소프트웨어 공학개론

Introduction to Software Engineering

2022-1학기 (Spring)
선문대학교 AI소프트웨어학과

3월 15일 (화)

강의 목차

선문대학교

Part I. 개념

- 2. 소프트웨어개발공정
  - 2.1 소프트웨어개발공정이란?
  - 2.2 폭포수 모델
  - 2.3 진화형 프로세스 모델
  - 2.4 프로세스프로그래밍

- 연습 문제

Part II. 활용

Part III. 실습

2. 소프트웨어개발공정

개요

- 수많은 사람들로 프로젝트를 구성 및 협력
  - 대규모 프로그램개발을 성공시키기 위해서 필요

- 프로젝트 구성원 간 개발진행방법의 사전 규정 및 이해 필요
  - "어떤 순서로 진행하는가",
  - "어떤 도구를 사용하는가",
  - "어떤 단계에서 어떤 것을 만들어서 진행시켜 가는가" 등

- 다양하게 제안된 소프트웨어개발모델
  - 프로젝트에서 축적된 성공과 실패 경험 토대

- 살펴볼 내용
  - 대표적인 소프트웨어개발모델 소개
  - 각각의 장점과 문제점들에 대해서 설명

2.1 소프트웨어개발공정이란?

개요

- 소프트웨어개발공정
  - 소프트웨어개발을 어떻게 진행할 것인가를 규정한 것

- 소프트웨어개발에서
  - 각 작업공정
    - 개발프로세스 또는 단순히 프로세스라고 부른다
  - 개발공정전체
    - 소프트웨어프로세스 라고 부른다

- 소프트웨어프로세스를 규정하면,
  - 각 프로세스의 작업과 그다음 작업으로 연결하기 위한 산출물(문서, 도표, 프로그램 등)이 결정

2.1 소프트웨어개발공정이란?

선문대학교

목차

- 각 프로세스의 개요와 소프트웨어프로세스모델에 대해서 설명

  - 2.1.1 대규모 소프트웨어의 개발
    - 1) 요구사항분석
    - 2) 설계
    - 3) 구현
    - 4) 테스트/검증
    - 5) 운용/유지보수

  - 2.1.2 소프트웨어프로세스모델

2.1 소프트웨어개발공정이란?

2.1.1 대규모 소프트웨어의 개발

- 대규모 소프트웨어의 개발에서는
  - 개발계획 작성
    - 개발목적, 개발기간, 개발비용, 개발형태, 이용기간, 이용형태, 개발 후의 취급방법 등을 규정
  - 개발계획을 토대로 개발체제 정비
  - 이후 아래 작업공정에 의해 소프트웨어 개발/운용
    - 1) 요구사항분석 공정
    - 2) 설계
    - 3) 구현
    - 4) 테스트/검증
    - 5) 운용/유지보수

2.1 소프트웨어개발공정이란?

2.1.1 대규모 소프트웨어의 개발

- 1) 요구사항분석
  - 어떤 소프트웨어를 작성해야되는가를 명확하게 하여 요구사양을 결정하는 공정
    - 고객과 이용자가 생각하고 있는 요구사항 도출
    - 개발자의 관점에서 구현성, 비용, 개발기간의 타당성 검토
    - 구현할 소프트웨어의 사양 기술

2.1.1 대규모 소프트웨어의 개발

- 2) 설계
  - 설계공정
    - 기본설계 (외부설계 또는 시스템설계)
      - 외부로부터 시스템을 바라보았을 때의 사양 설계
      - 예시
        - 전체의 골격이 되는 기본구조(아키텍처)
        - 유저인터페이스
        - 소프트웨어가 구현해야 되는 기능의 설계
      - 기본설계사양(외부설계사양 또는 시스템 설계사양) 결정
    - 상세설계 (내부설계 또는 프로그램설계)
      - 기본설계사양을 토대로 모듈구조 설계
      - 상세설계사양(내부사양 또는 프로그램사양) 결정

2.1 소프트웨어개발공정이란?

