2주차

소프트웨어 시스템 설계 및 개발

2025.1학기

CONTENTS

소프트웨어 공학

- 과학
  - 자연 세계를 이해하기 위해 관찰, 실험, 이론을 사용하는 지식 체계로 세상이 특정한 방식으로 작동하는 이유를 찾고 설명하려고 함
  - 이 과정에서 가설을 세우고 실험을 통해 자연 현상의 원리와 법칙을 발견
- 이학
  - 순수 과학이라고도 하며 실용적인 목적보다는 지식 자체의 확장을 목표로 함
  - 기본적인 과학적 원리와 법칙을 탐구하며 이러한 지식은 나중에 응용과학이나 공학의 기반으로 사용될 수 있음. 물리학, 화학, 생물학 등의 분야가 속함
- 공학
  - 과학적 원리를 실용적인 문제 해결에 적용하는 것으로 과학적 지식을 사용하여 기술을 설계, 개발, 구축함
  - 공학의 목적은 인간의 생활을 개선하고 효율적인 기계, 구조물, 시스템, 공정을 만드는 것
- 요약
  - 과학은 자연 세계의 이해를 추구, 이학은 이러한 지식을 순수한 탐구차원에서 확장
  - 공학은 이러한 과학적 원리를 인간의 필요를 충족시키기 위한 실용적 문제 해결에 적용

소프트웨어 공학

- 소프트웨어 공학이란
  - 컴퓨터 소프트웨어의 개발, 운영, 유지보수를 체계적, 규율적, 그리고 측정 가능한 방법으로 접근하는 기술과 공학의 한 분야
  - 이 분야는 고품질의 소프트웨어를 경제적으로 개발하는 방법론, 도구, 프로세스를 연구
  - 소프트웨어 공학의 목적
    - 생산성을 향상시키고,
    - 소프트웨어의 품질을 보장하며,
    - 프로젝트의 시간과 비용을 최적화

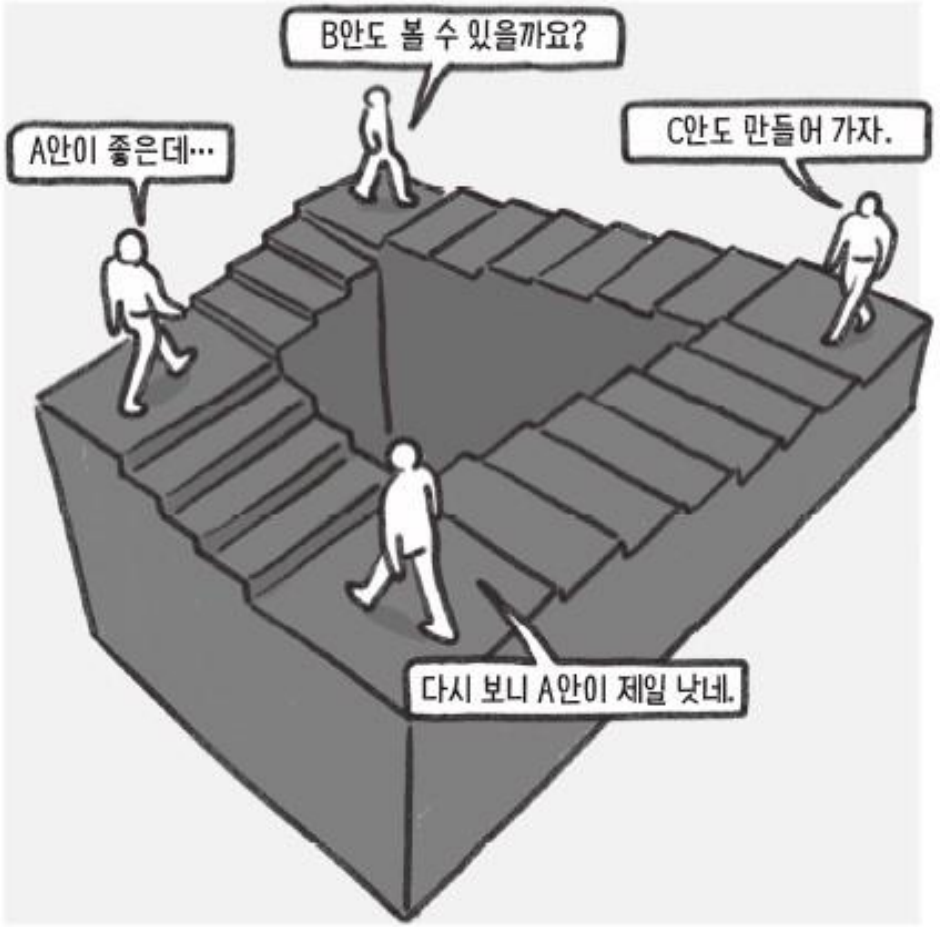
V.S.

