 상위 객체 구성

특수화(Specialization) -> 상위 객체를 구체화하여 하위 객체 구성

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

24주차 - 객체 지향 분석 및 설계

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 객체 지향 분석의 개념

1) 사용자의 요구사항을 분석하여 모든 클래스, 연관된 속성과 연산, 관계 등을 정의하여 모델링하는 작업

2) 객체는 클래스로부터 인스턴스화되고, 이 클래스를 식별하는 것이 객체지향 분석의 목적

2. 객체지향 분석의 방법론

1) 럼바우 방법

- 가장 일반적인 방법, 객체 모델, 동적 모델, 기능 모델로 나누어 수행

2) 부치 방법

- 미시적 개발 프로세스와 거시적 개발 프로세스를 모두 사용

3) Jacobson 방법

- Use Case 강조하여 사용하는 분석 방법

4) Coad와 Yourdon 방법

- E-R 다이어그램, 객체 행위 모델링

5) Wirfs-Brock 방법

- 분석과 설계 간의 구분이 없음

- 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행

3. 럼바우의 분석 기법 (객동기)

1) 모든 소프트웨어 구성요소를 그래픽 표기법을 이용해 모델링

2) 객체 모델링 기법(OMT)

3) 객체 모델링 -> 동적 모델링 -> 기능 모델링

- 객체 모델링

- 정보 모델링, 객체들 간 관계를 규정하여 객체 다이어그램으로 표현

- 동적 모델링

- 상태 다이어그램을 이용해 시간에 흐름에 따른 객체들의 동적인 행위를 표현

- 기능 모델링

- 자료 흐름도(DFD)를 이용하여 자료 흐름을 중심으로 처리 과정 표현

4. 객체지향 설계 원칙 (SOLID 원칙, 단일,개방,리스코프,인터페이스,의존)

1) 단일 책임 원칙

- 객체는 단 하나의 책임만 가짐

- 응집도 높고, 결합도 낮게 설계

2) 개방-폐쇄 원칙

- 기존 코드 변경하지 않고 기능 추가할 수 있도록 설계

- 공통 인터페이스를 하나의 인터페이스로 묶어 캡슐화하는 방법이 대표적

3) 리스코프 치환 원칙

- 자식 클래스는 최소한 자신의 부모 클래스에서 가능한 행위는 수행할 수 있어야 한다

- 자식 클래스는 부모 클래스의 책임을 무시하거나 재정의하지 않고 확장만 수행

4) 인터페이스 분리 원칙

- 자신이 사용하지 않는 인터페이스와 의존관계나 영향을 받으면 안됨

- 인터페이스가 갖는 하나의 책임

5) 의존 역전 원칙

- 객체들 간의 의존관계가 성립될 때 추상성 낮은 클래스보다 추상이 높은 클래스와 의존 관계를 맺어야 함

- 일반적으로 인터페이스를 활용하면 이 원칙은 준수됨

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

25주차 - 모듈

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 모듈의 개요

1) 모듈화를 통해 분리된 시스템의 각 기능

- 루틴 : 기능을 가진 명령들의 모임

- 메인 루틴 : 프로그램 실행의 큰 줄기가 되는 것

- 서브 루틴 : 메인 루틴에 의해 필요할 때 마다 호출되는 루틴

2) 단독으로 컴파일 가능, 재사용 가능

3) 독립성을 높이려면 결합도는 약하게 응집도는 강하게 모듈의 크기는 작게 만들어야 됨

2. 결합도(Coupling)(자스 제외 공통내용) 짜스 제외하고 공통내용이야!!

1) 모듈 간에 상호 의존하는 정도

2) 자료 결합도, 스탬프 결합도, 제어 결합도, 외부 결합도, 공통 결합도, 내용 결합도

<- 결합도 약함 결합도 강함 ->

- 자료 결합도

- 모듈 간의 인터페이스가 자료 요소로만 구성

- 모듈 간 내용을 전혀 알 필요 없고 서로에게 영향이 전혀 없음

- 스탬프 결합도

- 모듈 간 인터페이스로 배열, 레코드 등 자료 구조가 전달될 때

- 두 모듈이 동일한 자료 구조를 조회하는 경우

- 제어 결합도

- 다른 모듈의 내부의 논리적인 흐름을 제어하기 위해 제어 신호를 전달하는 결합도

- 한 모듈이 다른 모듈의 상세한 처리 과정을 알아 이를 통제하는 경우나 처리 기능이 두 모듈에 분리되어 설계된 경우

- 하위 모듈이 상위 모듈에게 명령을 내리는 권리 전도 현상 발생

- 외부 결합도

- 어떤 모듈에서 선언한 데이터를 다른 모듈에서 참조할 떄 결합도

- 참조되는 데이터 범위를 각 모듈에서 제한할 수 있다.

