# 2강. 소프트웨어 프로세스

| 메뉴 | 학습에 앞서 |
|---|---|

■ 학습개요

소프트웨어 개발 프로세스는 개발 활동들의 절차이며 프로세스 모델은 프로세스의 추상화된 표현이다. 개발 조직이나 응용 분야에 맞게 적절히 변형되어 적용될 수 있는 프로세스 모델로서 폭포수 모델, 반복 진화형 모델, 점증적 모델, 나선형 모델 및 V 모델 등을 학습할 것이다. 또한 요구사항이 자주 바뀌는 중소형의 시스템에 적합한 것으로, 소프트웨어 엔지니어에게 큰 흥미를 주고 있는 애자일 방법론의 철학과 실천 기술들을 알아볼 것이다.

■ 학습목표

| | |
|---|---|
| 1 | 소프트웨어 개발 프로세스와 프로세스 모델의 정의를 기술할 수 있다 |
| 2 | 요구분석, 설계 및 코딩과 같은 보편적인 개발 활동에 관해 설명할 수 있다. |
| 3 | 잘 알려진 프로세스 모델들을 살펴보고 각 모델이 가지는 장점과 단점들을 숙지한다. |
| 4 | 애자일 방법론의 철학과 XP 방법론의 실천 기술들을 습득한다. |

■ 주요용어

| 용어 | 해설 |
|---|---|
| 소프트웨어 프로세스 모델 | 실제 프로세스를 추상화하여 간략히 표현한 것 |
| 폭포수 모델 | 전통적 소프트웨어 개발 프로세스 모델로 선형 순차 모델 |
| 알파 테스트 | 완전히 개발된 시스템을 개발 현장에서 테스트하는 것으로 주문형 제품의 경우 개발자와 고객 사이에 인수에 대한 동의가 이루어질 때까지 수행함 |
| 나선형 모델 | 전체 생명주기에 걸쳐 프로토타이핑과 위험 분석을 통해 위험을 관리하고 최소화하려는 목적을 가짐 |
| V 모델 | 폭포수 모델의 확장된 형태로 분석이나 설계 등의 단계와 상응하는 테스트 단계가 존재함 |
| 애자일 방법론 | 변화를 수용하고 협업을 강조하며 제품의 빠른 인도를 강조한 반복적 방법 |

| 메뉴 | 학습하기 (PPT 내용) |
|---|---|

2.1 소프트웨어 프로세스 개요

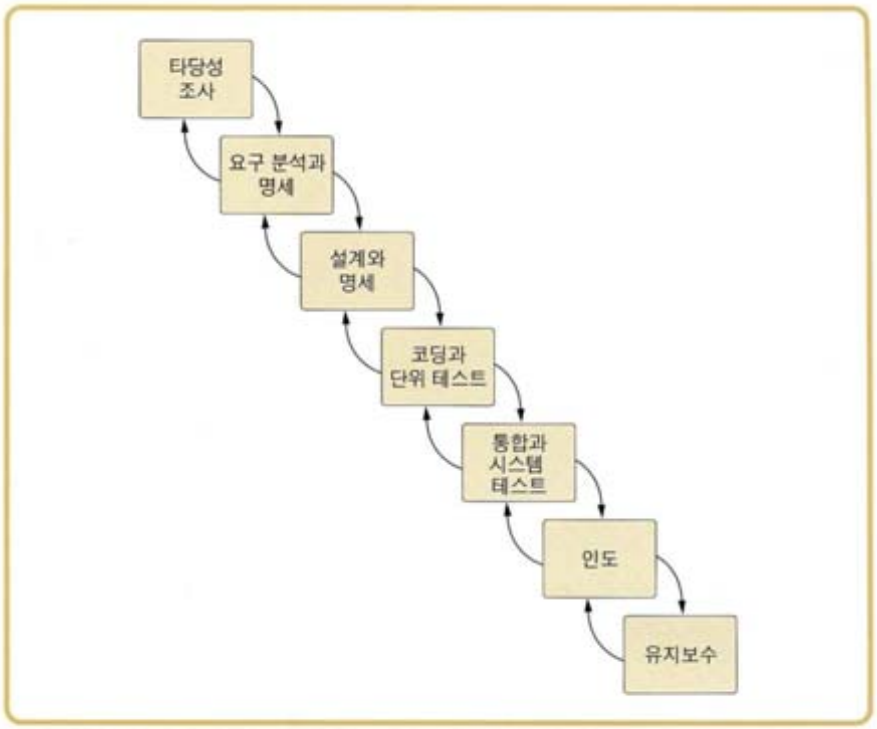
소프트웨어 프로세스
- 소프트웨어 시스템을 개발하거나 유지보수할 목적으로 수행되는 활동 일체 또는 절차
- 누가 어떤 활동을 언제 하는지를 정의
- 개발 조직은 적당한 프로세스 모델을 보유하여 공통의 개발 문화와 공통의 기술을 제공해야 함
- 프로세스 모델이 존재해야 하는 이유
  - 전체 프로세스의 이해에 도움을 줌
  - 구조화된 방법을 개발에 적용
  - 자원사용에 대한 사전 계획을 가능하게 함
  - 자원 사용을 통제할 수 있음
  - 시스템 개발 과정을 추적할 수 있음

