1주 2강

소프트웨어공학의 이해

이번 주차에는…

소프트웨어 공학의 이해

- 공학
- 소프트웨어 개발 과정
- 소프트웨어 공학

1. 공학

- 공학의 사용 예
  - 전기공학과, 건축공학과, 토목공학과 등의 대학교에서 학과 명으로 사용
- 공학의 특성
  - 제약 사항: 정해진 기간, 주어진 비용
    - 과학적 지식을 활용하여 문제를 해결하는데 한정된 기간과 비용의 제약을 받음
- 소프트웨어 공학
  - 소프트웨어 + 공학
  - 취지: '소프트웨어 개발 과정에 공학적인 원리를 적용하여 소프트웨어를 개발'
  - 목적:
    - S/W 개발의 어려움 해결
    - 효율적 개발을 통한 생산성 향상
    - 고품질 소프트웨어 제품

2. 소프트웨어의 개발 과정

- 소프트웨어 개발 생명주기(SDLC Software Development Life Cycle)
  - 계획 단계에서 유지보수 단계에 이르기까지 일어나는 일련의 과정

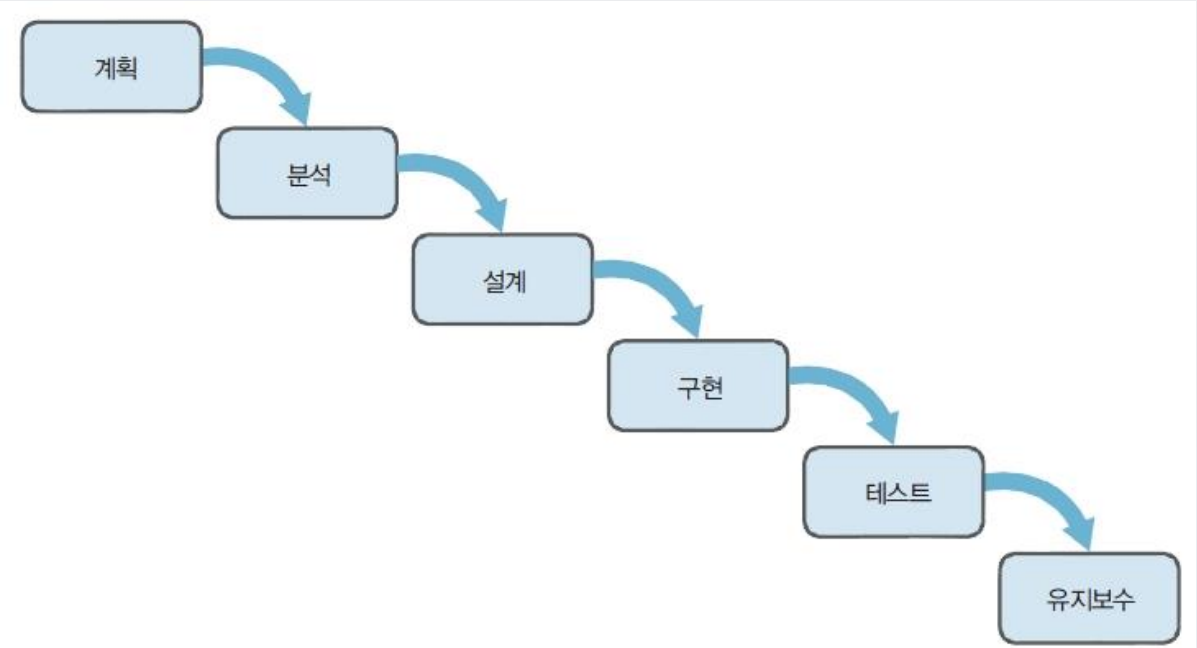


그림 1-8 소프트웨어 개발 생명주기 SDLC: Software Development Life Cycle

3. 소프트웨어 공학

- 정의

> *품질 좋은 소프트웨어를 경제적으로 개발하기 위해*
> *계획을 세우고, 개발하며,*
> *유지 및 관리하는 전 과정에서*
> *공학, 과학 및 수학적 원리와 방법을 적용하여*
> *필요한 이론과 기술 및 도구들에 관해*
> *연구하는 학문*