소프트웨어 공학

- 소프트웨어 공학에서 생산성 향상, 품질 보증, 시간과 비용 절약을 위한 다양한 전략
  - 요구사항 관리
  - 설계 및 분석
  - 품질보증 계획 수립
  - 애자일 방법론
  - 지속적 통합 및 배포 CI/CD
  - 다양한 테스트(자동화, 성능, 보안)
  - 코드 재사용 및 모듈화
  - 버전 관리 시스템 사용
  - 프로젝트 관리 도구 사용
  - 효율적 팀 구성 및 협업 도구 사용
  - 코드리뷰, 교육 및 멘토링

요구사항 관리

- **목적** : 프로젝트의 성공적인 완수를 위한 사용자 및 시스템 요구사항의 정의, 문서화 및 관리
- **핵심 구성 요소**
  - 요구사항 수집: 이해관계자 인터뷰, 설문조사, 사용 사례 분석
  - 요구사항 분석 및 명세: 명확성, 일관성, 검증 가능성 확보
  - 요구사항 검증 및 확정: 이해관계자와의 검토 및 승인
  - 변경 관리: 변경 요청의 관리 및 추적
- **중요성** : 프로젝트 범위의 정의, 예산 및 일정 계획의 기준, 품질 목표 설정의 출발점

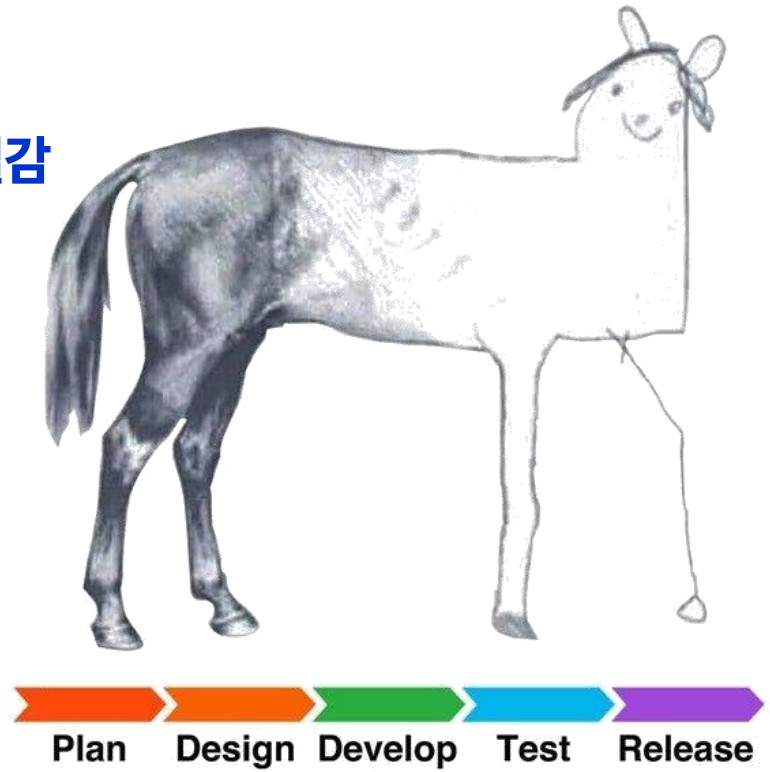


설계 및 분석

- **목적** : 시스템 설계가 요구사항을 만족하는지 확인하고 잠재적인 문제를 사전에 식별
- **핵심 구성 요소**
  - 설계 원칙과 패턴 적용: 재사용성, 유지보수성, 확장성 확보
  - 아키텍처 검토: 계층화, 컴포넌트 분리, 인터페이스 정의
  - 코드 검토 및 정적 분석: 버그, 성능 문제, 보안 취약점 식별
- **중요성** : 개발 초기 단계에서의 문제 해결은 비용 및 시간 절감, 프로젝트 성공률 향상