2.1.1 대규모 소프트웨어의 개발

- 3) 구현
  - 프로그래밍언어를 사용하여 요구사양 및 설계사양에 대응하는 프로그램 기술(코딩) 공정

- 4) 테스트/검증
  - 프로그램이 올바르게 동작하는지를 검사하기 위하여 코딩완료된 프로그램을 테스트하는 공정
    - 단위테스트: 프로그램단위로 수행
    - 통합테스트: 여러 개의 프로그램을 모아서 수행
    - 시스템테스트: 시스템전체를 수행
    - 인수테스트: 개발 완료 후에 '개발 의뢰처'에서 수행

- 5) 운용/유지보수
  - 소프트웨어를 운용하고 유지관리하는 공정
    - 운용시에 이상이 발생한 경우, 소프트웨어 수정/변경
    - 소프트웨어 운용에 앞서, 문서 작성 (예: 운용매뉴얼, 조작설명서, 유지보수매뉴얼 등)

2.1.2 소프트웨어프로세스모델

- 팀단위 소프트웨어 개발
  - 팀 멤버들은 팀에서 정한 소프트웨어프로세스에 따라 서로의 산출물 공유
    - 개발공정을 이해하고 의사소통을 하지 않으면, 요구사항에 적합한 소프트웨어 개발에 어려움 발생
      - 개발순서, 사용도구, 개발환경, 문서체계 등을 사전에 규정 및 안내 필요
  - 소프트웨어프로세스의 유형 (소프트웨어프로세스모델 또는 소프트웨어생명주기모델)
    - 개발 전에 각 프로젝트에 적합한 유형 결정
    - 프로젝트 관리에 이용 (예: 해당 프로젝트가 예정대로 진행되고 있는지 등)
    - 유형에 따라, 개발 기간 및 비용 등에 차이 발생 (예: 소프트웨어가 이용가능하게 될 때까지의 기간과 개발비용)
    - 생명주기(Life Cycle)
      - 일반적으로 인간의 일생을 말함
      - (소프트웨어 등) 상품이 출시되어 판매완료 되기까지의 주기 등을 나타내는 용어

2.2 폭포수모델

개요

- 폭포수모델(Waterfall Model)
  - 초기에 개발된 전통적인 소프트웨어프로세스모델
    - 간결하고 이해하기 쉬움
    - 오래전부터 수많은 개발현장에서 사용
    - 현재까지도 사용되고 있음
  - 그러나, 많은 문제점들이 지적되어 왔음

목차

- 2.2.1 폭포수모델의 개발순서
- 2.2.2 폭포수모델의 문제점

2.2 폭포수모델

2.2.1 폭포수모델의 개발순서

- 폭포수모델 (위에서 아래로 폭포수가 흘러떨어지는 자연 현상에 착안)
  - 각 공정을 한단계씩 순차적으로 진행
    - 공정: 요구분석에서 설계(기본설계, 상세설계), 구현, 테스트 및 검증, 운용 및 유지보수
  - Top-down방식(위에서 아래로 순차적으로 진행)의 개발유형 (참고: Bottom-up방식)

- 각 공정은 작업결과를 산출물(사양서)에 모아서 다음 공정으로 넘겨준다
  - 산출물: 문서, 도표, 프로그램 등
  - 사양서: (각 공정의 이름을 붙여서)
    - 요구사양서
    - 기본설계사양서
    - 상세설계사양서
    - 테스트사양서 등