주요 프로세스 활동
- 소프트웨어 명세: 소프트웨어의 기능과 운영상 제약 조건을 정함
- 소프트웨어 개발: 소프트웨어를 설계하고 프로그래밍 함
- 소프트웨어 검증: 고객이 원하는 것을 수행하는지 검사함
- 소프트웨어 진화: 소프트웨어를 변경함

고려 사항
- 개발 조직마다 프로세스가 다름
- 프로젝트 유형에 따라 다름
- 대형 시스템의 경우에는 부분마다 다른 프로세스를 적용

2.2 폭포수 모델(waterfall model)

폭포수 모델
- 선형 순차 모델, 고전적 소프트웨어 생명 주기라고도 함
- 기본적으로 각 단계는 병행 수행되지 않고 거슬러 반복되지 않으며 한 방향으로 진행됨
- 실제로 수정을 위한 재작업을 위해 앞 단계로의 피드백이 불가피




[그림 2-1] 폭포수 모델

타당성 조사
- 문제점을 파악하고 해결 방안을 제시하여 투입 비용 대비 이익을 평가
    조직 측면의 타당성: 조직의 전략적 목표를 충족하는가
    경제적 타당성
    기술적 타당성: 정해진 시간 안에 현재의 기술 수준으로 개발할 수 있는가
    운영의 타당성: 운영/사용 능력과 다른 시스템과의 연동 가능성 판단
- 시간적 제약과 정신적 압박감이 존재
- 타당성 조사 보고서: 문제의 정의, 기술적/경제적 타당성, 해결 방안과 기대 효과, 비용과 인도 날짜 등을 포함

요구 분석과 명세
- 프로젝트의 성패를 좌우하는 중요한 단계로 무엇을 개발할지 결정
- 요구사항이란
    문제의 해결을 위해 시스템이 갖추어야 하는 조건이나 능력
    명세서나 계약서 또는 표준에 명시되어야 함
- 요구사항 명세서(SRS) : 의뢰자와 개발자 간의 의사소통 수단으로 정확하고 일관성 있으며 완전해야 함
- 요구사항 명세서의 내용
    시스템의 목적과 범위:  문제점을 구체적으로 기술하고 대안을 제시
    기능적 요구사항
    비기능적 요구사항

요구사항 검증을 위한 평가 기준

설계와 명세
- ‘what'을 ’how'로 변환하는 작업: 요구사항들을 구현 작업에 적합하게 명확하고 조직화된 구조로 바꾸는 것
- 설계 단계의 구분
  시스템 아키텍쳐 설계, 프로그램 설계, 인터페이스 설계
- 설계 방법
  전통적 설계 방법: 구조적 분석의 결과에 구조적 설계 방법을 적용
  객체지향 설계 방법

