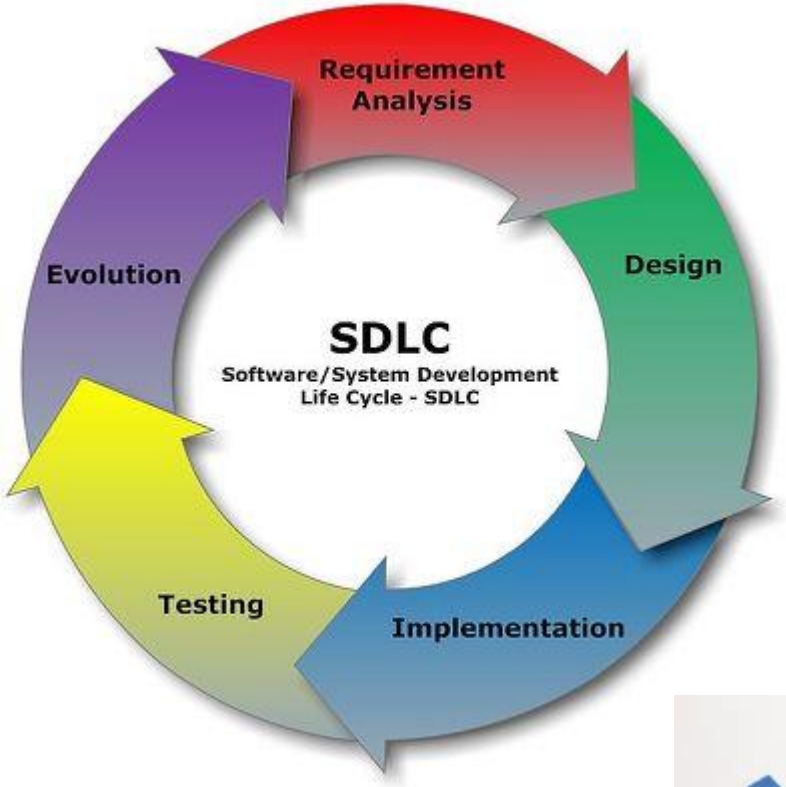
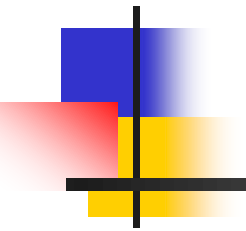

Requirement Analysis
Design
Implementation
Testing
Evolution
SDLC
Software/System Development Life Cycle - SDLC


change



제 2장. 소프트웨어공학

March. 2018

Young-gon, Kim
ykkim@kpu.ac.kr
Department of Computer Engineering
*K*orea *P*olytechnic *U*niversity

Topics covered

- 소프트웨어공학의 정의
- 소프트웨어 프로세스
- 소프트웨어공학 실무
- 소프트웨어 개발 실무

1. 소프트웨어공학

◆ 소프트웨어 개발을 위한 현실

- 소프트웨어의 해결책이 개발되기 전에 **문제 이해** 노력
  - 특정 소프트웨어 제공 기능에 이해당사자의 관심 증대
- **설계**가 중요한 액티비티
  - 정보 기술에 대한 요구 증대
- 소프트웨어 **높은 품질**
  - 전략 및 전술적인 의사결정을 소프트웨어에 의존
- 소프트웨어의 **유지보수**
  - 특정 애플리케이션 가치 증대로 사용자 / 소프트웨어 수명 확장