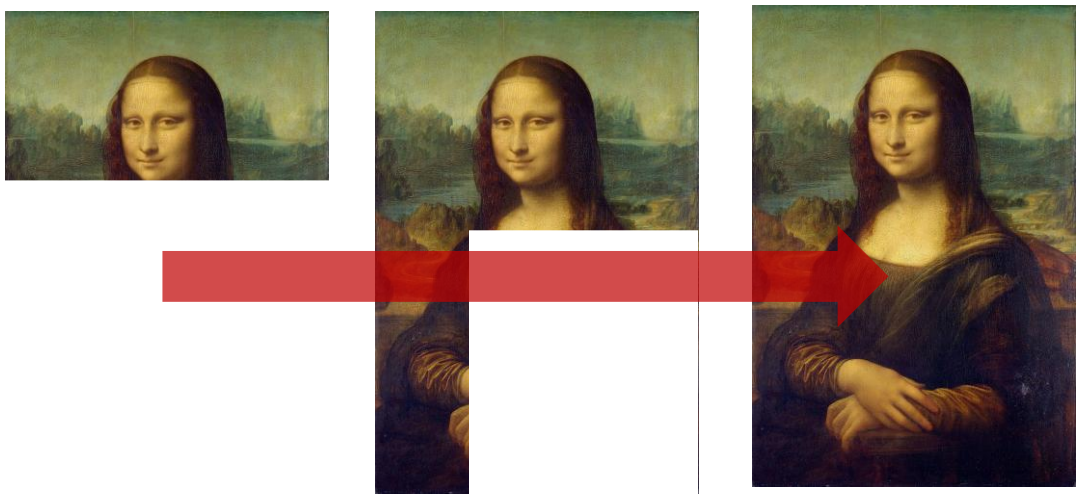
품질보증 계획

- **목적** : 제품 및 프로세스의 품질 요구사항을 명확히 하고, 이를 달성하기 위한 전략과 방법론 정의
- **핵심 구성 요소**
  - 품질 목표 설정: 사용자 만족도, 성능 기준, 오류율 감소 목표
  - 품질 관리 활동: 품질 검사, 오류 추적 및 수정, 리스크 관리
  - 품질 개선 방법론: Six Sigma, TQM, Lean Manufacturing
- **중요성** : 프로젝트의 성공적인 결과물 제공, 사용자 만족도 향상, 비용 절감



애자일 방법론

- **목적** : 빠르고 유연한 반응을 통해 고객의 변화하는 요구사항을 만족시키는 소프트웨어 개발 방법
- **핵심 구성 요소**
  - 스프린트/반복 작업: 짧은 개발 사이클을 통한 빠른 피드백과 개선
  - 사용자 스토리 및 백로그: 우선 순위에 따른 기능 정의와 관리
  - 일일 스탠드업 미팅: 팀원 간 진행 상황 공유 및 협업 강화
- **중요성** : 고객 만족도 향상, 프로젝트 위험 감소, 시간 대비 가치 극대화



CI/CD

- **목적** : 소프트웨어 개발 프로세스에서 지속적인 통합, 테스트, 배포를 통해 품질을 관리하고 배포 시간을 단축
- **핵심 구성 요소**
  - 자동화된 빌드 및 테스트: 코드 변경 사항의 지속적인 통합 및 검증
  - 배포 자동화: 스테이징, 프로덕션 환경으로의 빠르고 일관된 배포
  - 롤백 메커니즘: 문제 발생 시 이전 버전으로의 신속한 복구
- **중요성** : 개발 속도 및 효율성 향상, 배포 관련 위험 감소, 사용자 피드백 기반 빠른 개선

테스트

- **목적 : 소프트웨어의 신뢰성, 성능, 보안성을 검증하고 향상시키기 위한 다양한 테스트 실행**
- **핵심 구성 요소**
  - 자동화 테스트: 단위 & 통합 테스트, 시스템 테스트를 자동화하여 개발 과정에서 발생할 수 있는 오류를 신속하게 탐지 및 수정
  - 성능 테스트: 시스템의 응답 시간, 처리량, 리소스 사용량 등을 측정하여 성능 기준을 충족하는지 평가
  - 보안 테스트: 취약점 스캔, 침투 테스트를 통해 애플리케이션의 보안 취약점을 식별하고 개선
- **중요성 : 고품질의 소프트웨어 제공, 사용자 경험 향상, 보안 위협으로부터의 보호**