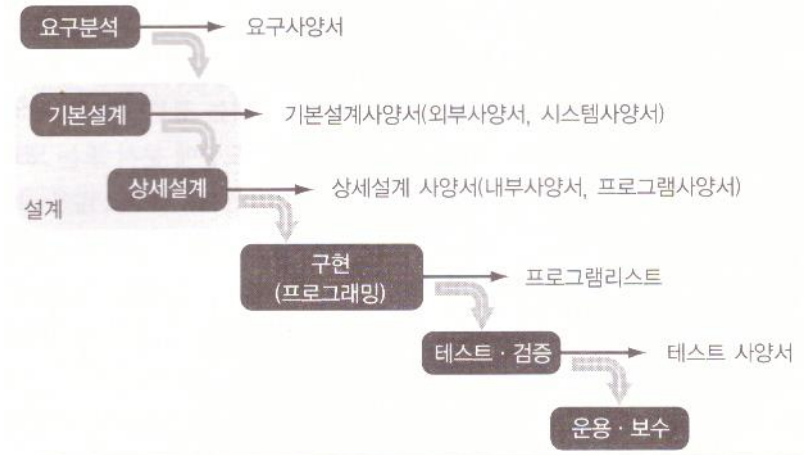


그림 2.1 폭포수모델

2.2 폭포수모델

2.2.1 폭포수모델의 개발순서

- 명확하게 구분된 각 공정마다 사양서 작성

- 리뷰(review): 각 공정에서 작성된 사양서의 재검토작업
  - 예) "언제 어느 공정까지 진행되었는지" 또는
    "예정대로 산출물이 완성되었는지" 등을 파악
  - 각 공정에서 리뷰가 수행되어 문제점을 추출 및 수정
  - 다음 공정에 문제를 넘겨주지 않도록 하기 위한 작업
  - 장점: 프로젝트관리가 수월하다는 것을 의미

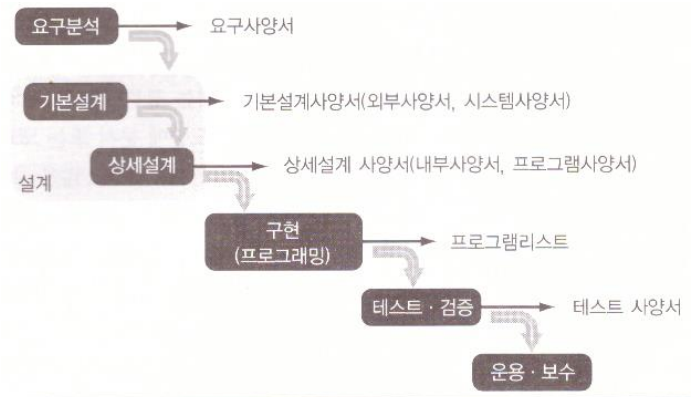


그림 2.1 폭포수모델

2.2 폭포수모델

2.2.1 폭포수모델의 개발순서

- 개발공정(그림 좌측)과 테스트(그림 우측)간 대응 관계
  - 테스트 및 검증공정: 단위테스트, 통합테스트, 시스템테스트, 인수테스트

- 프로그램에 대한 테스트 수행 순서
  - 1단계) 단위테스트
    - 모듈(작은 프로그램 뭉치) 등 기본단위별 동작
  - 2단계) 통합테스트
    - 여러 개의 모듈들을 통합하여, 전체적인 동작
  - 3단계) 시스템테스트
    - 완성된 프로그램 전체에 대한 동작 확인
  - 4단계) 인수테스트
    - 고객이 요구사양에 대한 적합성 확인