- 목표
  - 개발 과정에서의 생산성 향상
  - 고품질의 소프트웨어 생산 → 사용자 만족

4. 소프트웨어란

- 소프트웨어
  - 프로그램과 프로그램의개발, 운용, 보수에 필요한 관련 정보 일체
- 엔지니어링 작업의 결과
  - 프로그램 이외의 정보도 중요
  - S/W가 복잡해지면서 설계가 중요
- 다른 엔지니어링의 결과물

5. 소프트웨어의 특징

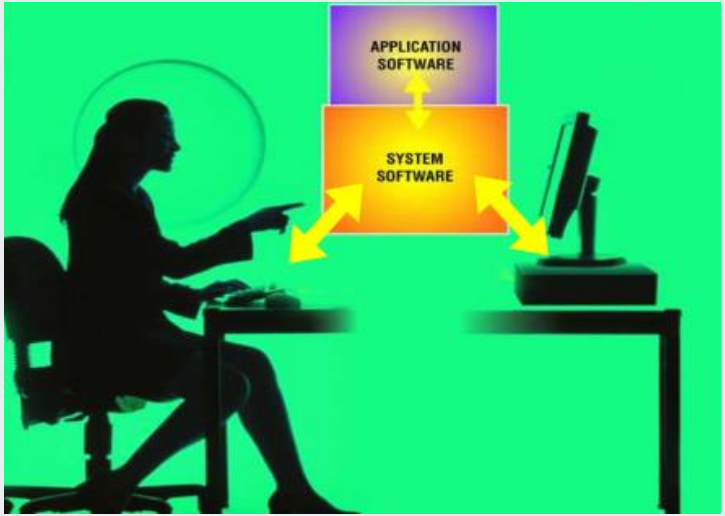
- 손에 잡히지 않음(intangible)
  - 개발 작업에 대한 이해가 힘듦
  - 구조가 쉽게 파악되지 않음
- 대량 생산하기 쉬움(reproducible)
  - 비용의 대부분은 개발
    - 다른 엔지니어링은 생산(제조) 단계의 비용이 큼
- 노동 집약적(labor-intensive)
  - 자동화가 어려움

6. 소프트웨어의 특징

- 잘 훈련 받지 않으면 제작이 어려움
  - 품질 문제를 인식하기 어려움
  - 객체지향, 컴포넌트, 분산 시스템 등의 다양한 기술 필요
- 쉽게 변경 가능
  - 완전한 이해 없이 변경 할 수 있음
- 닳아 없어지는 것이 아님
  - 설계 변경으로
    - 오류가 생기거나
    - 예상하지 못한 방향으로 복잡해질 수 있음

7. 소프트웨어의 특징

- 결론적으로
  - 설계가 취약하면 구조가 점점 악화
  - 소프트웨어의 수요는 높아 감
  - 지속적인 '소프트웨어 위기'
  - 소프트웨어 엔지니어링 기법을 배워야 함



8. 소프트웨어의 종류

- 주문형
  - 특정 고객의 수요를 만족하기 위하여 개발된 소프트웨어
  - 예) 웹사이트, 항공-교통제어 시스템, 재정관리 시스템
- 패키지형
  - 공개된 시장에서 판매
  - COTS(Commercial Off-The-Shelf)
- 임베디드 시스템
  - 하드웨어에 탑재
  - 변경이 어려움

9. S/W 종류별 차이

- 주문형, 범용, 임베디드 S/W의 차이점

종류	주문형	패키지형	임베디드
사용되는 카피의 수	낮음	중간	높음
소프트웨어 수행에 필요한 하드웨어 성능	낮음	높음	중간
개발 노력	높음	중간	낮음

10. 소프트웨어의 종류

- 리얼타임 소프트웨어
  - 제어, 모니터링 소프트웨어
  - 신속히 반응
  - 안전성 확보가 중요
- 자료 처리 소프트웨어
  - 비즈니스 업무 처리에 사용
  - 자료의 정확성과 보안이 관건
  - 일괄처리
- 두 가지 성격을 동시에 지니는 S/W도 있음

11. 소프트웨어 공학이란?

