1. 기존의 개발 생명주기에 대해 정리해보고, 최근 트랜드가 되고 있는 개발 방법론에는 무엇이 있는지 알아보기

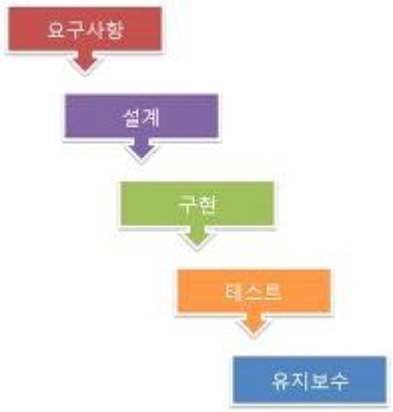
:: 개발 생명주기

1) 주먹구구식 개발 모델(Build-Fix Model)

-요구사항 분석, 설계 등의 단계 없이 일단 개발에 들어간 후 만족할 때까지 수정작업을 진행
장점: 개발 시작이 빠름
단점: 계획이 정확하지 않음
관리자는 프로젝트 진행상황 파악에 어려움
개발 문서가 없기 때문에 개발 및 유지보수에 어려움

2) 폭포수 모델(Waterfall Model)

-순차적으로 소프트웨어를 개발하는 전형적인 모델
-가장 많이 사용되고 대부분의 개발프로젝트에서 가장 많이 사용되는 모델
-각 단계는 [요구사항분석]-[설계]-[구현]-[테스팅]-[유지보수]로 나뉘며 각 단계가 지날 때마다 산출물이 나온다
장점: 각 단계별로 정형화된 접근 방법 가능
체계적인 문서화가 가능하여 프로젝트 진행을 명확하게 할 수 있음
단점: 앞 단계가 완료될 때까지 다음 단계들은 대기 상태여야 함
실제 작동되는 시스템을 개발 후반부에 확인 가능하기 때문에 고객이 요구사항 적용을 확인하는데 많은 시간이 걸림



3) 원형 모델(Prototyping Model)

-폭포수 모델의 단점을 보완한 모델
-점진적으로 시스템을 개발해 나가는 방법
-고객의 요구사항을 정확히 파악하기 어려울 때 사용하는 방법이며, 쉽게 말해 샘플을 먼저

개발한 후 고객의 피드백을 받고 샘플(Prototype=원형)을 폐기하고 다시 개발

장점: 원형을 바탕으로 빠른 개발과 고객의 피드백을 받을 수 있음
단점: 대규모의 프로젝트에는 부적합

4) 나선형 모델(Spiral Model)

-폭포수 모형과 원형 모형의 장점을 수용하고 위험분석을 추가한 점층적 개발 모델
-프로젝트 수행 시 발생하는 위험을 관리하고 최소화 하려는 것이 목적
-중간 중간 마다 위험분석을 하며 문제가 있을 경우 프로젝트를 중단하는 것이 목적
-주로 고 비용의 시스템 개발이나 많은 시간이 걸리는 큰 시스템 구축 시 선택되는 모델

장점: 프로젝트 위험요인을 최소화 할 수 있음
단점: 단계가 명확히 구분되어 있지 않다
개발자가 정확하지 않은 분석을 했을 경우 심각한 문제가 발생
다른 모델에 비해 프로젝트 관리가 복잡하다

5) RAD 모델(Rapid Application Development)

-점진적인 소프트웨어 개발 방식
-컴포넌트를 사용하여 매우 빠르게 폭포수 모델을 적용시키는 방법

장점: 짧은 기간에 기능이 완벽한 소프트웨어 시스템을 개발할 수 있다
단점: 소프트웨어의 주요 기능이 모듈화 되지 않으면 적용할 수 없다

6) 반복적 개발 모델(Iterative Development Model)

-고객의 필수 요구사항을 만족시키는 제품 일부를 먼저 개발
-고객에게는 분석과 훈련의 수단을 제공하고 개발자에게는 학습경험을 제공

장점: 중간단계 제품을 바탕으로 새로운 요구사항을 추가하는 개발로 요구사항 수용이 빠름
단점: 고객의 요구로 변화가 자주 발생하여 구조화가 힘들어질 수 있다