- 공통 결합도

- 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때 결합도

- 공통 데이터 영역을 조금만 변경해도 여러 모듈에 영향을 미침

- 내용 결합도

- 다른 모듈의 내부를 직접 참조하거나 수정 할 때 결합도

- 한 모듈에서 다른 모듈 내부로 제어가 이동하는 경우

3. 응집도(Cohesion) (기능순차 교환절차 시 논리우연) 기능순차와 교환절차 시 논리가 우연이야!!

1) 정보 은닉 개념 확장, 모듈이 독립적인 기능으로 정의되어 있는 정도

2) 기능적 응집도, 순차적 응집도, 교환적 응집도, 절차적 응집도, 시간적 응집도, 논리적 응집도, 우연적 응집도

<- 응집도 강함 응집도 약함 ->

- 기능적 응집도

- 모듈 내부 모든 기능 요소들이 단일 문제와 연관

- 순차적 응집도

- 모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용

- 교환적 응집도

- 동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우

- 절차적 응집도

- 모듈이 다수의 관련 기능을 가질 때 순차적으로 수행할 경우

- 시간적 응집도

- 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우

- 논리적 응집도

- 유사한 성격을 갖는 요소들로 하나의 모듈이 형성되는 경우

- 우연적 응집도

- 서로 관련 없는 요소로만 구성된 경우

4. 팬인/팬아웃

1) 팬인 (화살표 안으로 들어옴)

- 어떤 모듈을 호출하는 모듈의 수

2) 팬아웃 (화살표 밖으로 나감)

- 어떤 모듈에 의해 호출되는 모듈의 수

3) 시스템 복잡도 알 수 있음

4) 팬인이 높으면 재사용 좋지만, 단일 장애점이 발생할 수 있음

5) 팬아웃이 높으면 불필요하게 다른 모듈을 호출하는 지 검토 후 단순화 여부 검토

6) 복잡도를 최적화하려면 팬인은 높게, 팬아웃은 낮게 설계해야 함

5. N-S 차트

1) 논리의 기술에 중점을 둔 도형을 이용한 표현 방법

2) 박스 다이어그램, Chapin Chart

3) 연속, 선택 및 다중 선택, 반복 등의 제어 논리 구조 표현

4) goto나 화살표 사용 안함

5) 조건이 복합된 곳의 처리, 선택,반복 시각적으로 명확히 식별

6) 이해하기 쉽고, 코드 변환 쉬움

7) 읽기는 쉽지만 작성이 어렵고 임의로 제어를 전이할 수 없음

8) 총체적인 구조 표현과 인터페이스를 나타내기 어려움

9) 단일 입구, 단일 출구로 표현

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

26주차 - 공통 모듈

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 공통 모듈의 개요(정확명확완전일추) 정확하게 명확하게 완전 일추!!!

1) 여러 프로그램에서 공통적으로 사용할 수 있는 모듈

2) 자주 사용되는 기능들이 공통모듈로 구성됨

3) 재사용성 확보와 중복 개발 회피를 위해 설계 과정에서 공통 부분을 식별하고 명세를 작성

- 정확성 : 해당 기능이 필요하다는 것을 알도록 정확히 작성

- 명확성 : 중의적으로 해석되지 않도록 명확하게 작성

- 완전성 : 시스템 구현을 위해 필요한 모든 것을 기술

- 일관성 : 공통 기능들 간 상호 충돌이 발생하지 않도록 작성

- 추적성 : 요구사항의 출처, 관계를 파악할 수 있도록 작성

2. 재사용(함컴 애플)