- 고객의 문제를 해결해 주기 위하여 대규모의 품질 좋은 소프트웨어 시스템을 정해진 시간과 비용으로 개발하거나 발전시키는 체계적인 프로세스

- 고객의 문제를 해결
  - 소프트웨어 공학의 궁극적인 목표
  - 솔루션을 구매할 수 있 도 있음
  - 불필요한 기능을 추가하는 것은 문제 해결에 도움이 안됨
  - 문제를 파악하고 이해하기 위하여 효과적으로 커뮤니케이션 하여야 함

12. 소프트웨어 공학이란?

- 체계적인 개발과 발전
  - 잘 이해하고 있는 기술을 조직된 원리와 방법으로 적용하는 과정
  - 표준으로 받아들여진 기술 - ISO, IEEE, KS
  - 대부분은 발전적(evolution) 개발
- 대규모 고품질 S/W 시스템
  - 홀로 완성 시킬 수 없기 때문에 엔지니어링 기술 필요
  - 팀워크와 협동이 필요
  - 작업의 분할과 시스템의 각 부분이 잘 작동 하느냐가 관건
  - 최종 제품은 충분히 좋은 품질을 가지고 있어야 함

13. 소프트웨어 공학이란?

- 비용, 시간, 기타 제약
  - 한정된 자원
  - 얻는 이득이 비용을 초과하여야
  - 더 빠르고 싸게 개발하도록 다른 기업과 경쟁
  - 비용과 시간의 부정확한 예측으로 대부분의 프로젝트가 실패로 끝남

14. 소프트웨어 엔지니어링

- ‘소프트웨어 엔지니어링’ 이란 말은 1968년 NATO 컨퍼런스에서 처음 언급
  - 소프트웨어 개발에 엔지니어링 원리가 도입되어야 함을 처음으로 인식

- 엔지니어링은 자격증 제도가 필수
  - 공공을 보호할 목적
  - 엔지니어의 설계 작업은 수학, 과학, 경제 등의 원리를 적용할 수 있도록 충분한 연습이 필요
  - 도덕적 윤리 의식도 필요

15. 소프트웨어 공학의 정의

- IEEE
  - 소프트웨어의 개발, 운용, 유지보수 및 파기에 대한 체계적인 접근 방법
- W. Humphrey
  - 질 좋은 소프트웨어를 경제적으로 생산하기 위하여 공학, 과학, 수학적 원리에 의하여 소프트웨어를 개발하는 것
- 목표
  - 품질 좋은 소프트웨어를 최소의 비용으로 계획된 일정에 맞추어 개발하는 것

16. 시스템 공학

- 다른 요소에 탑재될 목적으로 만드는 S/W는 시스템 공학의 일부
- 시스템 공학의 절차

17. 소프트웨어 공학 관련자

- 사용자
  - 소프트웨어를 사용하는 사람들
- 고객
  - 소프트웨어에 대하여 비용을 지불하는 사람
- 소프트웨어 개발자
  - 요구 분석가, 데이터베이스 전문가, 기술 문서 작성자, 프로그래머, 테스트 엔지니어, QA 엔지니어
- 개발 관리자
  - 프로젝트 관리, 비즈니스 경영

18. 소프트웨어 프로젝트의 유형

- 소프트웨어 프로젝트의 유형
  - 새로운 시스템 개발
    - 창조력 발휘, 설계에 자유로움
    - 작업에 많은 시간이 소요됨
    - 소프트웨어 구조가 중요함
  - 진화 유형
    - 기존 시스템에 대한 깊은 이해가 필요함
    - 새로 추가하는 기능과 조화되어야 함
  - 컴포넌트 기반
    - 프레임워크나 컴포넌트를 기초로 시작함
    - 조립과 커스터마이징이 주된 작업

19. 소프트웨어 프로젝트 작업

- 요구 분석과 명세화
  - 도메인 분석
  - 문제의 정의
  - 요구 추출
    - 가능하면 많은 소스에서 입력을 취함
  - 요구 분석
    - 정보를 잘 정리하고 구성
  - 요구 명세화
    - 소프트웨어가 어떻게 작동하는지에 대한 자세한 명령을 작성

