과제 Day4
개요
1. 소프트웨어 공학
2. 소프트웨어 생명주기
    1. 폭포수모델
    2. 프로토타이핑
    3. 나선형모델
3. 개발방법론
    1. 애자일
    2. TDD
    3. UML
    4. PDD
4. 형상관리, 버전관리
5. 느낀점

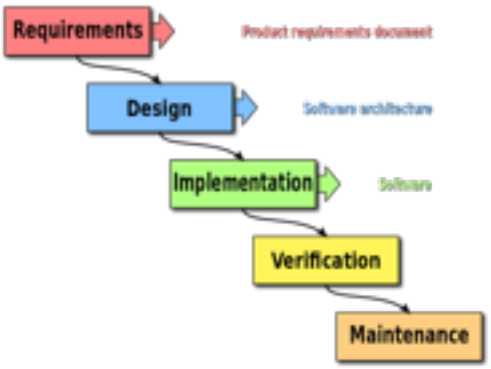
1. 소프트웨어 공학

이전 현업에서 일할때 개발 과정중에서 상당히 불필요한 작업, 혹은 비효율적이라고 생각하는 작업이 있었다.
개발팀 팀장은 사내에서 천재라 평가를 받는 사람이었지만 클라이언트의 말도안되는 요청, 혹은 개발이 완료된 기능에 대한 수정요청으로 인해 애초 30일로 주어졌던 기간이 부족했거나 추가근무를 해야하는 상황이 굉장히 빈번했다.

소프트웨어 공학은 새로운 어플리케이션의 개발, 운영, 유지보수 그리고 폐기까지의 과정을 체계화 하기위한 방법들을 다룬다. 그 중 어떻게 개발을 해야 좀 더 효율적으로 개발하고 무식한 고객의 니즈를 훌륭하게 반영하여 시장에 적합한 소프트웨어를 만들 수 있는지 그 방법론이 소프트웨어 생명주기이다.

2. 소프트웨어 생명주기
개발주기 모델에 대해 다음 3가지가 있다. 수업중 배운 생명주기 모델에는 4가지가 있다
주먹구구식, 폭포수모델, 프로토타이핑, 나선형모델
주먹구구식은 이름에서 모든게 설명되므로 제외하고 나머지 3가지에 대한 설명은 아래와 같다

1. 폭포수모델
“단계에서 단계로”
요구, 디자인(시스템적인 디자인으로 아키텍쳐를 말함), 구현, 검증, 유지보수를 계획하여 한번에 실행한다

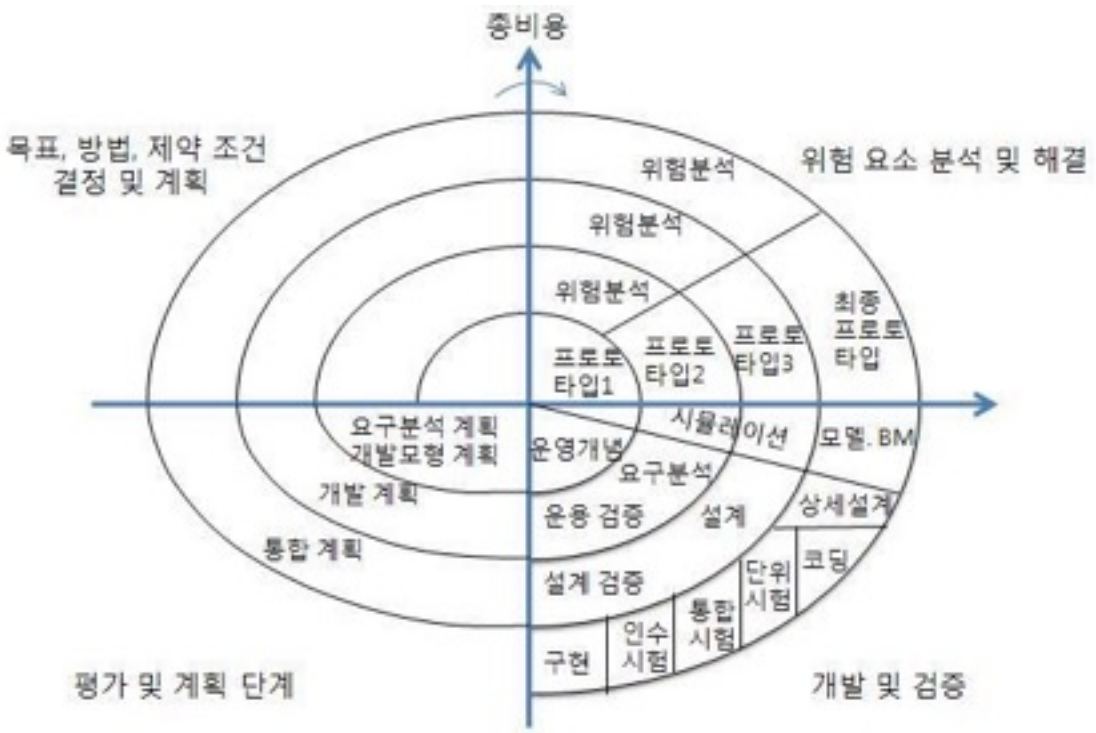


이러한 순서로 개발이 진행되는데
단점으로는 검증 과정에서 요구사항이 제대로 반영되지 않았거나, 수정된다면 그 모든 과정을 다시해야한다.

SI 프로젝트에서는 1년 이상의 프로젝트가 있다고 하는데
이론적으로 생각했을때 1년치의 계획을 한꺼번에 수립했을때 과연 그것이 제대로 지켜질까? 라는 의문점이 든다.

