**1과목 소프트웨어 설계**

1장 요구사항 확인

001 소프트웨어 생명 주기

소프트웨어 생명주기 – 소프트웨어를 개발하기 위해 정의하고 운용, 유지보수 등의 과정을 단계별로 나눈 것

폭포수 모형

-> 각 단계를 확실히 하고 결과를 철저하게 검토하여 승인을 거친 후 다음 단계 진행

프로토타입 모형

-> 사용자의 요구사항 정확히 파악

나선형 모형

-> 폭포수 + 프로토타입 + 위험 분석 기능

애자일 모형

-> 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기 반복하면서 개발과정 진행

002 스크럼 기법

스크럼 -> 팀이 중심이 되어 개발의 효율성을 높임

- 제품 책임자, 스크럼 마스터, 개발팀

스크럼 개발 프로세스

제품 백로그 -> 스프린트 계획 회의 -> 스프린트 -> 일일 스크럼 회의 -> 스프린트 검토 회의

-> 스프린트 회고

003 xp(eXtreme Programming) 기법

xp -> 고객의 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향산시킴

xp의 5가지 핵심 가치 : 의사소통, 단순성, 용기, 존중, 피드백

xp 개발 프로새스

• 사용자 스토리

• 릴리즈 계획 수립

• 스파이크

• 이터레이션(주기)

• 승인 검사

• 소규모 릴리즈

004 현행 시스템 파악

• 시스템 구성 파악 -> 기간 업무 / 지원 업무

• 시스템 기능 파악 -> 주요기능 / 하부기능 / 세부기능

• 시스템 인터페이스 파악 -> 데이터 종류, 형식, 프로토콜, 연계유형, 주기

• 아키텍처 구성 파악 -> 어떠한 기술 요소들이 사용되는지 최상위 수준에서 계층별로 표현한 아키텍쳐 구성도 작성

• 소프트웨어 구성 파악 -> 제품명 / 용도 / 라이선스 적용 방식 / 라이선스 수

• 하드웨어 구성 파악 -> 서버의 사양 / 수량 / 이중화 적용 여부

• 네트워크 구성 파악 -> 서버의 위치 / 서버 간의 네트워크 연결 방식

005 개발 기술 환경 파악

운영체제 / 데이터베이스관리시스템 / aws(미들웨어)

-> 각각의 고려사항 확인

006 요구사항 정의

유형

기술하는 내용 : 기능 요구사항 vs 비기능 요구사항

기술 관점과 대상의 범위 : 시스템 요구사항 vs 사용자 요구사항

프로세스 : 도출 -> 분석 -> 명세 -> 확인

도출 : 요구사항이 어디에 있는지, 어떻게 수집할 지 식별하고 이해하는 과정

분석 : 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 걸러내기 위한 과정

명세 : 요구사항을 바탕으로 모델을 작성하고 문서화

확인 : 요구사항 명세서가 정확하고 완전하게 작성되었는지 검토

007 요구사항 분석

소프트웨어 개발의 실질적인 첫 단계, 사용자의 요구사항을 이해하고 문서화하는 활동

- 자료 흐름도

- 자료 사전

= : 자료의 정의 / + : 자료의 연결 / ( ) : 자료의 생략 /

[ | ] : 자료의 선택 / { } : 자료의 반복 / \* \* : 자료의 설명

008 요구사항 분석 CASE와 HIPO

CASE : 요구사항을 자동으로 분석하고 요구사항 분석 명세서를 기술하도록 개발된 도구

- SADT / SREM / RSL / REVS / PSL/PSA / TAGS

HIPO : 시스템의 분석 및 설계나 문서화할 때 사용되는 기법, 시스템 실행 과정인 입력, 처리, 출력의 기능

- 가시적 도표(도식 목차) / 총체적 도표(총괄도표, 개요 도표) / 세부적 도표(상세 도표)

009 UML(Unified Modeling Language)

UML : 시스템 개발자와 고객, 개발자 상호간의 의사소통이 원할하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어