◆ 현실에 대한 결론

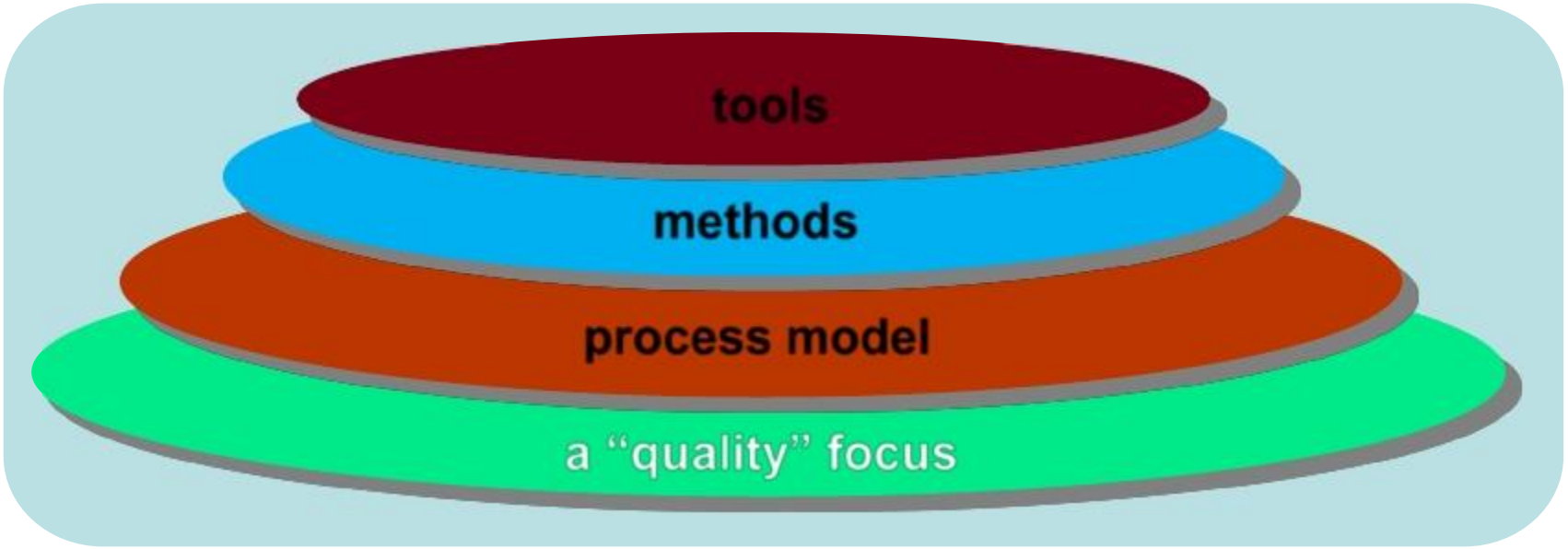
- 모든 형태, 모든 애플리케이션 도메인에 걸친 **소프트웨어는 공학**적으로 관리
- 소프트웨어공학 필요 : **경제성 / 효율성.**

1. 소프트웨어 공학 정의

◆ 소프트웨어 공학이란

- IEEE
  - 소프트웨어에 대한 **개발, 운용, 유지보수**에 대한 **체계**적이고 **엄격**하고 **정량**적인 접근 방법
  - **소프트웨어 공학을 적용하는 것**과 같은 접근 방법의 연구

◆ 체계적인 기술



1. 소프트웨어 공학 계층 기술

- **품질** 에 집중
  - 계속적인 **프로세스 개선** 문화 촉진
  - 소프트웨어공학에 **효율적인 개발을 지속적으로 유도**
  - 소프트웨어공학을 **지지**하는 것의 기본원리
- 프로세스 계층
  - 컴퓨터 소프트웨어를 **합리적으로 빨리 개발**
  - 소프트웨어공학 기술을 **효율적으로 전달해야 할 프레임워크 정의**
  - 소프트웨어 **프로젝트 관리** 기반 형성, **기술적인 방법** 적용, **작업 산출물** 생성 , **이정표** 수립 , **품질보장 , 변경이 관리**되는 상황 확립
- 소프트웨어 방법
  - 소프트웨어를 **개발하기위한 실용 기술** 제공
  - 방법 : **커뮤니케이션 , 요구분석 , 설계모델링 , 프로그램 작성 , 테스팅**
- 소프트웨어 도구
  - 프로세스와 방법을 **자동적으로 반자동적**으로 지원
  - 소프트웨어 개발 지원시스템 – **컴퓨터 자원 소프트웨어공학.**

