요구사항 확인

chap1. 소프트웨어 개발방법론

- 소프트웨어 생명주기 모델(SDLC)
    - 시스템의 요구분석부터 유지보수까지 전 공정을 체계화한 절차
    - 소프트웨어 생명주기 모델 프로세스 (**요설구테유**)
        - 요구사항 분석
        - 설계
        - 구현
        - 테스트
        - 유지보수
    - 소프트웨어 생명주기 모델 종류 (**폭프나반**)
        - 폭포수 모델
            - 개발 시 각 단계를 확실히 마무리 지은 후 다음 단계로 넘어감.
            - 가장 오래된 모델.
            - 선형 순차적 모형, 고전적 생명주기 모형
            - 요구사항 변경 어려움
        - 프로토타이핑 모델
            - 고객이 요구한 주요 기능을 프로토타입으로 구현, 고객의 피드백을 반영하여 소프트웨어를 만들어 나감.
        - 나선형 모델
            - 위험을 최소화하기 위해 점진적으로 완벽한 시스템으로 개발해 가는 모델
        - 반복적 모델
            - 구축대상을 나누어 병렬적으로 개발 후 통합, 반복적으로 개발 해 점증 완성시킴.

구분	폭포수 모델	프로토타이핑 모델	나선형 모델	반복적 모델
특징	순차적 접근	프로토타입 개발	위험분석,반복 개발	증분방식으로 병행 개발
장점	이해 용이, 관리 편리	요구분석 용이, 타당성 검증	위험성 감소,변경에 유연 대처	병행 개발로 인한 일정 단축 가능
단점	요구사항 변경이 어려움	프로토타입 폐기에 따른 비용 증가	단계 반복에 따른 관리 어려움	병행 개발에 따른 관리 비용 증가

- 소프트웨어 개발 방법론
    - 소프트웨어 개발방법론 종류 (**구정객컴애제**)
        - 구조적 방법론
            - 전체 시스템을 기능에 따라 나누어 개발, 이를 통합하는 분할과 정복 접근 방식의 방법론
            - 프로세스 중심 하향식 방법론
        - 정보공학 방법론

        - 객체지향 방법론
        - 컴포넌트 기반 방법론
        - 애자일 방법론
            - 절차보다 사람이 중심이 되어 변화에 유연, 신속 적응, 효율적으로 시스템을 개발할 수 있는 개발 방법론.
        - 제품 계열 방법론
    - 애자일 (Agile)
        - 절차보다 사람이 중심. 변화에 유연, 신속 적응. 효율적으로 시스템 개발
        - 개발 시간이 짧고, 폭포수와 대비됨. 개발과 함께 즉시 피드백을 받아 유동적 개발 가능
        - 등장 배경
            - 소프트웨어 개발 환경의 변화
            - 기존 개발방법론의 한계
        - 애자일 방법론의 유형 (**X린스**)
            - XP
                - eXtreme programming, 의사소통 개선과 즉각적 피드백으로 품질을 높이기 위한 방법론
                    - XP 5가치
                        - 용기, 단순성, 의사소통, 피드백, 존중
                    - XP 12가지 기본 원리
                        - 짝 프로그래밍, 공동 코드 소유, 지속적 통합, 계획 세우기, 작은 릴리즈, 메타포어, 간단한 디자인, 테스트 기반 개발, 리팩토링, 40시간 작업, 고객 상주, 코드 표준
            - 린
                - 도요타 낭비 요소 제거, 품질 향상
                - Just in Time, 칸반 보드 사용
            - 스크럼
                - 매일 정해진 시간, 짧은 시간의 개발. 프로젝트 관리 중심 방법론
                - 주요개념
                    - 백로그
                    - 스프린트
                    - 스크럼 미팅
                    - 스크럼 마스터
                    - 스프린트 회고
                    - 번 다운 차트
- 비용산정, 일정관리 모형
    - 비용산정 모형 개념 - 소프트웨어 규모파악을 위한 투입자원, 소요 시간 파악, 실행 가능한 계획 수립을 위해 비용을 산정하는 방식
    - 비용산정 모형 분류
        - **하향식 산정, 상향식 산정기법**