코딩과 단위 테스트
- 설계 결과를 프로그램으로 작성하고 구현된 모듈이 명세서를 만족하는지 확인함
- 고려 사항
  코딩 표준의 준수
  테스트 계획, 테스트 방법과 테스트 수준
  코드 인스펙션

통합과 시스템 테스트
- 모듈들을 통합하고 최종적으로 전체 시스템이 요구사항을 만족하는지 확인
- 통합과 테스트 작업은 점증적 방식으로 함
- 알파 테스트
  소프트웨어 개발 현장에서 수행
  일반 소프트웨어의 경우 독립적 테스트 팀이 수행하며 베타 버전을 릴리스 함
  주문형 소프트웨어의 경우 실제 사용 환경을 시뮬레이션한 후 개발자와 고객 사이에 제품의 인수에 대한 동의가 이루어질 때 까지 수행(인수 테스트)
- 베타 테스트
  고객의 실제 환경에서 수행
  제품의 출시 전에 미리 제품을 평가 받음

인도와 유지보수
- 유지보수: 고객에게 전달한 후 폐기되기 전까지 일어나는 수정 및 보완 활동
- 유지보수의 네 종류
  수정 유지보수(corrective maintenance)
  적응 유지보수(adaptive maintenance)
  완전 유지보수(perfective maintenace)
  예방 유지보수(preventive maintenace)
- 유지보수 대신 소프트웨어 진화라는 표현을 사용하기도 함
- 유지보수 노력이 적게드는 소프트웨어를 개발하는 것이 중요

폭포수 모델의 장단점

-장점

선형 모델로 단순하고 이해가 쉬움
단계별로 정형화된 접근 방법과 체계적 문서화가 가능
프로젝트 진행 상황을 명확히 알 수 있음

-단점

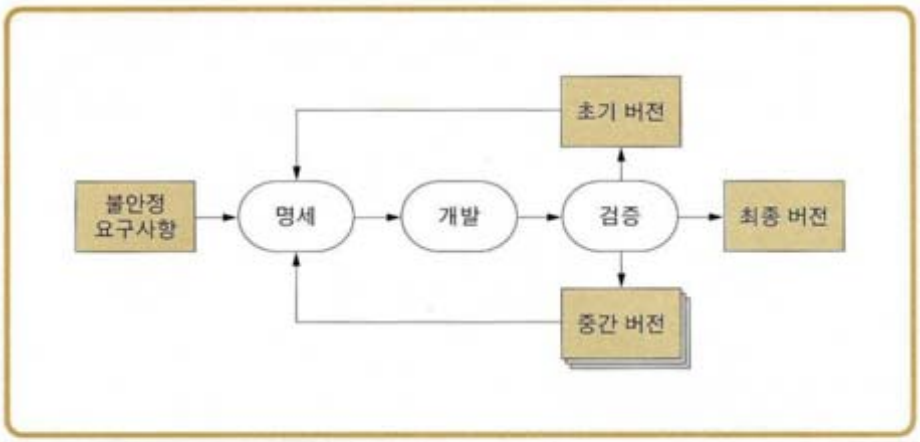
요구사항을 완벽하게 작성해야만 함
변경을 수용하기 어려움
시스템의 동작을 후반에나 볼 수 있음
대형 프로젝트에 적용하기 어려움
문서화를 위한 노력이 지나침
위험 분석 결여, 일정의 지연 가능성이 큼

2.3 반복 진화형 모델

반복 진화형 모델

- 초기 버전을 만들고 요구사항을 정제하여 새로운 버전을 개발하는 작업을 반복하면서 시스템을 완성해 가는 방식

분명한 요구사항과 시스템의 범위를 정하는 노력이 선행 됨
한번의 진화 단계에서 프로토타이핑을 통해 요구사항을 보완
최종 버전이 나온 후 유지보수 단계로 들어감