2. Software Process

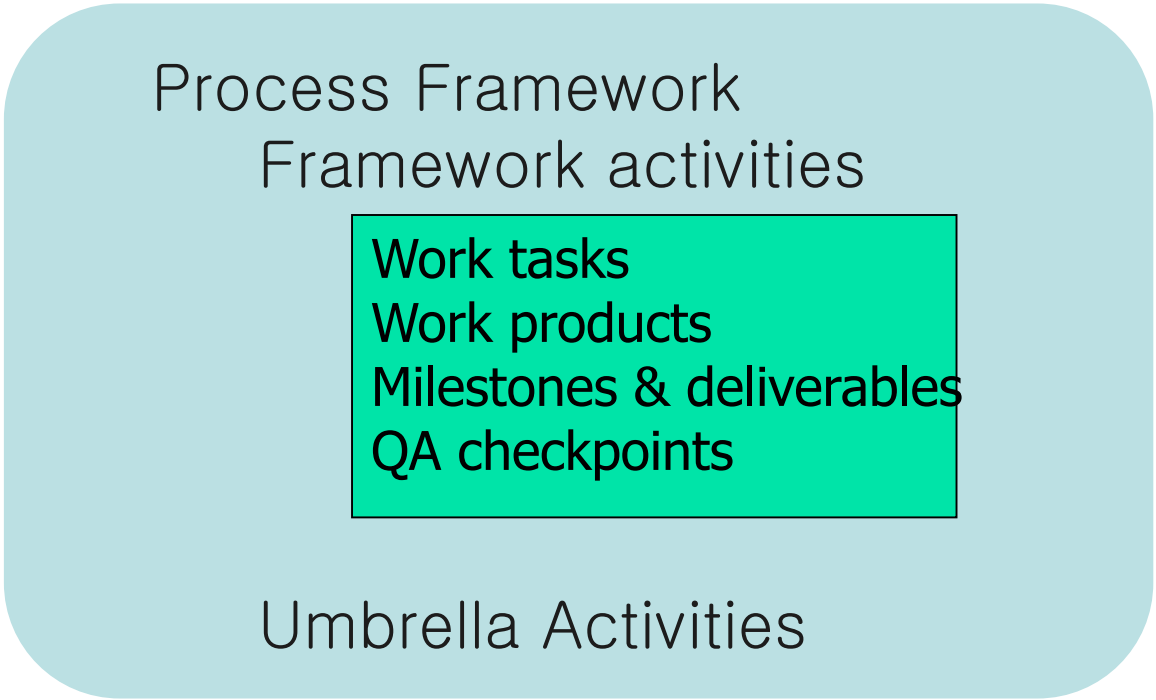
◆ 프로세스

- **작업 산출물**을 생성하기 위해 행해지는
  - Sets {Activity, Action, Task}
- 액티비티
  - 넓은 의미의 **목표 달성** (이해관계자와의 커뮤니케이션)
  - 애플리케이션 도메인, 프로젝트의 크기, 노력의 복잡도, 소프트웨어가 제공되는 엄격도에 **무관하게 적용**
- 액션
  - 주된 **작업 산출물** (구조설계 모델) 을 생성하는 태스크로 구성
- 태스크
  - **작지만 잘 정의** 된 명백한 결과를 생성하는 **목표** (단위 테스트 수행).

2.1 Process framework

◆ 프로세스 프레임워크

- 모든 프로젝트에 적용 가능한 프레임워크 액티비티를 찾아내어 **소프트웨어공학 프로세스가 완전** 하게 되도록 하는 기반 확립.



2.1 Process Activities

- **커뮤니케이션**
  - 요구사항 수집 목적 :
    - 이해관계자 **목표 이해**, 소프트웨어 **특징 / 기능 정의**
- **계획 수립**
  - 소프트웨어공학 **작업 정의**
  - 수행 **태스크**, 발생할 **위험**, 요구되는 **자원**, **작업산출물, 작업스케줄**
- **모델링**
  - 요구사항 분석 : 문제 **이해와 해결** 노력
  - 설계 : 모델 생성
- **구축**
  - **코드 생성**
  - **테스팅**
- **배치**
  - 소프트웨어 (전체 도는 증분) 가 **고객에게 전달**
  - 고객 : **프로젝트 평가 , 평가에 대한 피드백.**