1) 최적화, 누구나 이해할 수 있고 사용 가능하도록 사용법 공개해야 함

2) 재사용 규모에 따른 분류

- 함수와 객체 : 클래스나 메소드 단위의 소스 코드를 재사용

- 컴포넌트 : 독립적인 기능을 수행하는 실행 코드 기반 모듈, 수정 없이 인터페이스를 통해 통신하는 방식

- 애플리케이션 : 공통된 기능 제공하는 애플리케이션을 공유하는 방식으로 재사용

3. 효과적인 모듈 설계 방안

1) 결합도 줄이고 응집도 높여 모듈의 독립성과 재사용성 높임

2) 모듈의 제어 영역 안에서 모듈의 영향 역역을 유지

3) 하나의 입구, 하나의 출구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

27주차 - 코드

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 코드의 개요

1) 정보를 신속 정확 명료하게 전달

2) 코드의 기능(식사 분배 표준 간소) 식사 분배를 표준, 저는 간소하게 주세요!!

- 식별 기능 : 데이터 간의 성격에 따라 구분

- 분류 기능 : 특정 기준이나 동일한 유형에 해당하는 데이터를 그룹화

- 배열 기능 : 의미를 부여하여 나열

- 표준화 기능 : 다양한 데이터를 기준에 맞추어 표현

- 간소화 기능 : 복잡한 데이터를 간소화

2. 코드의 종류(순블십그연표합)순 블록이 열그룹 있네 연상이랑 표 합치자!

1) 순차 코드

- 일정 기준에 따라 최초 자료부터 차례로 일련번호 부여

- 순서코드, 일련번호 코드

- ex) 1,2,3

2) 블록 코드

- 공통성 있는 것끼리 블록으로 구분 블록 내에서 일련번호 부여

- 구분 코드

- ex) 1001~1100, 1101~1200

3) 10진 코드

- 10만큼 계속 나눠서 분류, 도서 분류식 코드

- ex) 1000, 1100,1110

4) 그룹 분류 코드

- 대분류, 중분류, 소분류

- ex) 1-01-001

5) 연상 코드

- 명칭이나 약호, 문자 기호를 이용해 코드 부여

- ex) TV-40

6) 표의 숫자 코드

- 물리적 수치를 코드에 적용, 유효 숫자 코드

- ex) 120-720-1500

7) 합성 코드

- 2개 이상의 코드를 조합

- ex) KE-711

3. 코드 부여 체계

1) 이름만으로 개체의 용도와 적용 범위를 알 수 있도록 코드 부여

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

28주차 - 디자인 패턴

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 디자인 패턴의 개요

1) 각 모듈의 세분화된 역할이나 모듈들 간의 인터페이스 같은 코드 작성 수준의 세부적인 구현 방안을 설계할 때

참조할 수 있는 전형적인 해결 방식 또는 예제

2) 새로 해결책 구상 보단 문제에 해당하는 디자인 패턴을 참고

3) 변형을 가하면 유사한 형태의 다른 패턴으로 변화되는 특징

4) GoF(Gang of Four) 가 구체화 및 체계화

5) 생성패턴 5개, 구조 패턴 7개, 행위 패턴 11개 총 23개 패턴(생구행)

2. 디자인 패턴 사용 장단점

1) 범용적 코딩 스타일로 인해 구조파악 쉬움

2) 생산성 높임, 비용 시간 절약

3) 초기 투자 비용이 부담스러울 수 있음

4) 객체 지향 기반이므로 다른 기반의 애플리케이션 개발에는 적합하지 않음

3. 생성 패턴 (추상훈 보디빌더 팩하다가 프로 실패해서 싱글됨)

1) 객체의 생성과 참조 과정을 캡슐화하여 유연성을 더해줌

- 추상 팩토리

- 구체적인 클래스에 의존하지 않고 인터페이스를 통해 서로 의존하는 객체들의 그룹으로 생성하여 추상적으로 표현

- 연관된 서브클래스를 묶어 한번에 교체 가능

- 빌더

- 작게 분리된 인스턴스를 건축 하듯이 조합하여 객체를 생성

- 객체의 생성과정과 표현방법을 분리, 동일한 객체 생성에서도 서로 다른 결과 만들수 있음

- 팩토리 메소드

- 객체 생성을 서브 클래스에서 처리하도록 분리하여 캡슐화

- 상위 클래스는 인터페이스만 정의 실제 생성은 서브 클래스가

- 가상 생성자 패턴

- 프로토타입

- 원본 객체 복제

- 비용이 큰 경우 사용

- 싱글톤

- 어디서든 참조할 수 있지만 한 번에 하나만 가능

- 클래스 내 인스턴스가 하나뿐임을 보장, 메모리 낭비 최소화

4. 구조 패턴 (어브바 컴퓨터 (됐고)데코 프로퍼플 줘)