- 사물 / 관계 / 다이어그램

사물 -> 구조 사물, 행동 사물, 그룹 사물, 주해 사물

관계 -> 연관 관계, 집합 관계, 포함 관계, 일반화 관계. 의존 관계, 실체화 관계

다이어그램 -> 구조적 : 클래스, 객체, 컴포넌트, 배치, 복합체 구조, 패키지 다이어그램

행위 : 유스케이스, 순차, 커뮤니케이션, 상태, 활동, 상호작용 개요, 타이밍 다이어그램

010 주요 UML 다이어그램

유스케이스 다이어그램

사용자와 외부 시스템들이 개발될 시스템을 이용해 수행 가능한 기능을 사용자 관점에서 표현

시스템 범위 / 액터 / 유스케이스 / 관계

클래스 다이어그램

시스템을 구성하는 클래스

클래스 / 제약조건 / 관계

순차 다이어그램

시간의 흐름에 따라 상호 작용하는 과정을 여러가지 요소를 사용하여 그림으로 표현

액터 / 객체 / 생명선 / 실행 상자 / 메시지

문제 정리

폭포수 모형 – 각 단계를 매듭짓고 다음 단계를 진행하는 개발 방법론

프로토타입 모형 – 사용자의 요구사항을 정확히 파악하기 위해 sw에 대한 견본품을 만들어 최종 결과물을 예측하는 모형

나선형 모형 – 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 sw를 개발함

소프트웨어 공학 패러다임 – 폭포수 모형, 프로토타입 모형, 나선형 모형, 4세대 기법,

애자일 모형

애자일은 계획을 따르기 보다 변화에 반응하는 것에 가치를 둠 / 문서화보다는 실행되는 sw에 가치를 둠

4세대 기법 – 사용자와 개발자가 쉽게 접근하고 사용할 수 있는 4세대 언어를 사용하는 모형

XP 5가지 핵심 가치 – 의사소통, 단순성, 용기, 존중, 피드백

폭포수 모형 개발 순서 : 타당성 검토 -> 계획 -> 요구 분석 -> 설계 -> 구현 -> 시험

-> 유지보수

데이터 흐름도의 구성 요소 : 프로세스. 자료 흐름, 자료 저장소, 단말

구조적 다이어그램 : 클래스, 객체, 컴포넌트, 배치, 복합체 구조, 패키지 다이어그램

소프트웨어 구현 단계 : 작업 계획 -> 코딩 -> 컴파일 -> 테스트

소프트웨어 공학은 소프트웨어의 품질과 생산성 향상을 목적으로 함

단계 다이어그램은 물리 화학 등에서 사용되는 다이어그램으로 요구사항 모델링과는 관계가 없음

AVL 다이어그램은 이진 탐색 트리에서 활용되는 다이어그램임

공학적으로 잘 작성된 소프트웨어의 특징 : 사용자 요구사항 충족, 높은 신뢰성, 유지 보수의 용이성, 쉬운 인터페이스, 충분한 테스팅 수행

모델링은 분석 및 설계 단계에서 개발하고자 하는 것을 시작적으로 표현한 것

위험 분석 – 개발 과정에 포함

애자일은 개발 방법론 / 구조적 기반 X

요구사항 분석은 사용자의 요구가 수시로 발생할 수 있어 열거와 구조화 어려움

도메인 분석은 요구에 대한 정보를 수집하고 배경을 분석하여 이를 토대로 모델링을 하게 됨

설계 명세서 작성은 설계 과정에서 수행

순차 다이어그램 = 교류 다이어그램

키워드 : 1. 폭포수 모형 2. 애자일 선언 3. XP(eXtreme Programming) 4. 요구사항 분석

5. 자료 흐름도 6. 자료 사전 7. 관계 8. 다이어그램 종류 9. 유스케이스 다이어그램

10. 순차 다이어그램

2장 화면 설계

011 사용자 인터페이스

사용자 인터페이스 – 사용자와 시스템 간의 상호작용이 원활하게 이뤄지도록 도와주는 장치나 소프트웨어

CLI / CUI / NUI / VUI / OUI

기본원칙 – 직관성, 유효성, 학습성, 유연성

설계지침 – 사용자 중심, 사용성, 일관성, 단순성, 결과 예측 가능, 가시성, 심미성, 표준화, 접근성, 명확성, 오류 발생 해결

012 UI 표준 및 지침

UI 표준 : 전체 시스템에 포함된 모든 UI에 공통적으로 적용될 내용으로 화면 구성이나 화면 이동 포함

UI 지침 : UI 요구사항, 구현 시 제약사항 등 UI 개발 과정에서 꼭 지켜야 할 공통의 조건을 의미