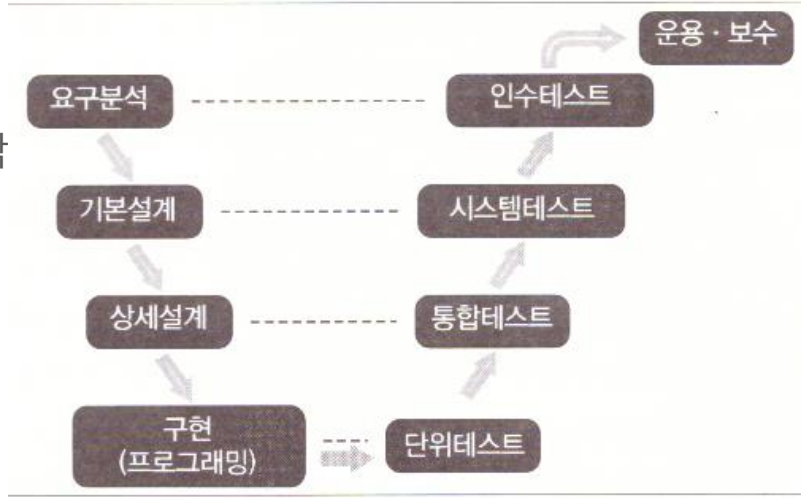


그림 2.2 | 폭포수모델의 테스트 • 검증 공정

2.2 폭포수모델

2.2.2 폭포수모델의 문제점

- 특징
  - 폭포수모델에서는 개발초기단계에서 요구사양을 결정 필요
  - 이미 유사한 소프트웨어 개발경험이 있거나, 요구사양이 명확한 대규모 소프트웨어를 개발하는 경우에 적합
  - 각 공정에서 사양서의 재검토작업(리뷰)에 의해 후속 공정에 문제를 넘겨주지 않도록 한다.

- 현실적 어려움
  - 문제점을 모두 찾아내면 좋겠지만, 그렇게 하는 것도 상당히 어려운 일이다.

2.2.2 폭포수모델의 문제점

- 문제점
  - 리뷰에 의해 문제점을 찾아내지 못하고, 후속 공정에서 문제점 발견된 경우
    - 재작업 발생으로 인한 개발 지연
      - 재작업: 문제점의 원인이 되는 공정으로 되돌아가서 문제점이 발견된 공정에 이르기까지의 제반작업들을 다시 수행
    - 수정에 필요한 비용 증가
      - 최초 단계에서 발생한 문제점의 발견이 늦어지면 늦어질수록
      - 재작업이 커지면 커질수록
- 대처 방안
  - 각 공정에서 충분히 리뷰를 수행
  - 문제점이 없는 사양서를 후속공정으로 넘겨주는 방법

- 현실적 어려움
  - 실제로는 다음과 같은 이유들 때문에 재작업을 완전하게 없애는 것이 어려움

2.2 폭포수모델

2.2.2 폭포수모델의 문제점

- 재작업 제거가 어려운 이유
  - 1) 요구사항분석의 어려움
    - 개발 초기단계에서 모든 요구사항들을 추출하여 명확하게 기술하는 것은 어려움
      - 애매모호한 내용이 포함되기 쉬움
      - 개발 의뢰처의 의도를 충분하게 이해하기 어려워 오해된 내용이 포함되기 쉬움
    - 향후 완성될 시스템의 모습을 상상하기 어려움
      - 필요한 기능을 나열해 보거나 사용하기 수월한 유저인터페이스를 생각
      - 실제로 이용해봐야지 비로소 고객이 기대하고 있던 기능과 다르다는 것을 알게되는 경우가 많음
  - 2) '개발초기단계에서 결정되는 요구사양'의 변경 요구 발생
    - 사양을 동결하고, 그 이후의 변경의뢰는 기본적으로 용인하지 않는 식의 방침을 취하기 어려움
    - 재작업을 줄일 수는 있지만, 고객의 요구사항을 충분히 반영시킨 시스템을 작성할 수 없게 된다

2.2.2 폭포수모델의 문제점

- 재작업 제거가 어려운 이유
  - 3) 구현의 어려움
    - 개발자가 구현방법을 상세하게 검토해 봄으로써 비로소 구현하기 곤란한 기능이었음을 발견하는 경우가 많음
    - 개발초기단계에서 모든 것을 통찰해서 개발할 기능들을 결정하기 어려움
  - 4) 개발 기간 장기화로 인한 환경 변화
    - 개발기간이 길어지면 환경변화에 의해 새로운 요구사항이 발생에 따른 변경 요구
    - 신기술의 등장에 의해, 구현방식 및 이용형태의 변경 요구