코드 재사용 및 모듈화

- **목적** : 코드의 재사용성과 모듈성을 통해 개발 효율성을 향상시키고 유지보수를 용이하게 함
- **핵심 구성 요소**
  - 코드 재사용: 라이브러리, 프레임워크, 패턴의 재사용을 통한 개발 시간 단축 및 오류 감소
  - 모듈화: 기능별로 코드를 모듈로 분리하여 의존성 관리, 코드 가독성 및 재사용성 향상
  - 인터페이스 정의: 모듈 간의 상호작용을 위한 명확한 인터페이스 정의를 통한 결합도 감소
- **중요성** : 코드 베이스의 확장성, 유연성 향상, 개발 및 유지보수 비용 절감

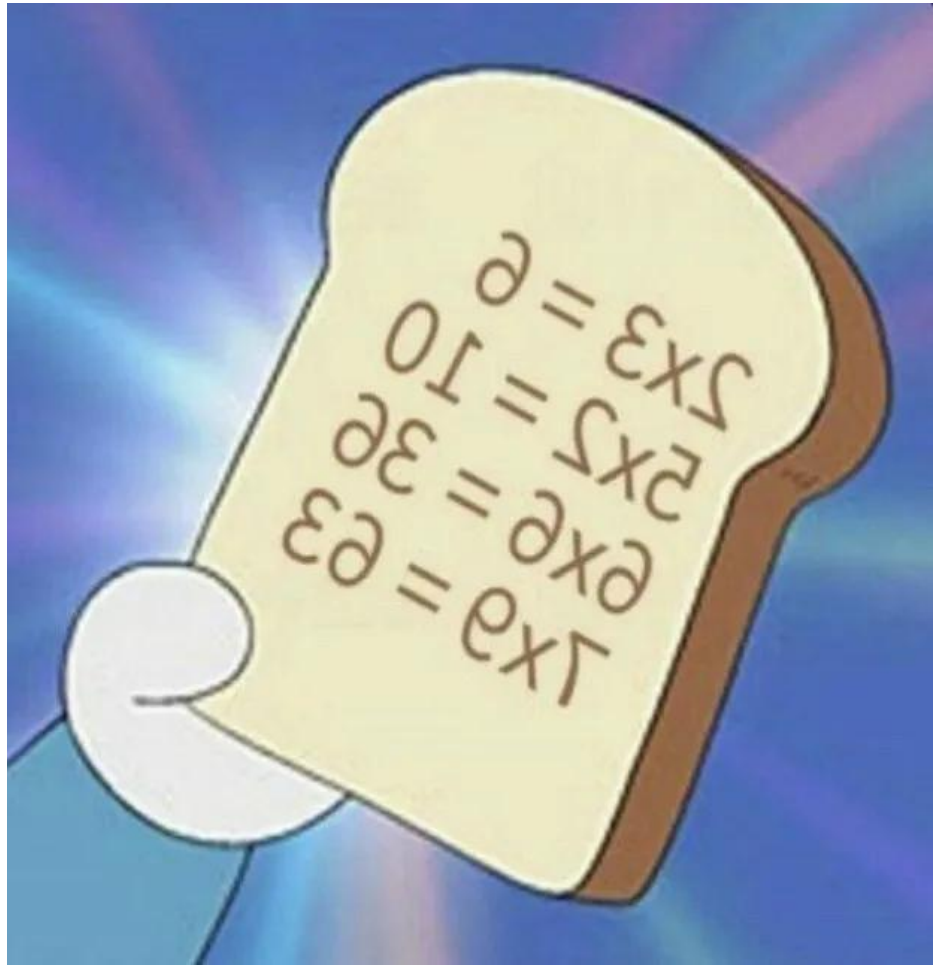
버전관리

- **목적** : 소프트웨어 개발 과정에서 발생하는 변경 사항을 효율적으로 관리하고 추적
- **핵심 구성 요소**
  - 소스 코드 버전 관리: Git, SVN 등의 도구를 사용한 코드 변경사항의 버전 관리 및 기록
  - 브랜치 전략: 기능 개발, 버그 수정, 릴리즈 관리를 위한 효율적인 브랜치 전략 수립
  - 변경 사항의 리뷰 및 통합: 코드 리뷰 과정을 통한 품질 관리, 팀원 간의 협업 및 지식 공유 촉진
- **중요성** : 소프트웨어의 안정성, 협업 효율성 향상, 프로젝트의 이력 관리 및 복구 용이성