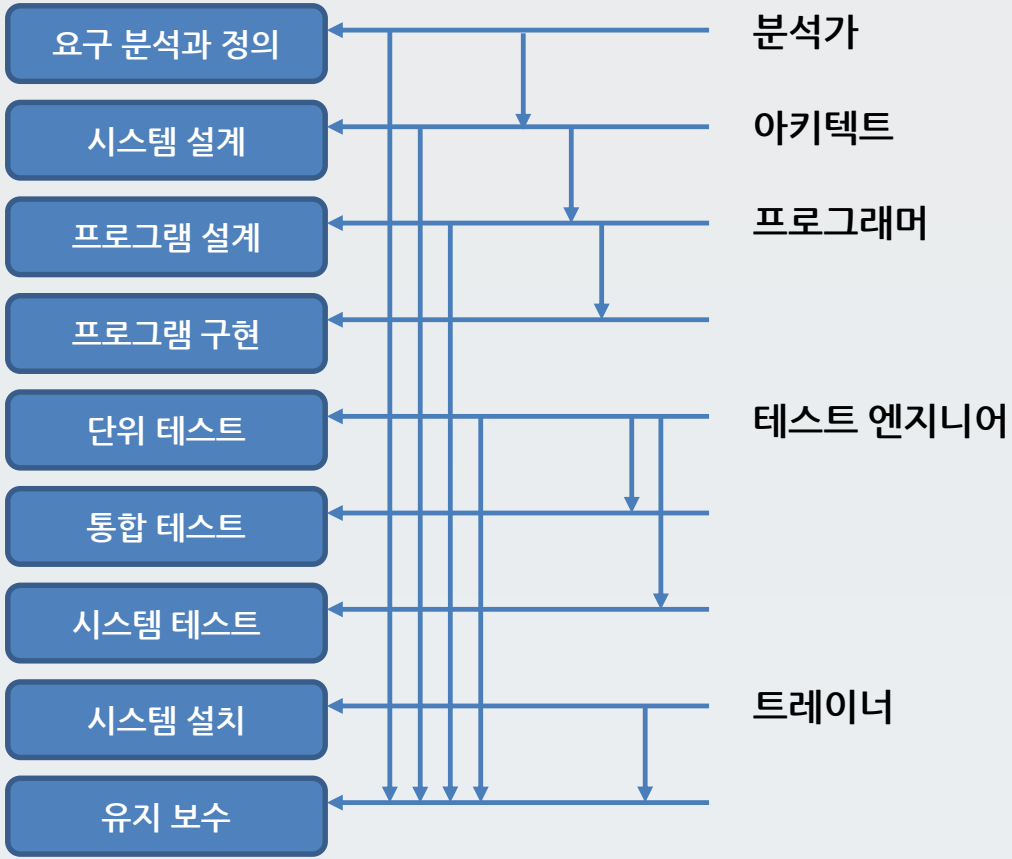
20. 소프트웨어 프로젝트 작업

- 설계
  - 요구가 가용 기술로 어떻게 구현되어야 하는지를 기술
  - 중요 기술
    - 시스템 엔지니어링 : 어떤 부분이 하드웨어, 소프트웨어가 되어야 하는지 결정
    - 소프트웨어 구조 : 시스템을 서브 시스템으로 분할하고 서브 시스템이 어떻게 작동하는지 결정
    - 서브 시스템의 내부에 대한 성세 설계
    - 사용자 인터페이스 설계
    - 데이터베이스 설계