2.3 진화형 프로세스모델

개요

- 진화형(성장형 또는 발전형) 프로세스모델
  - 폭포수모델의 문제점들을 해결하기 위한 모델
  - 개발초기단계부터 구현가능한 프로그램을 부분적으로 만들어서 실제로 동작시켜서 기능을 확인하면서 개발을 진행하는 모델
  - 프로그램의 작성작업을 몇 번이고 반복해서 목표에 맞추는 방식
  - 반복방법에 따라서 다양한 모델이 있음

목차

- 2.3.1 진화형 개발 (프로토타이핑)
- 2.3.2 시험형 프로토타이핑 (나선형모델)
- 2.3.3 진화형 프로토타이핑 (반복형/점증형 개발프로세스모델)
- 2.3.4 애자일프로세스모델

2.3 진화형 프로세스모델

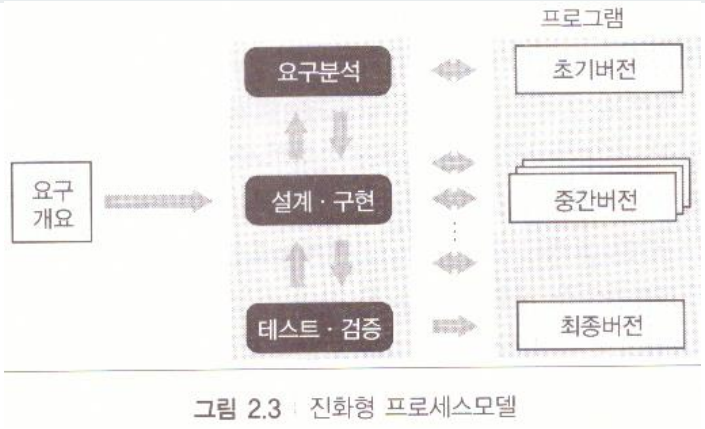


그림 2.3 진화형 프로세스모델

2.3.1 진화형 개발

- 현실적인 문제점
  - 1) 복수 개의 공정이 혼재 (예: 대규모 개발)
  - 2) 재작업 발생 (예: 폭포수모델)

- 진화형(성장형 또는 발전형) 프로세스모델
  - 문제점(1,2)을 해결하기 위한 소프트웨어개발기법
  - 요구사항의 변화에 적극 대응함으로써, 실제 소프트웨어개발에 적합
    - 문제점: 소프트웨어개발공정을 여러번 반복하게 되므로, 개발에 필요한 공수 및 비용의 산정이 어려움
  - 소프트웨어개발공정을 반복하여, 소프트웨어를 진화/성장
    - 초기단계(요구사항분석)에서 실행가능한 프로그램(첫번째 버전)을 작성
    - 실제로 동작시켜서 기능을 확인하고 사용자의 의견을 청취하여 개선을 목적으로 하는 프로그램(중간버전) 작성
    - 이를 반복하여, 최종적인 프로그램(최종버전)의 완성도를 높임

2.3 진화형 프로세스모델

2.3.1 진화형 개발

- 프로토타입(prototype)
  - 진화형 프로세스모델에서 도중에 작성하는 프로그램
- 프로토타이핑(Prototyping)
  - 프로토타입 작성이 수반되는 소프트웨어개발공정
  - 시험형 프로토타이핑: 실제 시스템 개발 전, 문제점 추출 목적으로 시험적으로 프로토타입을 작성하는 공정
    - 프로토타입의 시험결과를 토대로 요구사양 확정, 실제 시스템을 처음부터 다시 만들어감
  - 진화형 프로토타이핑: 프로토타입의 시험결과를 토대로 기능 변경 및 추가 확장하여 실용적인 프로그램으로 발전시키는 공정