2.2 Umbrella Activities

- **소프트웨어 프로젝트 추적과 제어**
  - **계획 대비 진행사항 평가 , 스케줄 유지**하기 위한 액션
- **리스크 관리**
  - 프로젝트 **결과**와 프로덕트의 **품질에 영향을 주는 위험요소** 평가
- **소프트웨어 품질 보증**
  - 소프트웨어 **품질 보증**하기 위한 필요한 **액티비티 정의및 수행**
- **기술 검토**
  - **에러**가 다음 액티비티로 **전파 전 발견및 제거**위해 **작업산출물 평가**
- **측정**
  - 소프트웨어 **납품 지원 프로세스 , 프로젝트 / 프로덕트 측정 척도** 정의
- **소프트웨어 형상 관리**
  - 프로세스 전반에 걸쳐 **변경이 미치는 영향**을 관리
- **재사용 관리**
  - 컴포넌트를 포함한 프로덕트 **재사용 기준 정의 , 컴포넌트 생성 기법**
- **작업 산출물 준비와 제작**
  - 모델 , 로그 , 서식 , 리스트등 **작업 산출물 생성** 액티비티.

2.3 프로세스 적응

- 하나/다른 프로젝트에 채택된 프로세스의 차이점
  - 액티비티, 액션, 태스크의 **전반적인 흐름과 그것들 사이의 상호성**
  - 각 프레임워크 액티비티에서 **액션과 태스크가 정의** 되는 정도
  - **작업 산출물**이 확인되고 요구되는 정도
  - **품질보증 액티비티**가 적용되는 방법
  - **프로젝트 추적과 제어 액티비티**가 적용되는 방법
  - 프로세스를 기술할 때 **전체적인 상세함과 정확**한 정도
  - **고객 및 다른 이해관계자**가 프로젝트에 연관되는 정도
  - 소프트웨어 **팀에게 주어지는 자율성**의 수준
  - **팀의 조직과 역할**이 규정된 정도.

3. 소프트웨어 공학 실무

- 소프트웨어공학 실무 : **소프트웨어 프로세스 모델**
  - 소프트웨어공학 작업을 위한 **골격구조**
    - **일반적인 프레임워크** : 커뮤니케이션, 계획, 수립, 모델링, 구축, 배치
    - **보호활동** : 관리 , 제어 , 측정 , 검토

- 소프트웨어공학 실무의 핵심
  - **문제의 이해** : 커뮤니케이션과 분석
  - **해결 방안의 계획** : 모델링과 소프트웨어 설계
  - **계획을 실행** : 코드 작성
  - **정확성을 위해 결과 조사** : 테스트와 품질 보증.

3.1 문제의 이해(코뮤니케이션과 분석)

- 필요한 **질문의 답에 노력** 필요
  - 문제의 **해결 방안에 누가 이해관계**를 가지나 ?
    - 누가 이해관계자인가 ?
  - **모르는 것**이 무엇인가 ?
    - 어떤 데이터 , 기능 , 특징들이 문제 해결을 위해 요구되나 ?
  - **문제가 나누어질** 수 있나 ?
    - 이해하기 쉬운 좀 더 쉬운 작은 문제로 나누는 것이 가능한가 ?
  - **문제를 그래픽**으로 나타낼 수 있나 ?
    - 분석 모델이 만들어 질 수 있나 ?

3.2 해결 방안의 계획(모델링과 설계)

- **코딩 전에 설계를 조금**하는데 노력 필요
  - **전에 유사한 문제를 본 적**이 있나 ?
    - 가능한 해결방법 중에 인지할 수 있는 패턴이 있나 ?
    - 요구되는 데이터, 기능, 특징을 구현해 놓은 소프트웨어가 있나 ?
  - **유사한 문제가 해결**되었나 ?
    - 그렇다면 해결방안의 요소들을 재사용할 수 있나 ?
  - **부문 문제가 정의**될 수 있는가 ?
    - 그렇다면 해결방안은 부문문제에도 명백하게 적용될 수 있나 ?
  - **효율적으로 구현될 수 있도록 해결방안**을 잘 표현할 수 있나 ?
    - 설계모델이 만들어 질 수 있는가 ?

3.3 계획을 실행(코드 작성)

- 계획은 **길을 잃지 않고 일을 진행할 수 있게 도움** 제공
  - **해결방안이 계획**과 잘 맞는가 ?
    - 소스코드로 설계모델을 추적할 수 있나 ?
  - 해결방안의 각 부분들이 **입증할 수 있을 정도**로 올바른가 ?
    - 설계와 코드를 검토했나 ?
    - 알고리즘에 정확성 검증을 하였나 ?