[그림 2-2] 반복 진화형 모델

반복 진화형 모델의 장단점

- 장점

요구사항이 완성되지 못한 경우에도 초기 버전을 만들고 점차적으로 명확한 요구사항을 도출함

- 단점

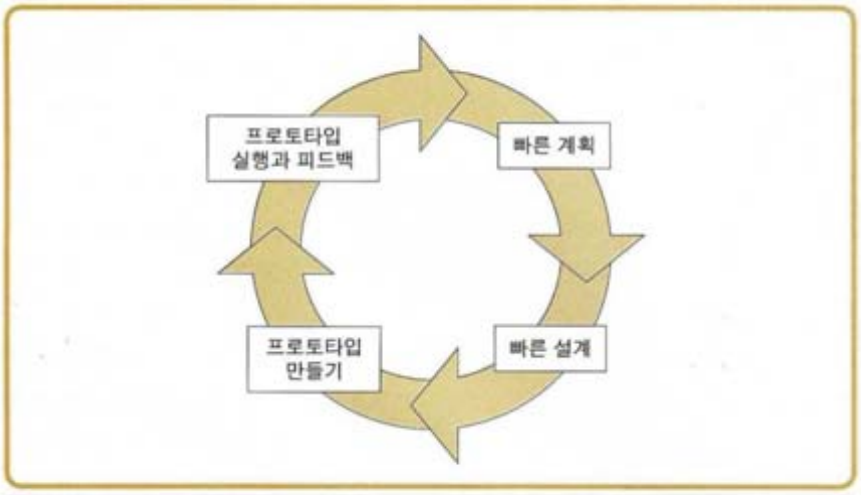
관리적 관점에서 개발 비용의 예상이 힘들고 재작업이 잦아지면 종료 시점이 늦춰질 가능성이 큼
공학적 관점에서 잦은 수정 작업은 소프트웨어 구조에 악영향을 주어 유지보수에

문제가 생기게 됨

고려 사항
- 50만 라인 이하의 중소형 시스템에 적합
- 대형 시스템에서 사용자 인터페이스를 개발하는 경우에 적합
- 폭포수 모델을 사용하는 경우에도 프로토타이핑 방법을 먼저 적용하기도 함

점증적 모델과의 차이점
- 진화형 모델은 요구사항이 불안정하고 명확하지 못할 때 사용
  명확히 이해할 수 없는 새로운 기술을 적용할 때 사용
  한꺼번에 모든 기능을 포함해 인도해야 하는 경우
- 점증적 모델은 요구사항의 중요도에 따라 작업 순서를 정함

프로토타이핑(prototyping) 방법
- 소프트웨어 요구사항을 파악하기 위한 좋은 방법
  일반적으로 사용자는 소프트웨어의 입출력과 처리 기능을 자세히 요구하지 못함
  개발자도 알고리즘의 효율성이나 운영체제 호환성 및 상호작용 형태를 정확히 파악하기 힘듦



[그림 2-3] 프로토타이핑 과정

프로토타이핑의 종류
- evolutionary prototyping
  잘 알고 있는 부분부터 시작하여 계속적으로 발전시켜 완제품을 만드는 방법
- throwaway prototyping
  프로토타입을 고객과의 의사소통 수단으로만 사용하는 경우

프로토타이핑 방법의 장단점
- 장점
  프로젝트의 실현 가능성, 소프트웨어의 개발 가능성을 판단할 수 있음

개발자와 사용자 간의 의사소통이 명확해 짐
기능적 요구사항 외에도 성능이나 유용성 등의 품질 요구를 할 수 있음
시스템을 미리 사용함으로써 사용자 교육 효과가 있음
개발 단계에서 유지보수가 일어나는 효과가 있음