분류	설명	종류
허향식 산정방법	경험이 많은 전문가에게 비용 산정 의뢰	전문가 판단 델파이 기법

분류	설명	종류
상향식 산정 방법	세부적 요구사항과 기능에 따른 비용 계산	코드 라인 수 (Loc) Man Month COCOMO 푸트남 기능점수(FP) 모형

- 비용산정 모형 종류
  - 델파이 기법
    - 전문가의 경험적 지식을 통한 문제 해결 및 미래예층을 위한 기법.
  - Loc (Lines of Code) 모형
    - 소프트웨어 각 기능의 원시 코드 라인 수를 통해 예측치를 구하고 비용을 산정.
    - 측정이 쉽고 이해가 쉬움.
    - 예측치 계산 식 구글링해서 외우기.
  - Man Month 모형
    - 한사람이 1개월 동안 할 수 있는 양을 기준으로 프로젝트 비용 산정.
  - COCOMO 모형
    - **보헴**이 제안한 모형으로 규모에 따라 비용을 산정하는 방식
    - 비용산정 결과는 프로젝트를 완성하는데 필요한 노력으로 산정
    - 규모에 따라 **조직형, 반 분리형, 임베디드형** 으로 구분
  - 푸트남 모형
    - 소프트웨어 개발주기의 단계별로 요구할 인력의 분포를 가정하는 방식
  - 기능점수 FP 모형
    - 요구 기능을 증가시키는 인자별로 가중치를 부여하고, 요인별 가중치를 합산하여 총 기능의 점수를 계산하여 비용을 산정하는 방식
- 일정관리 모델
  - 프로젝트가 일정 기한 내에 적절히 완료될 수 있도록 관리하는 모델
    - 주 공정법 (CPM)
    - PERT
    - 중요 연쇄 프로젝트 관리(CCPM)

---

chap2. 현행 시스템 분석

- 현행 시스템 파악
  - 현행 시스템이 어떤 하위 시스템으로 구성되어 있고, 제공 기능 및 연계 정보는 무엇이며 어떤 기술 요소를 사용하는 지 파악
  - 현행 시스템 파악 절차
    - 1. 구성/기능/인터페이스 파악
    - 2. 아키텍처 및 소프트웨어 구성 파악
    - 3. 하드웨어 및 네트워크 구성 파악
  - 현행 시스템 구성/기능 및 인터페이스 파악
    - 현행 시스템 구성 현황 파악 : 조직의 주요 업무를 처리하는 기간 업무와 이를 지원하는 지원 업무로 구분하여 파악
    - 기능 현황 파악 : 단위 업무 시스템이 현재 제공하고 있는 기능 파악

        - 인터페이스 현황 파악 : 단위 업무 시스템이 다른 시스템과 주고 받는 데이터의 종류, 데이터 형식, 프로토콜, 연계 유형, 주기 파악.
- 소프트웨어 아키텍처
    - 소프트웨어 아키텍처 개념 - 여러 가지 소프트웨어 구성요소와 그 구성요소가 가진 특성 중 외부에 드러나는 특성, 구성요소 간의 관계를 표현하는 시스템의 구조나 구조체
    - 소프트웨어 아키텍처 프레임워크
        - 소프트웨어 집약적 시스템에서 아키텍처가 표현해야 하는 내용 및 이들 간의 관계를 제공하는 아키텍처 기술 표준.
    - 소프트웨어 아키텍처 프레임워크 구성요소
        - 아키텍처 명세서
        - 이해관계자
        - 관심사
        - 관점
        - 뷰
        - 근거
        - 목표
        - 환경
        - 시스템
    - **소프트웨어 아키텍처 4+1 뷰**
        - 고객의 요구사항을 정리해 놓은 시나리오를 4개의 관점에서 바라보는 소프트웨어적인 접근 방법.
        - 4개의 분리된 구조로 구성되는 아키텍처 개념을 제시하고 이의 체크 방법으로 유스케이스 사용
        - 소프트웨어 아키텍처 4+1 뷰 구성요소 (**유 논구프배**)
            - **1은 유스케이스 뷰**
            - **4는 논리 뷰, 구현 뷰, 프로세스 뷰, 배포 뷰**