한국형 웹 콘텐츠 접근성 지침 : 장애인이 비장애인과 동등하게 접근할 수 있는 웹 콘텐츠의

제작방법 제시

-> 웹 콘텐츠 접근성(사용성) 지침 준수를 위한 고려사항

전자정부 웹 표준 준수 지침 : 정부기관의 홈페이지 구축 시 반영해야 할 최소한의 규약을 정의

013 UI 설계 도구

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

종류 -> 1. 와이어프레임, 2. 목업, 3. 스토리보드, 4. 프로토타입, 5. 유스케이스

1. 페이지에 대한 개략적인 레이아웃 등 기초적인 뼈대를 설계하는 단계

2. 와이어프레임보다 좀 더 실제 화면과 유사하게 만든 정적인 형태의 모형

3. 와이어프레임에 설명, 페이지 간 이동 흐름 등을 추가한 문서

4. 와이어프레임이나 스토리보드에 인터랙션을 적용함으로 테스트가 가능한 동적인 형태의 모형

5. 사용자 측면에서의 요구사항으로 사용자가 원하는 목표를 달성하기 위해 수행할 내용 기술

014 UI 요구사항 확인

새로 개발할 시스템에 적용할 UI 관련 요구사항을 조사해서 작성하는 단계

목표정의 -> 활동 사항 정의 -> UI 요구사항 작성 -> 요구사항 요소 확인 -> 정황 시나리오 작성 -> 요구사항 작성

015 품질 요구사항

소프트웨어 품질 : 소프트웨어의 기능, 성능, 만족도 등 소프트웨어에 대한 요구사항이 얼마나 충족하는가를 나타내는 소프트웨어 특성의 총체임

ISO/IEC 9126 : 기능성, 신뢰성, 사용성, 효율성, 유지 보수성, 이식성

ISO/IEC 25010 : 기능 적합성, 성능 효율성, 호환성, 사용성, 신뢰성, 보안성, 유지 보수성, 이식성

기능성 : 소프트웨어가 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지 여부를 나타냄 – 적절성/적합성, 정밀성/정확성. 상호 운용성, 보안성, 준수성

신뢰성 : 소프트웨어가 요구된 기능을 정확하고 일관되게 오류 없이 수행할 수 있는 정도를 나타냄 – 성숙성, 고장 허용성, 회복성

사용성 : 사용자와 컴퓨터 사이에 발생하는 어떠한 행위에 대하여 사용자가 쉽게 배우고 사용할 수 있으며, 향후 다시 사용하고 싶은 정도를 나타냄 – 이해성, 학습성, 운용성, 친밀성

효율성 : 사용자가 요구하는 기능을 할당된 시간 동안 한정된 자원으로 얼마나 빨리 처리할 수 있는지 정도를 나타냄 – 시간 효율성, 자원 효율성

유지 보수성 : 환경의 변화 또는 새로운 요구사항이 발생했을 때 소프트웨어를 개선하거나 확장할 수 있는 정도를 나타냄 – 분석성, 변경성, 안정성, 시험성

이식성 : 소프트웨어가 다른 환경에서도 얼마나 쉽게 적용할 수 있는지 정도를 나타냄 – 적용성, 설치성, 대체성, 공존성

016 UI 프로토타입 제작 및 검토

프로토타입은 사용자 요구사항을 기반으로 실제 동작하는 것처럼 만든 동적인 형태의 모형으로 테스트가 가능함

프로토타입 종류 : 페이퍼 프로토타입 / 디지털 프로토타입

UI 프로토타입 계획 및 작성 시 고려사항 : 계획 시 고려 사항 / 작성 시 고려사항

UI 프로토타입 제작 단계 : 1단계 ~ 4단계

017 UI 설계서 작성

UI 설계서는 사용자의 요구사항을 바탕으로 UI 설계를 구체화하여 작성하는 문서로, 상세 설계 전에 대표적인 화면들을 설계함

UI 설계서 표지 – 다른 문서와 혼동되지 않도록 프로젝트명 또는 시스템명을 포함시켜 작성

UI 설계서 개정 이력 작성 – UI 설계서가 수정될 때마다 어떤 부분이 어떻게 수정되었는지를 정리해 놓은 문서

UI 요구사항 정의서 작성 – 사용자의 요구사항을 확인하고 정리한 문서로, 사용자 요구사항의 UI 적용 여부를 요구사항별로 표시

시스템 구조 작성 – UI 요구사항과 UI 프로토타입에 기초하여 전체 시스템의 구조를 설계한 것으로 사용자의 요구사항이 어떻게 시스템에 적용되는지 알 수 있음