- 단점

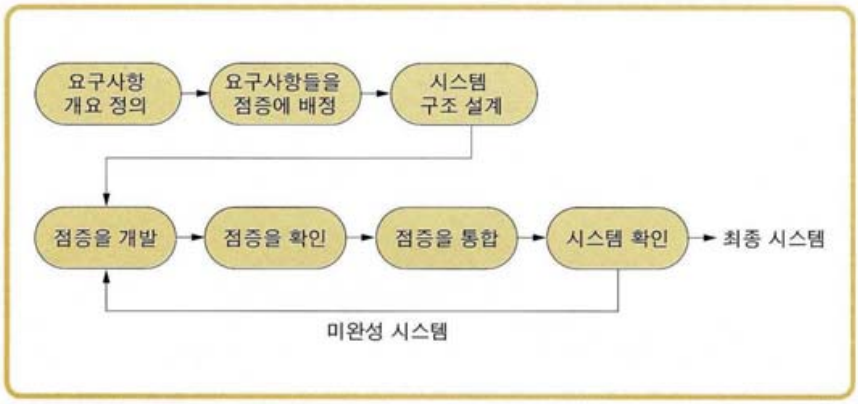
문서화가 힘들며 관리자는 진척 사항을 제어하기 힘들어 짐

## 2.4 점증적 모델(incremental model)

점증적 모델

- 여러 개의 모듈들로 분해하고 각각을 개발하여 인도하는 방식

선형 순차 모델을 여러번 적용하고 그 결과를 조합함
각 모듈을 점증이라 함
핵심 모듈을 먼저 개발하고 인도함




[그림 2-4] 점증적 개발 모델

- 진화형 모델과 다른 점

여러 점증을 동시에 개발할 수 있음
시스템 릴리스가 시간차를 두고 계속됨

장단점

- 장점

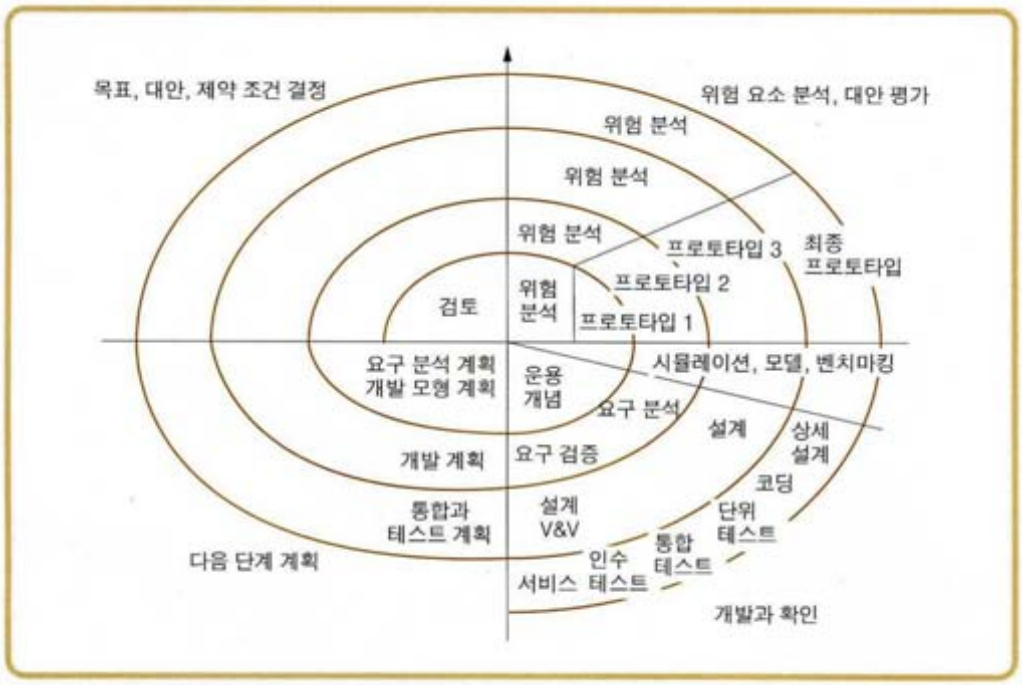
중요한 점증이 먼저 개발되므로 사용자는 시스템을 이른 시기에 사용
릴리스 방식이 요구사항 변화에 대응하기 용이함
점증들은 점차로 규모와 기능이 축소되어 관리가 어렵지 않음
먼저 개발되는 중요 부분이 반복적으로 테스트됨