7) V모델 (V Model)

- 폭포수 모형에 시스템 검증과 테스트, 작업을 강조한 모델
- 모듈의 상세설계를 단위테스트 과정에 검증하고 시스템 설계는 통합테스트 단계에 사용자의

요구는 시스템 테스트 과정에서 검증하는 방법

장점: 개발 후 발생하는 오류를 줄일 수 있음
단점: 반복이 없어 변경을 다루기가 쉽지 않음

8) 컴포넌트 기반 모델 (Component Based Development)

-이미 만들어진 컴포넌트를 기반으로 소프트웨어를 개발하는 방법

장점: 개발주기와 개발비용을 대폭 단축하는 생산성향상 효과가 있다
단점: 이미 만들어진 컴포넌트가 없다면 적용시킬 수 없다

:: **개발 방법론**
1) 애자일 개발 프로세스

애자일 소프트웨어 개발(Agile software development) 혹은 애자일 개발 프로세스는 소프트웨어 엔지니어링에 대한 개념적인 얼개로, 프로젝트의 생명주기동안 반복적인 개발을 촉진한다. 최근에는 애자일 게임 보급 등의 여파로 소프트웨어 엔지니어링 뿐 아니라 다양한 전문분야에서 실용주의적 사고를 가진 사람들이 애자일 방법론을 적용하려는 시도를 하고 있다.

- 종류
· 익스트림 프로그래밍(Extreme Programming, XP) - 애자일 개발 프로세스의 대표자로 애자일 개발 프로세스의 보급에 큰 역할을 하였다. 이 방법은 고객과 함께 2주 정도의 반복개발을 하고, 테스트와 우선 개발을 특징으로 하는 명시적인 기술과 방법을 가지고 있다.

· 스크럼 - 30일마다 동작 가능한 제품을 제공하는 스프린트(Sprint)를 중심으로 하고 있다. 매일 정해진 시간에 정해진 장소에서 짧은시간의 개발을 하는 팀을 위한, 프로젝트 관리 중심의 방법론이다.
· 크리스털 패밀리 - 이 방식은 프로젝트의 규모와 영향의 크기에 따라서 여러종류의 방법론을 제공한다. 그중에서 가장 소규모 팀에 적용하는 크리스털 클리어는 익스트림 프로그래밍만큼 엄격하지도 않고 효율도 높지 않지만, 프로젝트에 적용하기 쉬운 방법론이다.

· Feature-Driven Development - feature마다 2주정도의 반복 개발을 실시한다. Peter Coad가 제창하는 방법론으로써, UML을 이용한 설계 기법과도 밀접한 관련을 가진다.
Adaptive Software Development, ASD - 소프트웨어 개발을 혼란 자체로 규정하고, 혼란을 대전제로 그에 적응할 수 있는 소프트웨어 방법을 제시하기 위해 만들어진 방법론이다. 내용적으로는 다른 방법론들과 유사하지만, 합동 애플리케이션 개발(Joint Application Development, 사용자나 고객이 설계에 참가하는 개발 방법론)을 사용하고 있는 것이 조금 다르다.

· 익스트림 모델링 - 익스트림 모델링은 UML을 이용한 모델링 중심 방법론이다. 다만, 여타 모델링 방법들과는 달리, 언제나 실행할 수 있고 검증할 수 있는 모델을 작성하는 공정을 반복해서, 최종적으로는 모델로부터 자동적으로 제품을 생성하게 한다.

※애자일 개발 프로세스와 전통적인 개발 프로세스와의 차이

전통적인 개발 프로세스들은 폭포수 모델과 계획 기반 개발을 따르는 반면, 애자일 개발 프로세스는 그에 반한다는 점에서 가장 큰 차이를 가진다.

폭포수 모델과 계획 기반 개발 기법들은, 일련의 차례와 탄탄한 계획을 기반으로 하여 개발을 진행시킨다. 이것은, 이해하기도 쉽고 사용하기도 쉬운 바람직한 기법이기도 하지만, 이로 인해서 많은 부작용이 생길 수 있다. 가장 큰 부작용이 발생할 때는, 계획대로 진행되지 않을 경우이다. 이럴 경우에는 다음과 같은 부작용이 발생하게 된다.