3.4 정확성을 위해 결과 조사(테스트와 품질보증)

- 가능한 **모든 에러를 발견하기 위해 충분한 테스트를 설계**한 것은 확신 가능
  - 해결방안의 각 **컴포넌트 부분을 테스트하는 것이 가능**한가 ?
    - 적당한 테스트 전략이 수행 되었나 ?
  - 해결방안이 요구되는 **데이터 , 기능 , 특징들에 잘 맞는 결과**를 만들어 내는가 ?
    - 소프트웨어가 이해관계자의 요구사항과 비교해서 검증이 되었나 ?

3.5 일반적인 원칙

- 원칙 : 생각체계에서 필요로 하는 주요한 근원적인 법칙 / 가정
- 전반적인 **소프트웨어공학 실무에 초점**을 맞춘 7 가지 원칙
  - **1) 이유는 항상 존재한다**
    - 모든 개발 결정 전 : 시스템소프트웨어가 사용자에 가치를 높여 줄 수 있나 ?
  - **2) 간단하게 하라**
    - 모든 설계 간단 : 시스템을 좀더 쉽게 이해 및 유지보수 제공 -> 다수 반복 작업
  - **3) 비전을 유지 하라**
    - 비전을 가지고 있는 자율적인 설계가 : 성공적인 프로젝트 보장
  - **4) 당신이 만들어 내는 것을 다른 사람이 소비할 것이다**
    - 항상 당신이 하는 일은 다른 사람이 이해하도록 한다는 생각 : 명세화 , 설계 , 구현
  - **5) 미래에 대해 열려 있어라**
    - 일반 문제를 해결하는 시스템을 개발함으로써 모든 가능한 해결책 준비 : 재사용
  - **6) 재사용을 위해 미리 계획을 세워라**
    - 재사용을 계획 : 비용 절감 , 재사용 가능한 컴포넌트와 통합된 시스템 가치증대
  - **7) 생각하라**
    - 액션을 하기 전 : 안전하게 생각하는 것은 항상 좋은 결과 생성
    - 미리 생각 : 제대로 할 가능성 증대 , 다시 바로 할 수 있는지에 지식을 얻음.

4. 소프트웨어 개발 미신

- 소프트웨어와 소프트웨어 개발에 사용되는 **프로세스에 대한 잘못된 믿음**
  - **관리** 미신
    - 매니저 : 예산, 개발기간 스케줄, 품질 개선에 압력을 받음
    - 소프트웨어 개발 표준과 절차로 가득 찬 책
    - 개발기간이 지연되면 인력을 더 투입하면 해결
    - 소프트웨어를 외주를 주면 쉽게 개발 완료
  - **고객들**의 미신
    - 목표에 대한 일반 지식만으로 프로그램 작성 시작 충분 : 상세한 것 나중에 추가
    - 소프트웨어 요구사항이 계속 변하지만 : 소프트웨어 유연성으로 변화 쉽게 수용
  - **실무자들**의 미신
    - 일단 프로그램 작성하면 우리 업무 종료
    - 프로그램이 "작동" 될 때까지는 품질 평가 방법 없음
    - 소프트웨어공학은 많은 불필요한 문서를 생성하여 어쩔 수 없이 일정 지연

- 많은 소프트웨어 전문가 : 미신의 오류 인식
  - 현실을 인식하는 것 : 소프트웨어공학의 실질적인 해결방안 첫 단계.

Homework

◆ Chapter2. 소프트웨어공학

2.1 소프트웨어공학의 계층 기술

2.2 소프트웨어 개발 프로세스

2.3 소프트웨어공학 프로세스 프레임워크

2.4 소프트웨어공학 실무

2.5 소프트웨어공학 실무 원칙

Project

1장. 팀 프로젝트 개요

1.1 팀 프로젝트 제목

1.2 선정 이유

1.3 팀 운영 방법

2장. 팀 프로젝트 개요

2.1 시스템의 간략한 설명

2.2 유사 사례 시스템 소개