뷰	설명
유스케이스 뷰	유스케이스 또는 아키텍처 도출, 설계. 다른 뷰를 검증하는데 사용 사용자, 설계자, 개발자, 테스트 관점
논리 뷰	시스템의 기능적 요구사항이 어떻게 제공되는지 설명해주는 뷰 설계자, 개발자 관점
프로세스 뷰	시스템의 비기능적인 속성으로서 자원의 효율적인 사용, 병행 실행, 비동기, 이벤트 처리 등을 표현한 뷰 개발자, 시스템 통합자 관점
구현 뷰	개발 환경 안에서 정적인 소프트웨어 모듈의 구성을 보여주는 뷰 컴포넌트 구조와 의존성을 보여주고 컴포넌트에 관한 부가적인 정보 정의
배포 뷰	컴포넌트가 물리적인 아키텍처에 어떻게 배치되는가를 매핑해서 보여주는 뷰

- 소프트웨어 아키텍처 패턴
    - 소프트웨어를 설계할 때 참조할 수 있는 전형적인 해결 방식
    - 소프트웨어 아키텍처 패턴 유형 (**계클파브모**)
        - 계층화 패턴
            - 시스템을 계층으로 구분하여 구성한다.
            - 각 하위 모듈들은 특정한 수준의 추상화를 제공, 각 계층은 다음 상위 계층에 서비스 제공
        - 클라이언트 - 서버 패턴

            - 하나의 서버와 다수의 클라이언트로 구성됨.
            - 클라이언트를 통해 서버에 서비스를 요청하면 서버는 클라이언트에 서비스를 제공
            - 서버는 계속 클라이언트로부터 요청을 대기
        - 파이프-필터 패턴
            - 데이터 스트림을 생성하고 처리하는 시스템에서 사용 가능
            - 서브 시스템이 입력 데이터를 받아 처리하고, 결과를 다음 서브 시스템으로 넘겨주는 과정 반복
            - 필터 컴포넌트는 재사용성이 좋고, 추가가 쉬워 확장 용이
        - 브로커 패턴
            - 분리된 컴포넌트들로 이루어진 분산시스템에서 이용. 원격 서비스 실행을 통해 상호작용 가능 패턴.
            - 컴포넌트 간의 통신을 조정하는 역할 수행
        - 모델 - 뷰 - 컨트롤러 패턴
            - MVC 패턴, 대화형 애플리케이션을 모델, 뷰, 컨트롤러 3개의 서브 시스템으로 구조화하는 패턴.
            - 모델 - 핵심 기능과 데이터 보관
            - 뷰 - 사용자에게 정보 표시
            - 컨트롤러 - 사용자로부터 요청을 입력받아 처리
            - 각 부분이 별도의 컴포넌트로 분리되어 있어 서로 영향을 받지 않고 개발 작업 가능
            - 효율적 재사용 가능, 여러 개의 뷰가 있어야 하는 대화형 애플리케이션 구축에 적합.

- 소프트웨어 아키텍처 비용 평가 모델

    - 아키텍처 접근법이 품질 속성에 미치는 영향 판단, 아키텍처의 적합성을 평가하는 모델.
    - (**SACAA**)
    - SAAM - 변경 용이성과 기능성에 집중, 평가가 용이하여 경험이 없는 조직에서도 활용 가능한 비용 평가 모델
    - ARAM - 아키텍처 품질 속성을 만족시키는지 판단 및 품질 속성들의 이해 상충관계까지 평가하는 모델
    - CBAM - ATAM 바탕의 시스템 아키텍처 분석 중심으로 경제적 의사결정에 대한 요구를 충족하는 비용 평가 모델
    - ADR - 소프트웨어 아키텍처 구송요소 간 응집도를 평가하는 모델
    - ARID - 전체 아키텍처가 아닌 특정 부분에 대한 품질요소에 집중하는 비용 평가 모델

