**03월 12일 수업 소프트웨어 공학 개론**

소프트웨어 공학이란?  
소프트웨어 개발의 여러 문제들을 공학적인 방법으로 해결하는 것.

공학(Engineering)은 과학적인 지식을 바탕으로 실용적인 문제를 해결하는 것.  
컴퓨터공학, 전자공학, 생명공학, 수면공학, 교육공학 등

소프트웨어 공학은 개발 프로세스, 방법론, 아키텍처, 도구, 기법등을 모든 것을 포함 한다.

소프트웨어 개발의 전 과정을 통해 소프트웨어의 품질을 향상시키고, 비용을 절감하며, 개발 기간을 단축시키는 것을 목표로 한다.

소프트웨어 품질 : 생산성, 비용, 효율성, 안정성, 성능, 확장성, 유지보수성 등이 존재한다.

소프트웨어의 품질을 향상시키고, 비용을 절감하며, 개발 기간을 단축시키는 것을 목표로 해서 과학적인 지식을 바탕으로 실용적인 문제를 해결하는 것이 소프트웨어 공학이다.

시각적으로 보기 좋고, 사용하기 편리한 것 : UI/UX

SWEBOK의 약자 기억 필요  
SWEBOK(Software Engineering Body of Knowledge)

유지보수, Engineering, 수학 공학에 대한 기초지식 등이 필요하다.

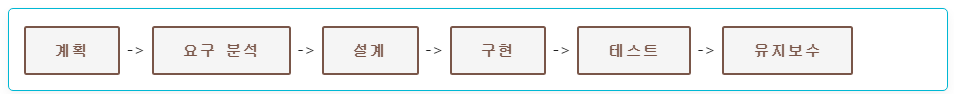
프로세스(Process)는 소프트웨어 개발을 위한 일련의 활동 내지는 과정을 의미한다.

* 정형화된 방법이 필요함.

방법론(Methodology)은 프로세스를 수행하기 위한 일련의 절차를 의미한다.

* 구체적인 방법을 의미한다.

SDLC(Software Development Life Cycle)



1. 계획 및 단계 : 무엇을 할 것인가?
2. 요구 분석 : 구체적으로 무엇을 할 것인가?

계획 및 단계와 요구 분석에서는 (Target, Goal), Domain 즉 목표가 필요하다.  
하고자하는 대상 영역을 Domain이라고 한다.  
그렇기에 계획 및 단계와 요구 분석에서 필요한 능력은 Domain의 개발 지식이 필요하다.  
은행업무 시스템, 대학 학사 시스템 등 이러한 시스템에 대한 지식이 필요하다.

Cf) SI : System Integration 즉 색깔이 없고 아무거나 한다.  
 특화된 영역을 가진 회사는 Domain이 있고, 미래에 도움이 된다.

* 문과쪽 특성이 필요하다.

테스트 -> 배포(배치) Deployment -> 유지보수

프로세스 모델중  
가장 유명한 모델이 폭포수 모델(Waterfall Model) (계단 모양)

계획, 분석, 설계, 구현, 테스트의 각 순서가 물 또는 시간이 흘러가듯이 흘러간다는 컨셉이다.  
특징 : 각 단계가 순차적으로 집행된다. -> 반대로 움직이지는 않는다.   
 🡺 기간과 비용을 선정하는데 유리함. (정부, 은행, 국방)

그렇기에 각 단계마다 검수라는 단계가 필요하다. => 갑과 을의 관계가 존재한다. 주로 돈.  
계획 -> 분석(중간중간 검수) -> 설계(중간중간 검수) -> 구현(중간중간 검수) -> 테스트(중간중간 검수) -> 최종 검수  
 => 검수 산출물이 존재한다.

요구사항의 변화가 적은 프로젝트에 적합하다.

V 모델 -> 폭포수 모델을 변형한 것으로 폭포수 모델의 약점을 부분적 보완함.  
분석 -> 분석 테스트 => 구현 -> 구현 테스트 => …..

프로토 타입(시작품, 시제품)  
POC -> 생각한 제품이 맞는지 확인하기 위해 최종 결과물을 만들어본다.

시스템의 일부분을 빠르게 구현, 사용자의 초기 요구사항을 파악해 프로토타입을 만듧, 요구 사항과 구현의 간극을 줄일 수 있도록 설계된 모델

단점 : 비용이 들어감.  
완전한 개발이 아니기 때문에 실제 개발물에서의 차이가 있을 수 있다.

프로토 타입을 개선한 모델을 나선형 모델이라고 한다.

계획 -> 분석 -> 설계 -> 구현 -> 프로토타입에 대한 사이클을 나선형으로 설계함  
계획은 원의 크기가 작기 때문에 규모가 작다. 🡺 빠르게 프로토타입을 구현해본다.  
프로토 타입을 만들고 잘 맞으면 다시 계획 -> 분석 -> 설계 -> 구현 -> 프로토타입 단계를 수행하고, 두번째 프로토 타입을 만들어서 확인한다. 이때는 나선형 구조로 인해서 원이 더 커지기에 규모 또한 더 커진다. 세번째와 네번째는 위 첫번째와 두번째 순서를 반복해서 진행한다.  
즉 사용자 평가가 포함된 반복적인 개발을 통해 요구사항의 변화에 유연하게 대처할 수 있다.

단점은 반복적 진행으로 인해 프로젝트 기간이 길어질 수 있고, 위험 관리를 위한 전문가 필요 등 추가 비용이 발생할 수 있다.

최종적인 모델을 UP(Unified Process)이다.  
현재 가장 많이 사용하는 모델은 애자일 모델인데, 애자일 모델의 초기버전이 UP이다.  
Unified Process는 UML(Unified Modeling Language)을 기반으로 하는 객체지향 소프트웨어 개발 프로세스이다.

UML : 시각적으로 표현도구 -> Rational 회사 => IBM에 인수됨  
RUP(Rational Unified Process) 저작권 및 라이선스  
공개 모델인 Open UP(Eclipse)와 Agile UP등이 있다.

텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

X축은 시간이고 Y축은 SDLC이다.

이 때 UP는 Phase라는 단계로 전환시켰다.  
Inception -> Elaboration -> Construction -> Transition  
도입기 정련기 구축기 전이기

도입기는 초반부  
정련기는 설계  
구축은 구현  
전이기는 개발 쪽에서 운영 쪽으로 전환되는 과정이다.

Eterative 반복적 increative 점차 증가하다.

I1에서 모든 SDLC를 전부 실행한다. 개발, 테스트, 배포까지 실행하되 비중이 다르다.  
즉 계획과 분석에 비중이 많고 실제로 개발과 배포는 해본다 => 배포때 문제점을 도출하기 위함.  
E1, E2에서도 반복하는데, 계획과 분석 비중이 디자인과 개발쪽으로 비중이 올라간다.  
마찬가지로 테스트와 배포를 실시한다.  
C1, C2, C3, C4에서도 모든 SDLC를 반복하는데, 계획 분석 디자인 개발 비중보다 개발과 테스트 비중이 올라간다.  
마지막 T1, T2에서는 계획에서 개발까지 비중이 거의 없는 반면, 테스트와 배포의 비중이 올라가면서 프로젝트가 끝나게 된다.