사이트 맵 작성 – 시스템 구조를 바탕으로 사이트에 표시할 콘텐츠를 한 눈에 알아 볼 수 있도록 메뉴별로 구분하여 설계한 것

프로세스 정의서 작성 – 사용자 관점에서 사용자가 요구하는 프로세스들을 작업 진행 순서에 맞춰 정리한 것으로 UI의 전체적인 흐름을 파악할 수 있음

화면 설계 – UI 프로토타입과 UI 프로세스를 참고하여 필요한 화면을 페이지별로 설계한 것

018 유용성 평가

유용성 – 사용자가 시스템을 통해 원하는 목표를 얼마나 효과적으로 달성할 수 있는가에 대한 척도로 UI의 주된 목적은 유용성이 뛰어난 UI를 제작하는 것

실행 차를 줄이기 위한 UI 설계 원리 검토

1. 사용 의도 파악 2. 행위 순서 규정 3. 행위의 순서대로 실행

평가 차를 줄이기 위한 UI 설계 원리 검토

1. 수행한 키 조작의 결과를 사용자가 빠르게 지각하도록 유도

2. 키 조작으로 변화된 시스템의 상태를 사용자가 쉽게 인지하도록 유도

3. 사용자가 가진 원래 의도와 시스템 결과 간의 유사 정도를 사용자가 쉽게 파악하도록 유도

019 UI 상세 설계

-> UI 설계서를 바탕으로 실제 설계 및 구현을 위해 모든 화면에 대한 자세한 설계를 진행하는 단계

UI 시나리오 문서 작성 원칙

UI 시나리오 문서 작성을 위한 일반 규칙

-> 주요 키의 위치와 기능 / 공통 내 요소 / 기본 스크린 레이아웃 / 기본 인터랙션 규칙 / 공통 단위 태스크 흐름 / 케이스 문서

UI 시나리오 문서의 요건

-> 완전성 / 일관성 / 이해성 / 가독성 / 수정 용이성 / 추적 용이성

UI 시나리오 문서로 인한 기대 효과

- 요구사항이나 의사소통에 대한 오류가 감소한다

- 개발 과정에서의 재작업이 감소하고, 혼선이 최소화된다

- 불필요한 기능을 최소화한다

- 소프트웨어 개발 비용을 절감한다

- 개발 속도를 향상시킨다

UI 요소

체크 박스 / 라디어 버튼 / 텍스트 박스 / 콤보 상자 / 목록 상자

020 HCI / UX / 감성공학

HCI : 사람이 시스템을 보다 편리하고 안전하게 사용할 수 있도록 연구하고 개발하는 학문

최종목표는 시스템을 사용하는데 최적의 사용자 경험을 만드는 것

UX : 사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하게 되는 총체적인 경험을 말함

주관성 / 정황성 / 총체성

감성공학 : 제품이나 작업환경을 사용자의 감성에 알맞도록 설계 및 제작하는 기술로 인문하회과학, 공학, 의학 등 여러 분야의 학문이 공존하는 종합과학

-> 기반 기술 / 구현 기술 / 응용 기술

문제 정리

GUI : 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스

OUI : 모든 사물과 사용자 간의 상호작용을 위한 인터페이스

CUI : 명령과 출력이 텍스트 형태로 이루어지는 인터페이스

사용자 인터페이스는 사용자의 편리성과 가독성을 높임으로써 작업 시간을 단축시키고 업무에 대

한 이해도를 높여준다

템플릿 – 일정한 모양으로 만들어진 틀 / 형판, 보기 관

목업 – 실제 화면과 유사하지만 실제로는 구현되지 않은 정적인 형태의 도형

아파치 – 서버 프로그램

Posture : 센서가 부착된 장갑을 통해 손가락의 움직임 정보를 다루는 기법

Module : 모듈화를 통해 분리된 시스템의 각 기능들로, 서브루틴, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위 등과 같은 의미로 사용됨

