**1장 요구사항 확인**

001 소프트웨어 생명 주기

- 소프트웨어를 개발하기 위한 설계, 운용, 유지보수 등의 과정을 각 단계별로 나눈 것임

폭포수 모형 / 프로토타입 모형 / 나선형 모형 / 애자일 모형

폭포수 모형 – 각 단계를 확실히 매듭짓고 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 개발 방법론

프로토타입 모형 – 실제 개발될 소프트웨어에 대한 견본품을 만들어 최종 결과물을 예측하는 모형

나선형 모형 – 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발하는 모형

계획수립 -> 위험분석 -> 개발 및 검증 -> 고객평가

애자일 모형 – 요구사항 변하에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발하는 모형

스크럼 / XP / 칸반 / Lean / 기능 중심 개발

소프트웨어 공학 – 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문

002 스크럼 기법

스크럼 – 팀이 중심이 되어 개발의 효율성을 높이는 기법

스크럼 팀 – 제품 책임자 / 스크럼 마스터 / 개발팀

스프린트 계획 회의 – 이번 스프린트에서 수행할 작업을 대상으로 단기 일정을 수립하는 회의

스프린트 – 실제 개발 작업을 진행하는 과정

일일 스크럼 회의 - 15분동안 진행상황을 점검하는 회의

스프린트 검토 회의 – 부분 또는 전체 완성 제품이 요구사항에 잘 부합하는지 테스팅하는 회의

스프린트 회고 – 정해놓은 규칙 준수 여부 및 개선할 점을 확인하고 기록하는 것

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 개발 기술 환경 파악

운영체제 – 컴퓨터 시스템의 자원을 효율적으로 관리하며 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어

고려사항 : 가용성 / 성능 / 기술 지원 / 주변 기기 / 구축 비용

데이터베이스 관리 시스템 – 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어

고려사항 : 성능 / 기술 지원 / 상호 호환성 / 구축비용

웹 애플리케이션 서버 – 동적인 콘텐츠를 처리하기 위한 미들웨어

고려사항 : 가용성 / 성능 / 기술 지원 / 구축 비용

오픈 소스 – 누구나 별다른 제한 없이 사용할 수 있도록 소스 코드를 공개한 소프트웨어

고려사항 : 라이선스 종류 / 사용자 수 / 기술의 지속 가능성

005 요구사항 정의

요구사항 – 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영되는데 필요한 제약조건

유형 : 기능 / 비기능 / 사용자 / 시스템

기능 요구사항 : 시스템이 무엇을 하는지, 어떤 기능을 하는지 등의 기능이나 수행과 관련된 요구사항

비기능 요구사항 : 품질이나 제약사항과 관련된 요구사항

사용자 요구사항 : 사용자 관점에서 본 시스템이 제공해야 할 요구사항

시스템 요구사항 : 개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항

006 요구사항 개발 프로세스

- 개발 대상에 대한 요구사항을 체계적으로 도출하고 분석한 후 명세서에 정리한 다음 확인 및 검증하는 일련의 구조화된 활동

도출 -> 분석 -> 명세 -> 확인

요구사항 도출 : 시스템, 사용자, 개발자 등 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항을 어떻게 수집할 것인지를 식별하고 이해하는 과정

요구사항 분석 : 사용자 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정

요구사항 명세 : 분석된 요구사항을 바탕으로 모델을 작성하고 문서화하는 것

요구사항 확인 : 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동

요구공학 – 요구사항을 정의하고 분석 및 관리하는 프로세스를 연구하는 학문

007 요구사항 분석

- 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화하는 활동

구조적 분석 기법 – 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법

구조적 분석 기법 도구 : 자료 흐름도 / 자료 사전 / 소단위 명세서 / 개체 관계도 / 상태 전이도 / 제어 명세서

자료 흐름도 – 자료의 흐름 및 변환 과정과 기능을 도형중심으로 기술하는 방법

프로세스 : 자료를 변환시키는 시스템의 한 부분을 나타내며 처리, 기능, 변환, 버블

자료 흐름 : 자료의 이동이나 연관관계를 나타냄

자료 저장소 : 시스템에서의 자료저장소

단말 : 시스템과 교신하는 외부 개체

자료 사전 – 자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것

008 요구사항 분석 CASE와 HIPO

요구사항 분석용 CASE - 요구사항을 자동으로 분석하고 요구사항 분석 명세서를 기술하도록 개발된 도구

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

HIPO - 시스템 실행 과정인 입력 처리 출력의 기능을 표현한 것

가시적 도표 / 총제적 도표 / 세부적 도표

009 UML의 개요

- 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호 간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어