- 단점

기능적으로 분해하기 어려울 수 있음
적당한 크기의 점증들로 나누기 어려움
점증을 개발하기 전에 명확한 요구사항을 정의해야 함

2.5 나선형 모델(spiral model)

나선형 모델
- 전체 생명주기에 프로토타이핑과 위험 분석을 계획적으로 사용하여 위험을 최소화하려는 목적을 가짐
- 반복 진화형 모델의 확장 형태로 주기적으로 순환되는 구조
  가장 중앙의 원을 타당성 조사 단계, 다음 원을 요구사항 정의 단계, 그 다음 원을 설계 단계
  각 단계는 목표와 대안->대안의 평가(위험 요소 분석)->개발->다음 단계 계획




[그림 2-5] 보엠의 나선형 모델

고려 사항
- 나선형 모델은 위험 관리를 지원하는 프로세스의 프레임워크(프로세스 생성기)
- 위험 관리에 비용이 드나 가치있음
  위험은 프로젝트 수행이나 제품의 품질에 악영향을 주는 잠재 요소
  실험적이고 복잡한 대형 프로젝트에 적합

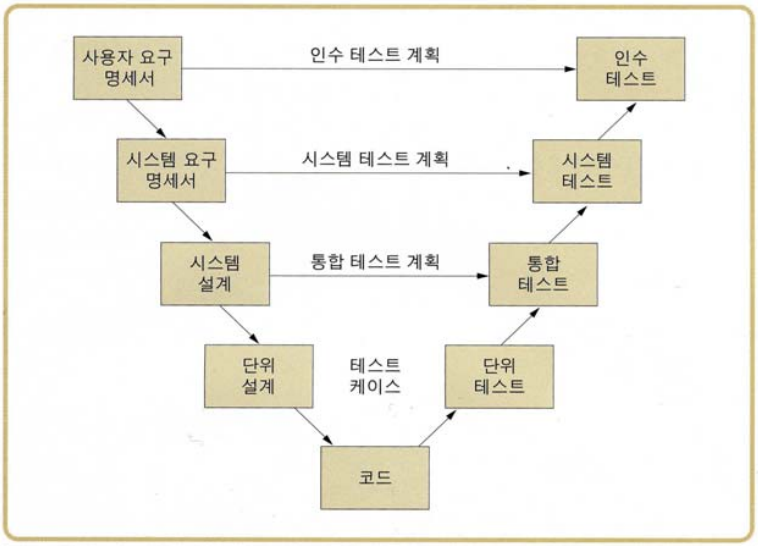
장단점
- 장점
  대형 프로젝트에서 위험 관리를 통해 성공 가능성을 높일 수 있음
  프로젝트 특성이나 개발 조직에 맞게 변형될 수 있음
- 단점
  경험이 부족하여 충분히 검증되지 못했음

모델 자체가 복잡함
프로젝트 관리가 어려움

## 2.6 V 모델

V 모델
- 폭포수 모델의 확장 형태로 분석이나 설계 단계에 상응하는 테스트 단계가 존재
- 테스트 작업을 중요시하여 적정 수준의 품질 보증을 지원함

특징
- 개발 단계의 작업을 확인하기 위해 테스트 작업을 수행
- 생명 주기 초반부터 테스트 작업을 지원
- 코드뿐만 아니라 요구사항과 설계 결과도 테스트할 수 있어야 함
- 폭포수 모델에 비해 반복과 재처리 과정이 명확함
- 테스트 작업을 단계별로 구분하므로 책임이 명확해 짐