Hash : 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것을 의미

ISO/IEC 9126 품질 특성 – 기능성 / 신뢰성 / 사용성 / 효율성 / 유지 보수성 / 이식성

기능성 – SW가 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지 여부를 나타냄

신뢰성 – SW가 요구된 기능을 정확하고 일관되게 오류 없이 수행할 수 있는 정도를 나타냄

프로토타입은 사용자의 요구사항을 개발자가 맞게 해석했는지 검증하기 위한 것으로 최대한 간단하게 만드는 것이 좋음

UI 설계서는 상세 설계 이전에 작성함

웹 콘텐츠 지침 사항 -> 소리 자동 재생 X

정황 시나리오는 사용자 관점에서 작성

프로토타이핑은 프로젝트의 실제 분석 작업이 완료되기 전 작성

현재 시스템의 변화는 사용자가 직관적으로 알 수 있도록 표시해야 함

키워드 : 1. 사용자 인터페이스의 구분 2. 사용자 인터페이스의 기본 원칙 3. 사용자 인터페이스의 설계 지침 4. 목업 5. 품질 요구사항 6. 기능성 7. 신뢰성 8. 사용성 9. 이식성 10. UI 요소

3장 애플리케이션 설계

021 소프트웨어 아키텍처

SW를 구성하는 요소들 간의 관계를 표현하는 시스템 구조

- 1. 모듈화 2. 추상화 3. 단계적 분해 4. 정보은닉

1. SW의 성능을 향상시키거나 시스템의 수정 및 재사용, 유지 관리가 용이하도록 시스템의 기능들을 모듈 단위로 나눈 것

2. 문제의 포괄적인 개념을 세분화하여 구체화 시키는 것

-> 과정 추상화 / 데이터 추상화 / 제어 추상화

3. 문제를 상위의 중요 개념부터 하위의 개념으로 구체화시킴

4. 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법

소프트웨어 아키텍처 품질 속성

-> 시스템 측면 / 비즈니스 측면 / 아키텍처 측면

소프트웨어 아키텍처 설계 과정

설계 목표 설정 -> 시스템 타입 결정 -> 아키텍처 패턴 적용 -> 서브시스템 구체화 -> 검토

022 아키텍처 패턴

아키텍처를 설계 할 때 참조할 수 있는 전형적인 해결 방식

1. 레이어 패턴 2. 클라이언트-서버 패턴 3. 파이프-필터 패턴 4. 모델-뷰-컨트롤러 패턴

1. 하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴

2. 데이터 스트림 절차의 각 단계를 필터 컴포넌트로 전송화하여 파이프를 통해 데이터를 전송하는 패턴

3. 서브시스템을 3개의 부분으로 구조화하는 패턴

기타 패턴

마스터-슬레이브 / 브로커 / 피어-투-피어 / 이벤트-버스 / 블랙보드 / 인터프리터 패턴

023 객체지향(Object-Oriented)