구성 요소 : 사물 / 관계 / 다이어그램

사물 : 다이어그램 안에서 관계가 형성될 수 있는 대상들

구조 사물 / 행동 사물 / 그룹 사물 / 주해 사물

010 UML - 관계

- 사물과 사물 사이의 연관성을 표현하는 것

종류 : 연관 / 집합 / 포함 / 일반화 / 의존 / 실체화

연관 관계 - 2개 이상의 사물이 서로 관련되어 있는 관계

집합 관계 - 하나의 사물이 다른 사물에 포함되어 있는 관계

포함 관계 - 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계

일반화 관계 - 하나의 사물이 다른 사물에 비해 더 일반적이거나 구체적인 관계

의존 관계 - 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계

실체화 관계 - 사물이 할 수 있거나 해야 하는 기능으로 서로를 그룹화 할 수 있는 관계

011 UML - 다이어그램

- 사물과 관계를 도형으로 표현한 것

구조적 다이어그램

클래스 다이어그램 / 객체 다이어그램 / 컴포넌트 다이어그램 / 배치 다이어그램 / 복합체 구조 다이어그램 / 패키지 다이어그램

행위 다이어그램

유스케이스 다이어그램 / 순차 다이어그램 / 커뮤니케이션 다이어그램 / 상태 다이어그램 / 활동 다이어그램 / 상호작용 개요 다이어그램 / 타이밍 다이어그램