1) 클래스나 객체를 조합하여 더 큰 구조로 만들 수 있게 해주는 패턴

2) 구조가 복잡한 시스템을 개발하기 쉽게 도와줌

- 어댑터

- 호환성 없는 클래스의 인터페이스를 다른 클래스가 이용할 수 있도록 변환

- 기존의 클래스를 이용하고 싶지만 인터페이스가 일치하지 않을 때 이용

- 브리지

- 구현부에서 추상층 분리, 서로가 독립적으로 확장할 수 있도록 구성한 패턴

- 기능과 구현을 두 개 별도의 클래스로 구현

- 컴포지트

- 여러 객체를 가진 복합 객체와 단일 객체를 구분 없이 다룰 때

- 트리 구조로 구성하여 표현

- 데코레이터

- 객체 간 결합을 통해 능동적으로 기능 확장

- 덧붙이는 것

- 퍼싸드(Facade)

- 복잡한 서브 클래스를 피해 더 상위에 인터페이스를 구성하여 서브 클래스 기능을 쉽게 사용

- 서브 클래스 사이 통합 인터페이스를 제공하는 Wrapper 객체 필요

- 플라이웨이트

- 인스턴스를 매번 생성하지 않고 가능한 한 공유해서 사용해 메모리 절약

- 다수의 유사 객체를 생성하거나 조작할 때 유용

- 프록시

- 접근이 어려운 객체에 연결할 때 인터페이스 역할 수행

- 네트워크 연결, 메모리의 대용량 객체로의 접근에 이용

5. 행위 패턴 (책임이 커! 인터밀란 반복중 메멘 옵서 상전 템 사러 방문)