- 객체들을 조립해서 작성할 수 있는 기법

1. 객체 -> 데이터와 데이터를 처리하는 함수를 묶어 놓은 하나의 소프트웨어 모듈

2. 클래스 -> 공통된 속성과 연산을 갖는 객체의 집합

3. 캡슐화 -> 데이터와 데이터를 처리하는 함수를 하나로 묶은 것

4. 상속 -> 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려 받는 것

5. 다형성 -> 하나의 메시지에 객체가 가지고 있는 고유한 방법으로 응답할 수 있는 능력

6. 연관성 -> 두 개 이상의 객체들이 상호 참조하는 관계

024 객체지향 분석 및 설계

- 사용자의 요구사항을 분석하여 문제와 관련된 클래스, 속성과 연산, 관계 등을 정의하여 모델링

Booch 방법 -> 미시적 개발 프로세스 / 거시적 개발 프로세스

Jacobson 방법 -> Use Case 강조

Coad와 Yourdon 방법 -> E-R 다이어그램

Wirfs-Brock 방법 -> 고객 명세서 평가

Rumbaugh 방법 -> 객체 모델 / 동적 모델 / 기능 모델

객체 모델링 -> 동적 모델링 -> 기능 모델링

객체지향 설계 원칙

단일책임 원칙 / 개방-폐쇄 원칙 / 리스코프 치환 원칙 / 인터페이스 분리 원칙 / 의존 역전 원칙

025 모듈

- 모듈화를 통해 분리된 시스템의 각 기능

결합도 -> 모듈 간의 상호 의존하는 정도, 두 모듈 사이의 연관관계

약함 -> 강함

자료 / 스탬프 / 제어 / 외부 / 공통 / 내용 /

응집도 -> 모듈의 내부 요소들 서로 관련

강함 -> 약함

기능적 / 순차적 / 교환적 / 절차적 / 시간적 / 논리적 / 우연적

팬인 -> 모듈을 제어하는 모듈의의 수

팬 아웃 -> 모듈에 의해 제어되는 모듈의 수

026 공통 모듈

- 여러 프로그램에서 공통적으로 사용할 수 있는 모듈

-> 정확성 / 명확성 / 완전성 / 일관성 / 추적성

재사용 -> 이미 개발된 기능을 파악하고 재구성하여 시스템이나 기능 개발에 사용하기 적합하도록 최적화 시키는 작업

027 코드

분류, 조합 및 집계를 용이하개 하고 자료의 추출을 쉽게 하기 위해 사용하는 기호

식별 기능 / 분류 기능 / 배열 기능 / 표준화 기능 / 간소화 기능

코드 종류

순차 코드 / 블록 코드 / 10진 코드 / 그룹 분류 코드 / 연상 코드 / 표의 숫자 코드 / 합성코드

028 디자인 패턴

- 모듈의 세부적인 구현 방안을 설계할 때 참조할 수 있는 해결 방식 또는 예제

생성패턴 – 프로그램에 유연성을 더해줌

-> 추상 팩토리 / 빌더 / 팩토리 메소드 / 프로토타입 / 싱글톤

구조 패턴 – 구조가 복잡한 시스템을 개발하기 쉽게 도와줌

-> 어댑터 / 브리지 / 컴포지트 / 데코레이터 / 퍼싸드 / 플라이웨이트 / 프록시

행위 패턴 – 하나의 객채로 할 수 없는 작업을 여러 객체로 분배하면서 결합도를 최소화 할 수 있도록 도와줌

-> 책임 연쇄 / 커맨드 / 인터프리터 / 반복자 / 중재자 / 메먼토 / 옵서버 / 상태 / 전략 / 템플릿 메소드 / 방문자

문제 정리

메소드 – 객체가 수행하는 기능으로 객체가 갖는 데이터를 처리하는 알고리즘

설계된 모듈을 프로그래밍 언어를 통해 구현 – 아키텍처 구현

ISP – 자신이 사용하지 않는 인터페이스와 의존 관계를 맺거나 영향을 받지 않아야 하는 원칙

DIP – 객체들간의 의존관계가 성립될 때, 추상성이 높은 클래스와 의존 관계를 맺어야 하는 원칙

SRP – 객체는 단 하나의 책임만 가져야 한다는 원칙

MVC 패턴 – 서브시스템을 3개의 부분으로 구조화하는 패턴

피어투피어패턴 – 피어를 하나의 ㅓㅁ포넌트로 간주하며, 각 피어는 서비스를 호출하는 클라이언트가 될 수도, 서비스를 제공하는 서버가 될 수 있는 패턴

브로커패턴 – 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 요청에 맞는 컴포넌트와 사용자를 연결해줌