스테레오 타입 - UML에서 표현하는 기본 기능 외에 추가적인 기능을 표현하는 것

012 유스케이스 다이어그램

기능 모델링 - 개발된 시스템이 갖춰야 할 기능을 정리한 후 사용자와 함께 정리된 내용을 공유하기 위해 그림으로 표현하는 것

유스케이스 다이어그램 / 액티비티 다이어그램

유스케이스 다이어그램 - 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점에서 표현한 것

유스케이스 다이어그램의 구성 요소

시스템 / 시스템 범위 / 액터 / 유스케이스 / 관계

013 활동 다이어그램

사용자의 관점에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현한 것

구성 요소 - 액션/액티비티 / 시작 노드 / 종료 노드 / 조건 노드 / 병합 노드 / 포크 노드 / 조인 노드 / 스윔레인

014 클래스 다이어그램

정적 모델링 - 사용자가 요구한 기능을 구현하는데 필요한 자료들의 논리적인 구조를 표현하는 것

클래스 다이어그램 - 클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한 것

구성요소 - 클래스 / 제약조건 / 관계

연관 클래스 - 연관 관계에 있는 두 클래스에 추가적으로 표현해야 할 속성이나 오퍼레이션이 있는 경우 생성하는 클래스

015 순차 다이어그램

동적 모델링 – 시스템의 내부 구성 요소들의 상태 변화 과정과 과정에서 발생하는 상호 작용을 표현한 것

동적 모델링 종류 – 순차 다이어그램 / 커뮤니케이션 다이어그램 / 상태 다이어그램

순차 다이어그램 – 시스템이나 객체들이 메시지를 주고 받으며 상호 작용하는 과정을 그림으로 표현한 것

구성 요소 – 액터 / 객체 / 생명선 / 실행 상자 / 메시지 / 객체 소멸 / 프레임

016 커뮤니케이션 다이어그램

- 시스템이나 객체들이 메시지를 주고받으며 상호작용하는 과정과 객체들 간의 연관을 그림으로 표현한 것

구성 요소 – 액터 / 객체 / 링크 / 메시지

017 상태 다이어그램

- 객체들 사이에 발생하는 이벤트에 의한 객체들의 상태 변화를 그림으로 표현한 것

객체의 상태 – 객체가 갖는 속성 값의 변화를 의미

구성 요소 : 상태 / 시작 상태 / 종료 상태 / 상태 전환 / 이벤트 / 프레임

018 패키지 다이어그램

- 유스케이스나 클래스 등의 요소들을 그룹화한 패키지간의 의존 관계를 표현한 것

- 패키지는 또 다른 패키지의 요소가 될 수 있음

구성 요소 : 패키지 / 객체 / 의존 관계

019 소프트웨어 개발 방법론

- 소프트웨어 개발, 유지보수 등에 필요한 여러 가지 일들의 수행 방법과 각종 기법 및 도구를 체계적으로 정리하여 표준화한 것

구조적 방법론 / 정보공학 방법론 / 객체지향 방법론 / 컴포넌트 기반 방법론 / 제품 계열 방법론 / 애자일 방법론

구조적 방법론 – 사용자 요구사항을 파악하여 문서화하는 처리 중심의 방법론

타당성 검토 단계 -> 계획 단계 -> 요구사항 단계 -> 설계 단계 -> 구현 단계 -> 시험 단계 -> 운용/유지보수 단계

정보공학 방법론 – 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료 중심의 방법론

정보 전략 계획 수립 단계 -> 업무 영역 분석 단계 -> 업무 시스템 설계 단계 -> 업무 시스템 구축 단계

객체지향 방법론 – 개체를 하나의 객체로 만들어 객체들을 조립해서 필요한 소프트웨어를 구현하는 방법론

요구 분석 단계 -> 설계 단계 -> 구현 단계 -> 테스트 및 검증 단계 -> 인도 단계

컴포넌트 기반 방법론 – 컴포넌트를 조합하여 하나의 새로운 애플리케이션을 만드는 방법론

개발 준비 단계 -> 분석 단계 -> 설계 단계 -> 구현 단계 -> 테스트 단계 -> 전개 단계 -> 인도 단계

제품 계열 방법론 – 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론

020 S/W 공학의 발전적 추세

소프트웨어 재사용 – 이미 개발되어 인정받은 소프트웨어를 다른 소프트웨어 개발이나 유지에 사용하는 것

합성 중심 / 생성 중심

소프트웨어 재공학 – 기존 시스템을 이용하여 보다 나은 시스템을 구축하고 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것

이점

소프트웨어 품질 향상 / 생산성 증가 / 수명 연장 / 오류 감소

CASE – 요구분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것

주요 기능

소프트웨어 생명 주기 전 단계의 연결 / 다양한 소프트웨어 개발 모형 지원 / 그래픽 지원

021 비용 산정 기법 – 하향식

- 과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 비과학적인 방법

전문가 감정 기법 / 델파이 기법

전문가 감정 기법 – 조직 내에 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법

델파이 기법 – 전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법

022 비용 산정 기법 – 상향식

- 프로젝트의 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비용을 산정하는 방법

LOC 기법 / 개발 단계별 인월수 기법 / 수학적 산정 기법

LOC 기법 – 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 비용을 산정하는 기법

노력(인월) = 개발기간 X 투입 인원

= LOC / 1인당 월평균 생산 코드 라인 수

개발 비용 = 노력 X 단위 비용

개발 기간 = 노력 / 투입 인원

생산성 = LOC / 노력

개발 단계별 인월수 기법 – 기능을 구현시키는 데 필요한 노력을 생명 주기의 각 단계별로 산정

023 수학적 산정 기법

- 상향식 비용 산정 기법으로 경험적 추정 모형, 실험적 추정 모형이라고 함

주요 수학적 산정 기법 – COCOMO / Putnam 모형 / 기능 점수 모형

COCOMO – LOC에 의한 비용 산정 기법

개발 유형 - 조직형 / 반분리형 / 내장형

모형 종류 – 기본형 / 중간형 / 발전형

Putnam – 소프트웨어 생명 주기의 전 과정 도안에 사용될 노력의 분포를 예상하는 모형

기능 점수 모형 – 소프트웨어의 기능을 증대시키는 요인별로 가중치를 부여하고 요인별 가중치를 합산하여 총 기능 점수를 산출하며 기능점수를 구한 후 비용을 산정하는 기법