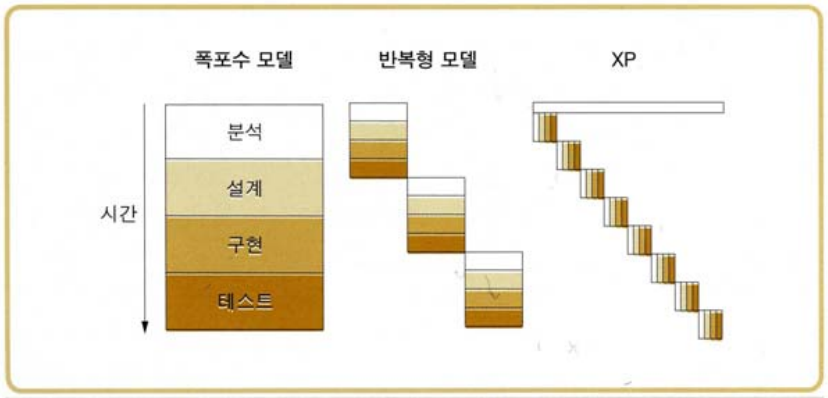
[그림 2-6] V 모델

## 2.7 애자일 방법(agile method)

애자일 방법
- 변화를 수용하고, 협업을 강조하며, 제품의 빠른 인도를 강조하는 반복적 방법
  문서화 작업보다 코드를 강조함, 문서보다 소프트웨어 자체에 집중할 것
  요구사항의 변화는 불가피하고 이것에 대응하는 것이 현실적
  기존의 개발 프로세스는 설계 기간이 길며 재작업시 오버헤드가 크다는 생각
- 요구사항이 바뀌기 쉬운 중소형의 비즈니스 시스템이나 전자 상거래 응용에 적합

익스트림 프로그래밍(eXtreme Programming)
- 대표적인 애자일 방법의 하나

- 좋은 실천 기술들을 적극적으로 적용할 것을 주문




[그림 2-7] 폭포수 모델 · 반복형 모델과 XP의 비교

XP의 실천 기술(practices)
- 작고 빈번한 릴리스, 빠른 피드백과 지속적 개선
- 고객도 개발 팀의 일원이 됨
- 프로세스 중심이 아닌 사람 중심의 방법
- 단순한 설계와 테스트 선행 개발
- 코드의 품질 개선을 위해 리팩토링을 제안

짝 프로그래밍(pair programming)
- 두 사람이 짝이 되어 한 사람은 코딩을 다른 사람은 검사를 수행
- 장점
  - 코드에 대한 책임을 공유
  - 비형식적인 검토를 수행
  - 코드 개선을 위한 리팩토링을 장려함
  - 생산성이 떨어지지 않음

테스트 선행 개발(test-first development)
- 테스트 케이스를 먼저 작성하고 이것을 통과하는 코드를 만들 것
- 태스크 별로 테스트 케이스를 만듦
  - 요구사항은 스토리 카드로 표현되고 스토리 카드는 태스크들로 분해됨
  - 요구사항과 코드와의 관계가 명확해 짐
- 통합 테스트를 강조하여 기존 소프트웨어에 오류가 유입되지 않도록 함
  - 테스트 케이스의 재사용

| 메뉴 | 연습문제 |
|---|---|

1. 소프트웨어의 개발 단계 중 요구분석 단계에 대한 설명이 아닌 것은?
   ① 소프트웨어 개발의 실질적 첫 단계이다.
   ② 소프트웨어가 수행해야 하는 기능을 기술하는 단계이다.
   ③ 문제점의 해결을 위해 목표를 명확히 도출하는 단계이다.
   ④ 소프트웨어의 내부 구조를 기술하는 단계이다.

<정답> ④
<해설> 소프트웨어의 내부 구조를 기술하는 단계는 설계 단계이다.