· 납기일 전 철야
· 철야에도 불구하고 납기일 지연
· 지연에 따른 비난과 스트레스로 개발자 에너지 소진
· 결국 납품된 솔루션은 고객의 요구를 충족하지 못함

이런 부작용은 근본적인 개발 프로세스 접근법의 차이에서 나타난다. 전통적인 개발 프로세스들은 공업에서 사용하는 정형적 프로세스 제어 모델을 따르고 있다. 정형적 프로세스 제어 모델은, 동일한 입력에 대해서 동일한 결과가 기대 될 경우에 적합하다. 하지만, 소프트웨어를 포함한 IT의 개발은 경험적 프로세스 제어 모델로 접근할 필요가 있다. 경험적 프로세스 제어 모델은 항상 불확실성을 수반하고 포용하고 있다. 애자일 개발 프로세스는 경험적 프로세스 제어모델로 개발을 관리한다.

2) 테스트 주도 개발(Test-driven development TDD)

테스트 주도 개발은 매우 짧은 개발 사이클을 반복하는 소프트웨어 개발 프로세스 중 하나이다. 우선 개발자는 바라는 향상 또는 새로운 함수를 정의하는 (초기적 결함을 점검하는) 자동화된 테스트 케이스를 작성한다. 그런 후에, 그 케이스를 통과하기 위한 최소한의 양의 코드를 생성한다. 그리고 마지막으로 그 새 코드를 표준에 맞도록 리팩토링한다. 이 기법을 개발했거나 '재발견' 한 것으로 인정되는 Kent Beck은 2003년에 TDD가 단순한 설계를 장려하고 자신감을 불어넣어준다고 말하였다.

3) 계획 기반 개발(Plan-driven development)

계획 기반 개발은 소프트웨어를 개발하는 과정에서 계획을 세우고 그 계획을 실천 하는데 많은 시간과 노력을 할애하는 개발 방법이다.
장점: 소프트웨어 개발이 조금 더 예측가능해지고, 효율적이게 된다
단점: 너무 계획에 치중을 하다 보니 개발 방법 자체가 너무 형식에만 신경을 쓰고 얽매이게

된다는 것과 이미 세워진 계획 앞에 융통성을 발휘 할 수 없는 부분이 발생할 수 있다
반드시 따라야 하는 단계들이 존재하기 때문에 전체적인 개발의 흐름 자체를 느리게 만들기도 한다

2. 형상관리(버전관리)란 무엇이고, 형상관리(버전관리) 방법에는 어떤 것들이 있는가?

소프트웨어 구성 관리(영어: Software Configuration Management) 또는 형상 관리는 소프트웨어의 변경사항(버전관리)을 체계적으로 관리하는 것을 말한다. 형상 관리는 일반적인 단순 버전관리 기반의 소프트웨어 운용을 좀 더 포괄적인 학술 분야의 형태로 넓히는 근간을 이야기한다.

- 형상관리(버전관리) 종류
:: 상용
IBM Rational ClearCase
Perforce
PTC Integrity

::비상용(오픈소스)
Subversion(SVN)
CVS
Git

위에 나열한 시스템 외에도 수많은 버전 관리 시스템이 존재한다

3. 느낀 점

소프트웨어 개발 방법론을 넘어서 조직 관리 방법론까지 영향을 미쳤다고 하는 스크럼을 매일 아침 경험하여 보며 조금이나마 애자일의 장점을 느껴본 것 같다. 다른 분들께선 어떻게 학습을 하고 계신지 또는 어떤 부분에서 재미를 느끼고 어려움을 느끼시는지 이야기하는 시간을 가지고 하루를 시작하니 수업에 더 집중하게 되는 것 같다.

이전에 팀프로젝트를 하며 팀원들이 밤새 수정해 온 소스를 취합하느라 많은 시간을 할애했던 경험과 업데이트 하며 버전 관리에서 어려움을 느꼈던 경험이 있어 현업에서는 어떻게 협업을 하는지 궁금했었고, 이번에 Git을 배우며 그 의문점이 조금은 풀린 듯하다.