- 디자인 패턴

    - 소프트웨어 공학의 소프트웨어 설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계 방법을 정리한 패턴
    - 디자인 패턴 구성요소 (**패문솔 사결샘**)
        - 패턴의 이름
        - 문제 및 배경
        - 솔루션
        - 사례
        - 결과
        - 샘플 코드
    - 디자인 패턴 유형 (**목범**)
        - 목적 (**생구행**)
            - 생성
            - 구조
            - 행위
        - 범위
            - 클래스
            - 객체

    - 디자인 패턴 종류
        - 생성 패턴 (**생빌 프로 팩앱싱**)
            - Builder
            - Prototype
            - Factory Method
            - Abstract Factory
            - Singleton
        - 구조 패턴 (**구 브데 퍼플 프록 컴 어**)
            - Bridge
            - Decorator
            - Facade
            - Flyweight
            - Proxy
            - Composite
            - Adapter
            - Mediator
            - Interpreter
        - 행위 패턴 (**행 미인이 템옵 스테 비커 스트 메체**)
            - Iterator
            - Template Method
            - Observer
            - State
            - Visitor
            - Command
            - Strategy
            - Memento
            - Chain of Resposibility

- 현행 시스템 분석서 작성 및 검토

    - 현행 시스템 관련 자료 수집
    - 수집 자료의 분석
    - 분석한 결과 기반 산출물 작성
        - 분석 산출물의 종류
            - **현기인 아소하네**
                - 정보시스템 구성 **현황**
                - 정보 시스템 **기능** 구성도
                - **인터페이스** 현황
                - 현행 시스템 **아키텍처** 구성도
                - **소프트웨어** 구성도
                - **하드웨어** 구성도
                - **네트워크** 구성도
                - 이 현기인 아소하네 구성도들은 사례(그림)을 보고 뭔지 맞춰야함.

- 개발 기술 환경 정의

    - 운영체제 현행 시스템 분석
        - 품질 측면

            - 신뢰도
            - 성능
        - 지원 측면
            - 기술 지원
            - 주변 기기
            - 구축 비용
    - 운영체제 종류 및 특징
        - PC
            - 윈도즈
            - 유닉스
            - 리눅스
        - 모바일
            - 안드로이드
            - iOS
    - 네트워크 현행 시스템 분석
        - OSI 7계층
        - 백본망, 라우터, 스위치, 게이트웨이, 방화벽 대상으로 분석
    - DBMS 현행시스템 분석 (**중접인관샤무백**)
        - 중복 제어
        - 접근 통제
        - 인터페이스 제공
        - 관계 표현
        - 샤딩/파티셔닝
        - 무결성 제약조건
        - 백업 및 회복
    - DBMS 현행 시스템 분석 (**가성호기구**)
        - 성능 측면
            - 가용성 - 장기간 시스템 운영 시 장애 발생 가능성
            - 성능 - 대용량 데이터 처리 성능
            - 상호 호환성 - 설치 가능한 운영체제 종류
        - 지원 측면
            - 기술 지원 - 공급 업체들의 안정적인 기술 지원 여부
            - 구축 비용 - 라이선스 정책 및 비용
    - 미들웨어 현행 시스템 분석
        - 미들웨어
            - 미들웨어는 분산 컴퓨팅 환경에서 응용 프로그램과 프로그램이 운영되는 환경 간 원만한 통신이 이뤄질 수 있도록 제어해주는 소프트웨어
            - 운영체제와 소프트웨어 애플리케이션 사이에 위치함.
            - WAS (웹 애플리케이션 서버)
                - 서버 계층에서 애플리케이션이 동작할 수 있는 환경을 제공하고 안정적인 트랜잭션 처리와 관리, 이기종 시스템과의 애플리케이션 연동 지원
        - 미들웨어 현행 시스템 분석 고려사항
            - 성능 측면
                - 가용성
                - 성능
            - 지원 측면