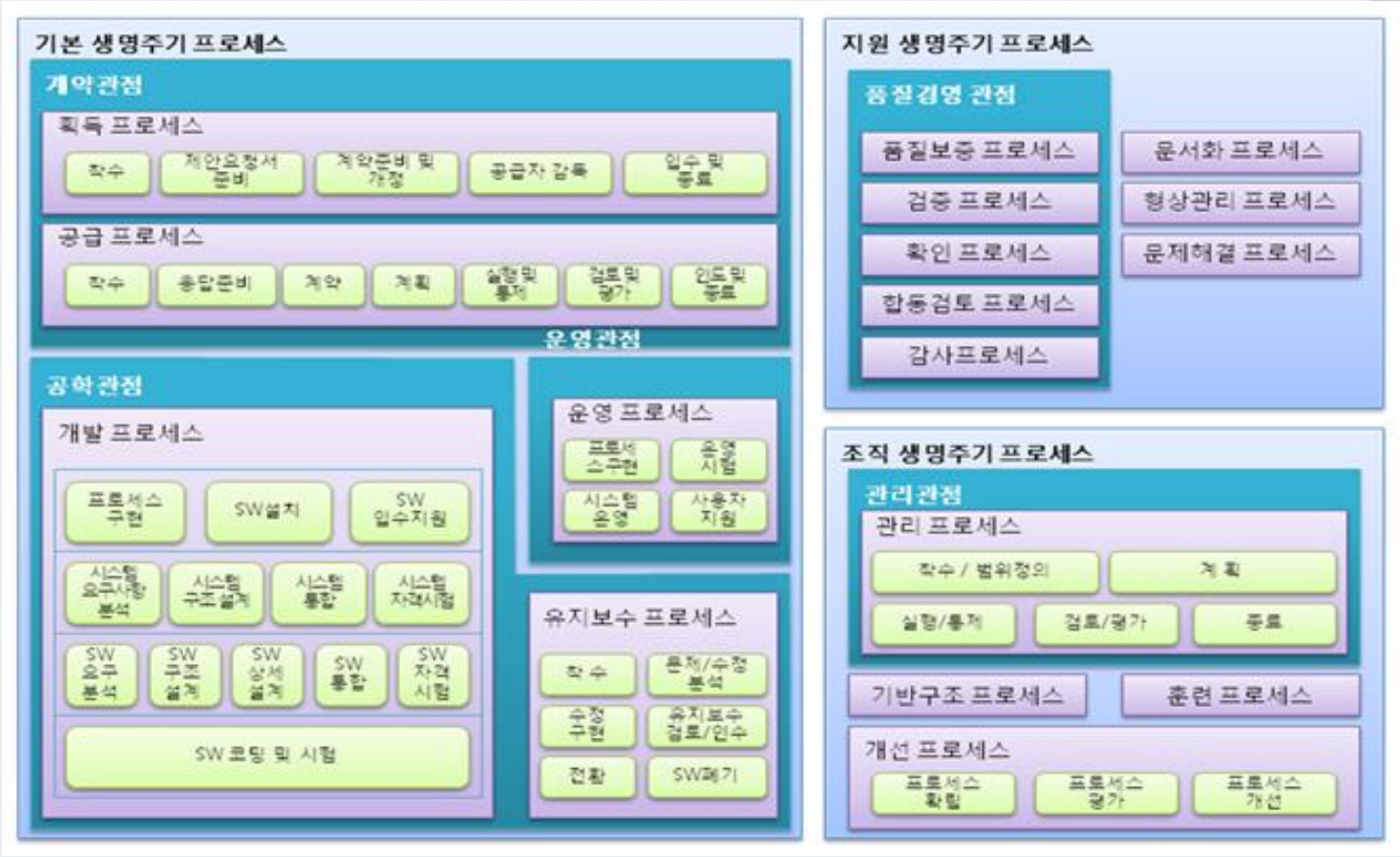
21. 소프트웨어 프로젝트 작업

- 모델링
  - 도메인이나 소프트웨어의 표현을 만들어 나가는 과정
    - 사용 사례 모델링
    - 정적 모델링
    - 동적 모델링
  - 프로그래밍
  - 품질 보증
    - 리뷰, 인스펙션
    - 테스트
  - 설치
  - 프로세스 관리

22. 작업과 담당자


요구 분석과 정의
시스템 설계
프로그램 설계
프로그램 구현
단위 테스트
통합 테스트
시스템 테스트
시스템 설치
유지 보수
분석가
아키텍트
프로그래머
테스트 엔지니어
트레이너

23. ISO 12207 - 프로세스 표준



24. 객체지향 소프트웨어 공학

- 객체 사이의 상호 작용을 중시
- 객체지향 모델링과 설계 언어
  - 유즈케이스 모델링
  - UML(Unified Modeling Language)
- 객체지향 개발 프로세스
  - 반복, 점증적
  - 잦은 변경과 빠른 기술 발전
- 객체지향 방법론
  - 각 단계 작업 방법

25. 소프트웨어 공학의 핵심 기술

- 추상(abstraction)
- 분석과 설계 방법
- 사용자 인터페이스 프로토타이핑
- 소프트웨어 구조
- 소프트웨어 프로세스
- 재사용
- 측정(measurement)
- 도구와 통합 환경

다음 시간

개발 단계의 소개