문서집계표(최종).HWP
문서집계표(최종수정).HWP
문서집계표(최종수정컨펌).HWP
문서집계표(컨펌V1).HWP
문서집계표(컨펌V2).HWP
문서집계표(컨핌V3).HWP
문서집계표(진짜최종).HWP
문서집계표(진짜진짜최종).HWP
문서집계표(진짜진짜진짜최종).HWP
문서집계표(회장님).HWP
문서집계표(회장님지시수정).HWP
문서집계표(회장님수정.V1).HWP.....

프로젝트 관리 도구 사용

- **목적** : 프로젝트의 계획, 실행, 모니터링, 및 종료 과정을 체계적으로 관리하기 위한 도구의 사용
- **핵심 구성 요소**
  - 작업 분할 구조(WBS): 프로젝트를 관리 가능한 작은 단위로 분할
  - 일정 관리: Gantt 차트, 마일스톤 설정으로 일정 계획 및 추적
  - 리소스 관리: 팀원, 예산, 장비 등 프로젝트 리소스 할당 및 최적화
  - 커뮤니케이션 플랫폼: 팀 내외 커뮤니케이션 채널 통합
- **중요성** : 프로젝트의 성공적인 완료를 위한 효과적인 계획, 통제 및 커뮤니케이션 지원



효율적 팀구성 및 협업 도구 사용

- **목적** : 다양한 역할, 기술, 배경을 가진 팀원들의 효율적인 협업을 촉진하기 위한 조직 구성 및 도구 활용
- **핵심 구성 요소**
  - 역할 기반 팀 구성: 프로젝트 요구에 맞는 역할 및 책임 할당
  - 협업 도구: Slack, Microsoft Teams, Asana 등을 통한 실시간 커뮤니케이션 및 협업
  - 지식 공유: 위키, 문서 공유 시스템을 통한 정보의 중앙화 및 접근성 향상
  - 원격 근무 지원: 클라우드 기반 도구를 사용한 원활한 원격 작업 환경 제공
- **중요성** : 다양한 전문 지식의 결합, 팀 내외 커뮤니케이션의 개선, 유연한 작업 환경 제공

코드리뷰
교육 및 멘토링

- **목적** : 코드의 품질을 향상시키고, 팀원 간 지식 전달 및 기술 발전을 촉진하기 위한 과정
- **핵심 구성 요소**
  - 코드리뷰: 동료의 코드 검토를 통한 버그, 코드 스타일, 설계 문제 식별 및 수정
  - 페어 프로그래밍: 두 명의 개발자가 한 작업을 같이 수행하여 문제 해결 능력과 코드 품질 향상
  - 교육 세션 및 워크샵: 최신 기술, 도구, 방법론에 대한 교육으로 지식 확장
  - 멘토링 프로그램: 경험 많은 개발자가 신입 개발자를 지도, 경력 개발 및 성장 지원
- **중요성** : 지속적인 학습 문화 조성, 코드 품질 및 팀 효율성 향상, 직원 만족도 및 유지 증가

우리 주변에서 프로젝트 찾기

- 조원 구성(랜덤)
- 아이스 브레이킹
- 주제 정하기
- 요구사항 정의
- 간단한 UX/UI 만들어보기
- 고객과의 미팅
- 변경 사항 협의
- 조별 발표자료 만들기

[주제 예시]

- [학교] 도서관 자리 예약 시스템
- [학교] 과제 질답 게시판
- [운동] 볼링 점수 계산기
- [운동] 모바일 홈 PT
- [가정] 레시피 재료 계산기
- [가정] 자취 물품 관리 장부
- [취미] 달리기 기록 비교
- [여성] 내 화장품 제품 리뷰관리