2. 사용자가 개발자 환경에서 개발자가 보고 있는 상황에서 행하는 시스템 테스트 방법은?
   ① 알파 테스트 ② 베타 테스트
   ③ 단위 테스트 ④ 구조 테스트

<정답> ①
<해설> 주문형 소프트웨어의 경우 알파 테스트를 인수 테스트라 한다. 일반 소프트웨어의 경우에는 사용자가 불특정 다수이기 때문에 독립적 테스트 팀이 수행하는 것이 보통이다.

3. 장래의 유지보수성 또는 신뢰성을 개선하거나 오류로 연결될 수 있는 잠재적 결함을 찾아 수정함으로써 미리 예방해 두는 목적의 유지보수 형태는?
   ① 수정 유지보수 ② 적응 유지보수
   ③ 완전 유지보수 ④ 예방 유지보수

<정답> ④
<해설> 환경의 변화를 반영하기 위한 것이 적응 유지 보수, 발견하지 못했던 오류를 수정하는 것이 수정 유지보수, 새로운 기능을 추가하거나 기존 기능을 개선하는 것이 완전 유지보수이다.

4. 폭포수 모델의 문제점이 아닌 것은?
   ① 변경을 수용하기 어려움
   ② 문서화를 위해 지나친 노력을 소모하는 경향

③ 초기에 요구사항을 명확히 작성해야 함
④ 프로젝트 진척사항의 관리가 어려움

<정답> ④
<해설> 선형 순차 모델로 단순하며 각 단계별 산출물을 통해 진행상황을 알 수 있음

5. 나선형 모델에 대한 설명으로 옳지 않은 것은?
① 소프트웨어를 개발하면서 발생할 수 있는 위험을 관리하고 최소화
② 반복적으로 개발을 지원하며 프로토타이핑 기법을 이용함
③ 작업 순서는 선형적이며 타당성 조사, 요구 분석, 설계, 코딩, 유지보수 순임
④ 대형 소프트웨어에 적합하며 프로젝트나 조직에 맞게 변형될 수 있음

<정답> ③
<해설> 나선형 모델은 중앙에서 시작하여 나선형으로 진행되는 반복 진화형의 확장 형태임

6. 애자일 방법과 관련이 없는 것은?
① 높은 안정성과 신뢰성을 요하는 소프트웨어
② 반복적 개발 방법
③ 익스트림 프로그래밍
④ 단순한 설계

<정답> ①
<해설> 보안이나 안전성 등을 요구하는 시스템의 개발에는 적당하지 않으며 중소형의 비즈니스 시스템이나 전자 상거래 응용의 개발에 적합하다.

| 메뉴 | **정리하기** |
|---|---|

## 1. 소프트웨어 프로세스란 무엇인가?

□ 소프트웨어 시스템을 개발하거나 유지보수할 목적으로 수행되는 활동들의 절차

## 2. 소프트웨어 프로세스 모델이란 무엇인가?

□ 실제 프로세스의 추상화된 요약 표현

3. 설계 단계는 무엇인가?

□ 요구사항 명세서에 표현된 ‘what'을 ’how'로 변환하는 단계로 요구사항들을 구현 작업에 적합하게 명확하고 구조화된 표현으로 바꾸는 단계

4. 점증적 프로세스 모델은 무엇인가?

□ 요구사항의 중요도에 따라 시스템을 여러 개의 모듈들로 분해한 후 각각을 개발하여 인도하는 방식

5. 나선형 프로세스 모델의 가장 큰 특징은 무엇인가?

□ 프로젝트 수행 시 발생하는 위험을 관리하고 최소화하려는 목적을 가지는 것

6. XP의 주요 실천 기술은 무엇인가?

□ 개발 주기의 잦은 반복, 고객의 참여, 짝 프로그래밍, 단순한 설계, 리팩토링 등

7. 테스트 선행 개발이란 무엇인가?

□ 테스트 케이스를 먼저 작성한 후 이것을 통과하는 코드를 만드는 것