        - 기술 지원
        - 구축 비용
    - 기술 환경 정의를 위한 자료 수집
        - 온라인 트랜잭션 처리(OLTP) 시스템
            - 시스템 구축 형태
            - 사용자 수
            - 트랜잭션 수
        - 웹/앱 애플리케이션 서버 (WEB/WAS)
            - 시스템 용도 및 서비스 형태
            - 시스템 구성 형태
            - 접속자 수

---

chap3. 요구사항 확인

- 요구공학 개념
    - 요구 공학은 사용자의 요구가 반영된 시스템을 개발하기 위해 사용자 요구사항에 대한 도출, 분석, 명세, 확인 및 검증하는 구조화된 활동

구분	기능적 요구사항	비기능적 요구사항
개념	시스템이 제공하는 기능, 서비스에 대한 요구 사항	시스템이 수행하는 기능 이외 사항, 시스템 구축에 대한 제약사항에 관한 요구 사항
도출 방법	특정 입력에 대해 시스템이 어떻게 반응해야 할 지 기술 특정 상황에 대해 시스템이 어떻게 동작해야 하는지 기술	품질 속성에 관련해 시스템이 갖춰야할 사항에 관한 기술 시스템이 준수해야 할 제한 조건에 관한 기술
특성	기능성, 완전성, 일관성	신뢰성, 사용성, 효율성, 유지보수성, 이식성, 보안성 및 품질 관련 요구사항, 제약 사항

사례

- 요구사항 개발 단계 구성 (CMM Level 3 프로세스 영역) (**도분명확**)
    - 도출 분석 명세 확인 및 검증
    - 도출 - 요구사항 소스. 도출 기법 (**인브델롤워설**)
        - 인터뷰
        - 브레인스토밍
        - 델파이 기법
        - 롤 플레잉
        - 워크숍
        - 설문 조사
    - 분석 - 요구사항 분류, 개념 모델링, 기술 구조 설계 , 요구사항 할당, 요구사항 협상
        - 분석 단계 절차
            - 요구사항 분류
            - 개념 모델링 생성 및 분석
            - 요구사항 할당
            - 요구사항 협상

            - 정형 분석
    - 명세 - 시스템 정의서, 시스템 요구사항 명세서, 소프트웨어 요구사항 명세서
        - 비정형 명세 기법
        - 정형 명세 기법
        - 요구사항 명세 및 검증 항목 (**명완검 일수 추개**)
            - 명확성
            - 완전성
            - 검증 가능성
            - 일관성
            - 수정 용이성
            - 추적 가능성
            - 개발 후 이용성
    - 확인 - 검토, 프로토타이핑, 모델 검증, 인수 테스트
        - 요구사항 확인 및 검증 단계의 주요 기법
            - 요구사항 검토
            - 정형 기술 검토 활용 (**동워인**)
                - 동료 검토
                - 워크 스루
                - 인스펙션
            - 프로토타이핑 활용
            - 모델 검증
            - 테스트 케이스 및 테스트를 통한 확인
            - CASE 도구 활용 검증
            - 베이스라인을 통한 검증
            - 요구사항 추적표
        - 상세 정형 기술 검토 기법 (**관기 인워감**)
            - 관리 리뷰
            - 기술 리뷰
            - 인스펙션
            - 워크 스루
            - 감사

- 요구사항 관리 단계 (CMM Level 2 프로세스 영역)

    - 요구사항 관리 단계 절차
        - 요구사항 협상
        - 요구사항 기준선 설정
        - 요구사항 변경관리
        - 요구사항 확인 및 검증

- 요구사항의 시스템화 타당성 분석

    - 기술적 타당성 검토
        - 성능,용량 산정의 적정성
        - 시스템 간 상호 운용성
        - IT 시장 성숙도 및 트렌드 부합성
        - 기술적 위험 분석

chap4. 분석 모델 확인하기