객체는 일정항 기억장소 가짐

트랜잭션 – 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위

시퀸스 – 특정 시간동안 수행되는 사건이나 행동 드으이 순서

서브 루틴 – 메인 루틴에 의해 필요할 때마다 호출되는 루틴

Inheritance – 이미 정의된 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려 받는 것

Class – 공통된 속성과 연산을 갖는 객체의 집합

Association – 2개 이상의 객체가 상호 관련되어 있음을 의미

제어 결합도 – 어떤 모듈이 다른 모듈 내부틔 논리적인 흐름을 제어하기 위해 제어 신호를 이용하여 통신하거나 제어 요소를 전달하는 결합도

공통 결합도 – 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도

스탬프 결합도 – 모듈 간의 인터페이스로 배열이나 레코드 등의 자료구조가 전달될 때의 결합도

기능적 응집도 – 모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도

순차적 응집도 – 모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도

논리적 응집도 – 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도

규모에 따른 재사용 대상 – 함수, 객체, 컴포넌트, 애플리케이션

공통 부분을 명세하기 위한 기법 – 정확성, 명확성, 완전성, 일관성, 추적성

컴포지트 패턴 – 여러 객체를 가진 복합 객체와 단일 객체를 구분 없이 다루고자 할 때 사용하는 패턴

퍼싸드 패턴 – 복잡한 서브 클래스들을 피해 더 상위에 인터페이스를 구성함으로써 서브 클래스들의 기능을 간편하게 사용할 수 있도록 하는 패턴

브리지 패턴 – 구현부에서 추상층을 분리하여 서로가 독립적으로 확장할 수 있도록 구성한 패턴

추상 팩토리 – 인터페이스를 통해 서로 의존, 연관되는 객체들의 그룹으로 생성하여 추상적으로 표현하는 패턴

인터프리터 패턴 – 언어에 문법 표현을 정의하는 패턴

데코레이터 패턴 – 객체 간의 결합으로 능동적으로 기능들을 확장할 수 있는 패턴

어댑터 패턴 – 호환성이 없는 클래스들의 인터페이스를 다른 클래스가 잉용할 수 있도록 변환해주는 패턴

플라이웨이트 – 가능한 공유해서 사용함으로써 메모리를 절약하는 패턴

분할 정복/분할 통치(Divide and Couper) : 큰 문제를 보다 작은 문제로 분할하여 해결하는 전략

동적 계획법(Dynamic Programming) : 아래 단계의 간단한 문제부터 해결하면서 상위로 나아가는 상향식 접근 방식

탐욕 알고리즘(Greedy Algorithm) : 완벽한 해결책보다는 차선책을 목표로 하며, 상황에 맞는 해결책을 즉석에서 모색하는 방식

백트래킹(Backtracking) : 깊이우선탐색 알고리즘을 이용한 기법으로 문제 해결을 위한 모든 가능성을 트리로 구축하는 방식

햐향식 방식 – 절차지향적 분석 기법에 적합

객체지향 설계 : 객체지향 설계 원칙을 기반으로 하여 소프트웨어를 설계하는 방식

데이터 흐름 설계 : 프로세스와 프로세스 간의 데이터 흐름에 중점을 두고 소프트웨어를 설계하는 방식

상향식 설계 : 최하위 수준에서 각각의 모듈을 설계하고 완성된 모듈을 결합해가면서 설계하는 방식

키워드 : 1. 파이프-필터 패턴 2. 캡슐화 3. 럼바우의 분석 기법 4. 객체지향 설계 원칙 5. 결합도 6. 응집도 7. 효과적인 모듈 설계 방안 8. 생성 패턴 9. 구조 패턴 10. 행위 패턴