2. 프로토타이핑
프로토타입을 만들어 검증하며 프로그래밍을 진행하는 방식이다. 프로토타입을 만드는 과정에서 시간이 많이 필요하고 커뮤니케이션의 중요도가 올라간다
하지만 폭포수모델에 비하면 실패율이 낮다. 가장 중요한 결점은 프로토타입으로 만들어진 결과물 때문에 코드의 질이 낮아 유지보수가 힘들어진다는 특징이 있다.

3. 나선형모델
폭포수모델과 프로토타이핑을 합친것으로 볼 수 있다
한 과정 한 과정에 대해 폭포수모델과 같은 주기를 빠르게한다. 대규모 프로젝트에 적합하다. 그런데 걱정이 되는 부분은 이미 만들어진걸 수정해야할땐 어떨까? 라는 생각이든다.



3. 개발방법론

1. 애자일
요즘 가장 hot한 개발방법론으론 애자일이 있다. Agile이란 기민한 이란 뜻으로 좋은것응ㄹ 빠르고 낭비없게 만드는 방법들의 집합이다.
예전에 어떤 책에서 읽었떤 애자일은 프로토타이핑과 비슷한 형태였는데 오늘 수업과정을 통해서 정확히 이해한 바로는 프로토타이핑만이 애자일은 아니다.
특히 기억에 남는것은 문서를 작성하고 코딩을 하는것이 아니라 코딩 후에 문서를 작성하여 개발을 가속화하는것이다. 이 문서를 작성할때에도 어떤 순서로 작성할것인지만 결정하고 개요는 나중에 작성하였다.
왜냐하면 개요따위를 작성하는데 시간을 낭비하고 싶지 않았기 때문이다.

2. TDD(Test-driven development) 테스트 주도 개발
SI에 PM으로 근무하는 사람으로부터 10만명의 유저가 프로그램을 실행했을때를 가정하여 소프트웨어를 테스트하는 프로그램이 있다는 이야기를 들었다. 현업에 있을때 다양한 경우의 수를 조합하여 테스트를 했었는데 1만번을 클릭 한 번에 테스트가 된다면 얼마나 좋을까 생각했던 적이 있다.

다양한 잘못된 상황을 코드로 구현하여 테스트하는데 이러한 코드를 구현하는것 때문에 시간이 비효율적일 수도 있겠다고 생각했지만 수업내용에 따르면 여러번 테스트를 하면서 낭비하는 시간 보다 한번 구현하는 시간이 오히려 효율적이라고 한다. 맞는말이다.

3. UML(Unified Modeling Language):
산출물을 명세화, 시각화하여 문서화할때 사용.
class, method간의 관계를 er다이어그램 처럼 관계를 한 눈에 파악할 수 있도록 만들어준다.
중요한것은 표준화 되어있다는 점이다.

4. PDD(Plan-Driven Development) 계획 기반 개발
계획을 세우고 해당 계획을 실행하는데 집중하는 개발 방식

4.형상관리(버전관리) 란 무엇이고 어떠한 방법이 있는가?

법률회사에 제직할 때 증거자료의 새로운 항목을 업데이트할때 해당 폴더를 날짜별로 분류했다.

똑같은 파일인데 작업한 날짜별로 폴더를 새로 만들어 그야말로 버전관리를 했던것이다.
이렇게 한 이유는 잘못 작업한 내용이 있을경우 이전의 날짜로 돌아가기 위한것인데 수정을 하는것 보다 아예 처음부터 새로 만드는것이 더 효율적이기 때문이다.

예전 학원에서 5인 1조로 개발을 했던적이 있는데 학원의 선생님이 이 형상관리 툴에대한 이해가 없어 질문을 해도 제대로 대답을 못하여 그냥 건너뛴적이 있다. 현업에선 있을 수 없는 일이지만, 5명중 가장 꼼꼼한 한명이 5명의 파일을 매일매일 받아서 합쳤던 기억이 있다. 그런데 이 작업이 시간도 많이 필요하고 오류도 많았다.

협업을 위해 이러한 과정을 빠르고 편리하게 해주는 기술이다.

오늘 수업으로 들었던 git이 평소에 쓰긴 썻지만 branch에 대한 이해가 부족했고 merge시 발생되는 conflict에 대한 사전지식이 부족한 상태여서 svn과의 차이를 제대로 이해하지 못했지만 git에 있는 branch와 merge기능은 정말 최고인것 같다. 이 숙제가 끝나면 이 부분에 대해 두어번 더 연습해볼 예정이다.

git, svn, mercurial 이라는 툴들이 있다.
svn은 업데이트 하는 과정이 git에 비해서 심플하다. 그래서 박정훈 강사님의 강의내용 대로 나의 단축키설정, 자바패스 등의 공유되지 않아도 되는 내용까지 한꺼번에 commit이 되어 다른 프로그래머와 디자이너에게 혼란을 줬던적이 있는데 git에서는 이런걸 쉽게 관리할 수 있고 테스트버전, 개발버전, 실제 배포버전, 다음버전을 branch로 관리하면 굉장히 효율적으로 업무를 진행할수 있는것이 git의 특징이다.

5. 느낀점
프로그래머 = 야근 이라는 공식은 굉장히 널리 알려졌다. SI에 있는 친구들을 보면 야근은 그들의 일상이다. 기획자가 이러한 개발방법들에 대해 무지하기 때문이다. 내가 있던 회사 기획자도 이러한 내용을 전혀 몰랐다.
그나마 팀장이 이 부분에 대한 이해가 있었기 때문에 나선형 모델과 같은 방법을 쓰긴했지만 오늘 수업내용에 따르면 그건 제대로 된 나선형모델이 아닌것 같다.

공부를 할때만 해도 시간은 한정되어있고 해야할 일은 많다. 어떻게하면 효율적으로 체력을 낭비하지 않고 더 많은 지식을 공부할지는 늘 중요한 문제이다.