1) 서로 상호작용하는 방법, 책임 분배 방법

2) 하나의 객체로 수행할 수 없는 작업을 여러 객체로 분배하여 결합도 최소화하도록 도와줌

- 책임 연쇄

- 요청 처리 가능한 객체가 2개 이상, 한 객체가 처리 못하면 다음 객체로 넘어감

- 객체들이 고리로 묶여 있음, 요청 해결될 때까지 고리를 따라 책임 전가

- 커맨드

- 캡슐화하여 재사용하거나 취소할 수 있도록 요청에 정보 저장 또는 로그에 남기는 패턴

- 각종 명령어를 추상 클래스와 구체 클래스로 분리하여 단순화

- 인터프리터

- 언어에 문법 표현을 정의, SQL 개발할 때 사용

- 반복자

- 접근이 잦은 객체에 동일한 인터페이스 적용

- 내부 노출 없이 순차적 접근 가능

- 중재자

- 상호작용을 캡슐화하여 객체로 정의

- 의존성을 줄여 결합도 감소

- 객체 간의 통제와 지시 역할 수행

- 메멘토

- 특정 시점에서 객체 내부 상태를 객체화, Ctrl + Z 기능

- 옵서버

- 객체의 상태가 변화하면 상속되어 있는 다른 객체들에게 상태 전달

- 분산 시스템에서 이벤트 생성발행, 수신할 때 이용

- 상태

- 객체의 상태에 따라 동일한 동작을 다르게 처리

- 객체 상태를 캡슐화하여 참조하는 방식

- 전략

- 동일한 계열의 알고리즘을 개별적으로 캡슐화하여 상호 교환

- 클라이언트는 독립적으로 원하는 알고리즘 선택, 영향 없이 알고리즘 변경 가능

- 템플릿 메소드

- 상위 클래스에서 골격 정의, 하위 클래스에서 세부 처리 구체화

- 유사한 서브 클래스를 묶어 공통된 내용 상위 클래스에서 정의, 코드의 양 줄임 유지보수 쉬움

- 방문자

- 각 클래스들의 데이터 구조에서 처리 기능 분리하여 별도의 클래스로 구성

- 분리된 처리 기능은 각 클래스를 방문하여 수행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

30주차 - 인터페이스 요구사항 검증

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 요구사항 검증

1) 설계 및 구현 전에 요구사항 명세서 검토하고 개발 범위 기준인 베이스라인 설정

2) 요구사항 검토 계획 수립 -> 검토 및 오류 수정 -> 베이스라인 설정

2. 인터페이스 요구사항 검토 계획 수립

1) 프로젝트 품질 관리 계획 참조하여 검토

- 검토 기준 및 방법

- 프젝 규모, 참여 인력, 검토 기간 등 고려

- 참여자

- 검토 참여자 선정

- TA : 기술 아키텍처 설계 구축

- SA : 소프트웨어 아키텍처 설계 구축

- 체크리스트

- 완전성, 일관성, 명확성 체크 가능한 체크리스트

- 관련 자료

- 검토에 필요한 자료 준비

- 일정

- 검토 일정 정하기

3. 인터페이스 요구사항 검토 및 오류 수정

1) 검토 체크리스트 항목에 따라 명세서 검토

2) 오류 발견 시 수정할 수 있ㄷ록 오류 목록과 시정 조치서 작성

3) 검토 결과를 검토 관련자들에게 전달

4. 인터페이스 요구사항 베이스라인 설정

1) 주요 의사 결정자에게 공식적으로 승인 받기

5. 요구사항 검증 방법

1) 요구사항 검토

- 수작업으로 분석하는 방법

- 동료검토 : 작성자가 직접 설명, 동료가 들으면서 결함 발견

- 워크스루 : 회의 전에 미리 배포하여 사전 검토 후 회의에서 결함 발견

- 인스펙션 : 작성자를 제외한 다른 검토 전문가들이 확인하여 결함 발견

2) 프로토타이핑

- 요구사항 정확히 파악 위해 견본품 만들어 최종 결과물 예측

3) 테스트 설계

- 요구사항은 테스트 할 수 있도록 작성되야 됨

- 테스트 케이스 생성 이후 테스트 가능한지 검토

4) CASE 도구 활용

- 일관성 분석을 통해 요구사항 변경사항 추적, 관리

6. 인터페이스 요구사항 검증의 주요 항목(완전 일등 명기인데 검추로 변경할게요)

1) 완전성

- 모든 요구사항 누락 않고 완전 반영

2) 일관성

- 요구사항이 모순 충돌 없이 일관성 유지

3) 명확성

- 모든 참여자가 요구사항을 명확히 이해 가능?

4) 기능성

- 요구사항이 어떻게 보단 무엇을에 중점

5) 검증 가능성

- 요구 모두 만족, 요구 내용과 일치하는지 검증 가능?

6) 추적 가능성

- 명세서와 설계서 추적 가능?