4장 인터페이스 설계

029 시스템 인터페이스 요구사항 분석

구성 – 독립적인 시스템들끼리 연동하여 상호 작용하기 위한 접속 방법이나 규칙을 의미

1. 인터페이스 이름 2. 연계 대상 시스템 3. 연계 범위 및 내용 4. 송신 데이터 5. 연계방식

6. 인터페이스 주기 7. 기타 고려사항 -> 요구사항 명세서

분석 – 기능적, 비기능적 요구사항으로 분류하고 조직화하여 요구사항 명제를 구체화하고 이해관계자에게 전달하는 과정

요구사항 선별 -> 요구사항 관련 자료 준비 -> 요구사항 분류 -> 요구사항 분석 및 명세서 구체화 -> 요구사항 명세서 공유 - 분석 절차

030 인터페이스 요구사항 검증

- 사용자들의 요구사항이 요구사항 명세서에 정확하고 완전하게 기술되었는지 검토하고 개발 범위의 기준인 베이스라인을 설정하는 것

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

요구사항 검증 방법

요구사항 검토 -> 동료검토 / 워크스루 / 인스펙션

프로토타이핑 / 테스트 설계 / CASE

요구사항 검증의 주요 항목 : 완전성 / 일관성 / 명확성 / 기능성 / 검증 가능성 / 추적 가능성 / 변경 용이성

031 인터페이스 시스템 식별

개발 시스템 식별 – 개발하고자 하는 시스템의 상세 식별 정보를 정의하고 목록을 작성하는 것

내•외부 시스템 식별 – 개발할 시스템과 연계할 내•외부 시스템들의 상세 식별 정보를 정의하고 목록을 작성하는 것

내•외부 시스템 환경 및 관리 주체 식별- IP, URL, PORT 정보 같은 실제 운용 환경 의미

내•외부 시스템 네트워크 연결 정보 식별 – 시스템 로그인 및 DB 정보 의미

인터페이스 시스템 식별 – 인터페이스에 참여하는 시스템들을 송신, 수신 시스템으로 구분하여 작성하는 것

032 송 • 수신 데이터 식별

식별 대상 데이터 – 송•수신 시스템 사이에서 교환되는 데이터

인터페이스 표준 항목 –> 시스템 공통부 / 거래 공통부

송•수신 데이터 항목 / 공통항목

정보 흐름 식별 – 개발할 시스템과 내•외부 시스템 사이에서 전송되는 정보들의 방향성을 식별하는 것

송•수신 데이터 식별 – 데이터 산출물을 기반으로 송•수신 데이터 식별

인터페이스 표준 항목과 송•수신 데이터 항목 식별 / 코드성 데이터 항목 식별

033 인터페이스 방법 명세화

인터페이스별로 송•수신 방법, 송•수신 데이터, 오류 식별 및 처리 방안에 대한 내용을 문서로 명확하게 정리하는 것

SOCKET : 서버는 통신을 위한 소켓을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하여 통신하는 네트워크 기술

인터페이스 통신 유형

단방향 : 시스템에서 거래를 요청만 하고 응답이 없는 방식

동기 : 시스템에서 거래를 요청하고 응답이 올 때까지 대기하는 방식

비동기 : 시스템에서 거래를 요청하고 다른 작업을 수행하다 응답이 오면 처리하는 방식

인터페이스 처리 유형

실시간 방식 : 사용자가 요청한 내용을 바로 처리해야 할 때 사용하는 방식

지연 처리 방식 : 데이터를 매건 단위로 처리할 경우 비용이 많이 발생할 때 사용하는 방식

배치 방식 : 대량의 데이터를 처리할 때 사용하는 방식

송•수신 방법 명세화 – 각각의 인터페이스에 대해 연계방식, 통신 및 처리 유형, 발생 주기 등 송•수신 방법을 정의하고 명세를 작성하는 것

송•수신 데이터 명세화 – 각각의 인터페이스에 대해 인터페이스 시 필요한 송•수신 데이터에 대한 명세를 작성하는 것

오류 식별 및 처리 방안 명세화 – 각각의 인터페이스에 대해 인터페이스 시 발생할 수 있는 오류를 식별하고 오류 처리 방안에 대해 명세를 작성하는 것

034 시스템 인터페이스 설계서 작성

- 시스템의 인터페이스 현황을 확인하기 위해 시스템이 갖는 인터페이스 목록과 각 인터페이스의 상세 데이터 명세를 정의한 문서

시스템 인터페이스 목록 – 업무 시스템과 내•외부 시스템 간 데이터를 주고받는 경우에 사용하는 인터페이스에 대해 기술한 것

시스템 인터페이스 정의서 작성 – 시스템 간의 연계를 위해 필요한 데이터 항목 및 구현 요건 등을 기술하는 것

035 미들웨어 솔루션 명세

- 미들과 소프트웨어의 합성어

- 응용 프로그램, 서버, 클라이언트 사잉에서 원만한 통신이 이루어지도록 다양한 서비스를 제공

- 시스템 간의 데이터 교환에 일관성을 보장함

DB – 클라이언트에서 원격의 데이터베이스와 연결하기 위한 미들웨어

RPC – 응용 프로그램의 프로시저를 사용하여 원격 프로시저를 로컬 프로시저처럼 호출하는 방식

MOM – 메시지 기반의 비동기형 메시지를 전달하는 방식

TP-Monitor – 온라인 트랜잭션 업무에서 트랜잭션을 처리 및 감시하는 미들웨어

ORB – 객체 지향 미들웨어로 코바 표준 스펙을 구현한 미들웨어

WAS – 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어