증대 요인 – 자료 입력 / 정보 출력 / 명령어 / 데이터 파일 / 필요한 외부 루틴과의 인터페이스

자동화 도구 – SLIM / ESTIMACS

024 프로젝트 일정 계획

- 프로세스를 이루는 소작업을 파악하여 예측된 노력을 각 소작업에 분배하여 소작업의 순서와 일정을 정하는 것

PERT – 전체 작업의 상호 관계를 표시하는 네트워크

CPM – 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는데 사용하는 기법

간트 차트 – 프로젝트의 각 작업들이 언제 시작하고 언제 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표

025 소프트웨어 개발 방법론 결정

- 프로젝트 관리와 재사용 현황을 소프트웨어 개발 방법론에 반영하고 확정된 소프트웨어 생명 주기와 개발 방법론에 맞춰 소프트웨어 계발 단계, 활동, 작업, 절차 등을 정의

절차

1. 프로젝트 관리와 재사용 현황을 소프트웨어 개발 방법론에 반영

2. 개발 단계별 작업 및 절차를 소프트웨어 생명 주기에 맞춰 수립

3. 결정된 소프트웨어 개발 방법론의 개발 단계별 활동 목적, 작업 내용, 산출물에 대한 매뉴얼 작성

프로젝트 관리 – 최소의 비용으로 사용자를 만족시키는 시스템을 개발하기 위한 전반적인 활동

일정 관리 / 비용 관리 / 인력 관리 / 위험 관리 / 품질 관리

026 소프트웨어 개발 표준

- 소프트웨어 개발 단계에서 수행하는 품질 관리에 사용되는 국제 표준

주요 소프트웨어 개발 표준

ISO/IEC 12207 / CMMI / SPICE

SIO/IEC 12207 - ISO에서 만든 표준 소프트웨어 생명주기 프로세스

기본 생명 주기 프로세스 / 지원 생명 주기 프로세스 / 조직 생명 주기 프로세스

CMMI – 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델

성숙도 – 초기 -> 관리 -> 정의 -> 정량적 관리 -> 최적화

SPICE – 소프트웨어의 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준

고객-공급자 / 공학 프로세스 / 지원 프로세스 / 관리 프로세스 / 조직 프로세스

SPICE의 프로세스 수행 능력 단계

불완전 / 수행 / 관리 / 확립 / 예측 / 최적화

027 소프트웨어 개발 방법론 테일러링

- 프로젝트 상황 및 특성에 맞도록 정의된 소프트웨어 개발 방법론의 절차, 사용기법 등을 수정 및 보완하는 작업

프로젝트 특징 정의 -> 표준 프로세스 선정 및 검증 -> 상위 수준의 커스터마이징 -> 세부 커스터마이징 -> 테일러링 문서화

고려사항 : 내부적 기준 – 목표 환경 / 요구사항 / 프로젝트 규모 / 보유 기술

외부적 기준 – 법적 제약사항 / 표준 품질 기준

028 소프트웨어 개발 프레임워크

- 소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러가지 기능들을 제공해주는 반제품 형태의 소프트웨어 시스템

주요 기능 – 예외 처리 / 트랜잭션 처리 / 메모리 공유 / 데이터 소스 관리 / 쿼리 서비스 / 로깅 서비스 / 사용자 인증 서비스

종류 – 스프링 프레임워크 / 전자정부 프레임워크 / 닷넷 프레임워크

스프링 프레임워크 – 자바 플랫폼을 위한 오픈 소스 경량형 애플리케이션 프레임워크

전자정부 프레임워크 – 대한민국의 공공부문 정보화 사업 시 효율적인 정보 시스템의 구축을 지원하기 위해 필요한 기능 및 아키텍처를 제공하는 프레임워크

닷넷 프레임워크 – windows 프로그램의 개발 및 실행 환경을 제공하는 프레임워크

특성 – 모듈화 / 재사용성 / 확장성 / 제어의 역흐름