7) 변경 용이성

- 명세서 변경 쉽도록 작성됨?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

33주차 - 인터페이스 방법 명세화

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 인터페이스 방법 명세화 개념

1) 내외부 시스템이 연계하여 작동할 때 송수신방법,데이터 등 내용을 문서로 정리하는 것

2) 시스템 연계 기술, 인터페이스 통신 유형, 처리 유형, 발생 주기 등 정보 필요

2. 시스템 연계 기술

1) 시스템끼리 연계할 떄 사용되는 기술

2) DB Link, API/Open API, 연계 솔루션, Socket, Web Service

- DB Link : DB에서 제공하는 DB Link 객체 이용

- API/Open API : 송신 시스템의 DB에서 데이터 읽어 제공하는 애플리케이션 프로그래밍 인터페이스 프로그램

- 연계 솔루션 : EAI(송수신데이터 식별하여 통제) 서버와 송수신시스템에 설치되는 클라이언트 이용

- Socket : 서버는 통신을 위한 소켓 생성해 포트 할당하고 클라이언트 통신 요청 시 연결하여 통신

- Web Service : 웹 서비스에서 WSDL, UDDI, SOAP 프로토콜 이용

3. 인터페이스 통신 유형

1) 시스템끼리 데이터 송수신하는 형태

2) 단방향, 동기, 비동기

- 단방향 : 시스템에서 거래 요청만 하고 응답 없음

- 동기 : 시스템에서 거래 요청하고 응답 올 때까지 대기

- 비동기 : 시스템에서 거래 요청하고 다른 작업 수행하다 응답 오면 처리

4. 처리 유형

1) 송수신 데이터를 어떤 형태로 처리

2) 업무 성격과 데이터 전송량 고려하여 실시간 방식, 지연 처리 방식, 배치 방식

- 실시간 방식 : 사용자 요청을 바로 처리

- 자연 처리 방식 : 데이터를 매건 단위로 처리할 경우 비용이 많이 발생할 때 사용

- 배치 방식 : 대량의 데이터 처리할 때 사용

5. 발생 주기

1) 송수신데이터가 전송되어 인터페이스가 사용되는 주기

2) 업무 성격, 데이터 전송량을 고려하여 매일, 수시, 주 1회 등으로 구분

6. 송수신 방법 명세화

1) 시스템 인터페이스 설계 시 작성한 아키텍처 정의서 기반으로 작성

7. 송수신 데이터 명세화

1) 인터페이스별로 테이블 정의서와 파일 레이아웃에서 연계하고자 하는 테이블, 파일 단위로 작성

8. 오류 식별 및 처리 방안 명세화

1) 각 인터페이스 대한 오류 식별, 처리방안 작성

2) 송신 시스템

- 데이터를 전송 형식에 맞게 변환 후 송신

3) 연계 서버

- 송수신 시스템 사이에 위치 현황 모니터링

4) 수신 시스템

- 처리 가능한 형식으로 변환 후 연계 프로그램에 반영

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

35주차 - 미들웨어 솔루션 명세

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 미들웨어의 개념

1) 분산 컴퓨팅 환경에서 사이에서 원만한 통신 이루어지도록 서비스 제공

2) 하드웨어, 프로토콜, 운영체제와 응용프로그램, 서버와 클라이언트

3) 표준화된 인터페이스 제공하여 일관성 보장

4) 위치 투명성

5) 내부 동작 확인하려면 별도 응용 소프트웨어 필요

6) 1:1, 1:N, N:M 형태

7) DB, RPC, MOM, TP-Monitor, ORB, WAS

2. DB(DataBase)

1) 클라이언트에서 원격의 데이터베이스와 연결하기 위한 미들웨어

2) DB를 이용해 구축한 시스템은 2-Tier 아키텍처 라 함

3) 마이크로소프트의 ODBC, 볼랜드의 IDAPI, 오라클의 Glue

3. RPC(Remote Procedure Call)

1) 원격 프로시저 호출

2) 응용 프로그램의 프로시저를 사용해 원격 프로시저를 로컬 프로시저처럼 호출

3) 이큐브시스템스의 Entera, OSF의 ONC/RPC

4. MOM(Message Oriented Middleware)

1) 메세지 기반 비동기형 메세지 전달

2) 온라인 업무보다 이기종 분산 데이터 시스템의 데이터 동기를 위해 사용

3) 독립적으로 실행되는 소프트웨어 간 상호 작용을 통해 하나의 통합된 시스템처럼 동작하게 만듦

4) IBM의 MQ, 오라클의 Message Q, JCP의 JMS

5. TP-Monitor(Transaction Processing Monitor)

1) 항공기나 철도 예약 업무 같은 온라인 트랜잭션 업무에서 트랙잭션 처리 감시

2) 사용자 증가해도 빠른 응답 속도 유지해야 하는 경우 사용

3) 오라클의 tuxedo, 티맥스소프트의 tmax

6. ORB(Object Request Broker)

1) 객체 요청 브로커

2) 객체 지향 미들웨어 코바 표준 스택 구현

- CORBA : 네트워크에서 분산 프로그램 객체 생성,관리하기 위한 규격

3) TP-Monitor 장점인 트랜잭션 처리와 모니터링 추가한 제품도 있음

4) Micro Focus의 Orbix, OMG의 CORBA

7. WAS(Web Application Service)

1) 동적인 콘텐츠 처리

2) 클라이언트/서버 보단 웹 환경 구현

3) JABA, EJB 컴포넌트 기반 구현

- EJB : 클라이언트 서버 모델의 서버 부분에서 자바 컴포넌트들을 설정하기 위한 아키텍처, 대규모 분산 환경 쉽게 구현

4) 오라클의 WebLogic, IBM의 WebSphere

8. 미들웨어 솔루션 식별

1) 시스템, 구분, 솔루션명, 버전, 제조사 등의 정보 목록 작성

9. 미들웨어 솔루션 명세서 작성

1) 솔루션별로 관련 정보를 상세하게 기술

2) 솔루션들을 검토하여 제약사항 환경 기능 등에 대해 명세서 작성