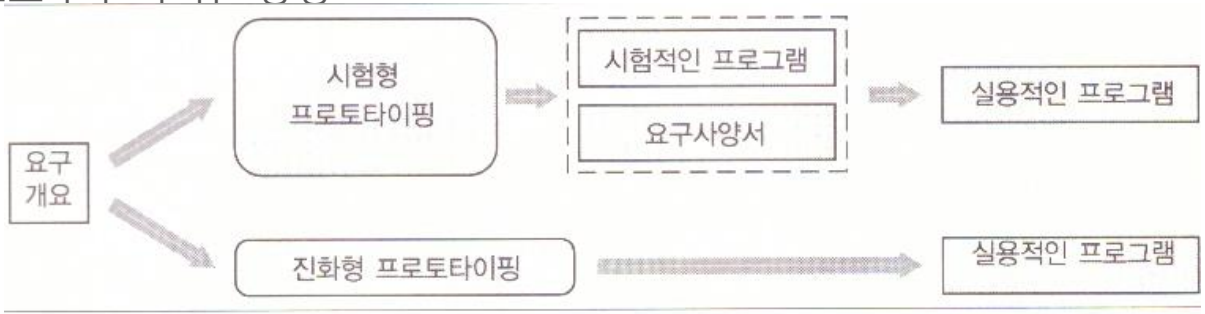


그림 2.4 | 프로토타이핑

2.3 진화형 프로세스모델

2.3.2 시험형 프로토타이핑

- 재작업을 대폭적으로 감소
  - 실제 시스템 개발 전, 요구사항분석 공정에서 조기에 문제점 추출 및 이를 요구사항에 반영
    - 일부 기능들에 대해서 분석, 설계, 구현, 평가를 수행
    - 프로토타입을 개발 및 평가하여 문제점 추출
      - 요구사항정의 및 설계에서 특히 중요한 부분이 계획대로 개발을 진행하더라도 치명적인 문제점이 발생하지 않음을 확인
      - 고객의 요구사항 중 애매모호한 부분을 우선 작성, 고객에게 프로토타입을 시험사용하도록 하여 기능 및 사용방법을 확인받음
  - 이후 실제 소프트웨어개발에 착수
- 폭포수모델의 장점을 이어받아서 문제점을 해결하는 모델
  - 그림: 시험형 프로토타이핑은 폭포수모델과 조합하여 사용하기 수월

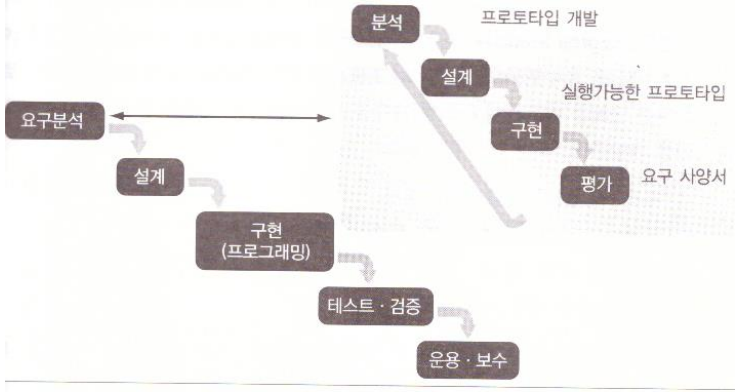


그림 2.5 시험형 프로토타이핑을 조합한 폭포수모델의 예

2.3 진화형 프로세스모델

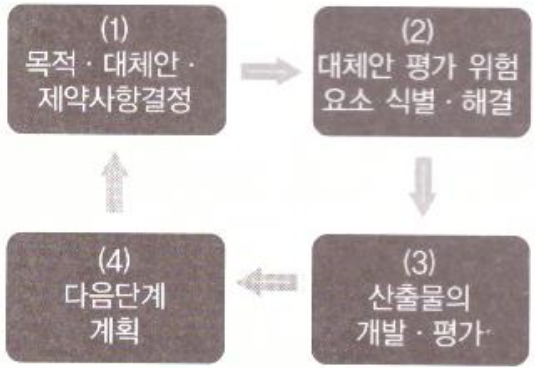
2.3.2 시험형 프로토타이핑

- 프로토타입 작성 목적
  - 보다 좋은 요구사양을 작성
  - 문제점을 추출하거나 사용법이 좋음을 확인
- 프로토타입 범위의 제약
  - 최종 시스템에서의 필수 기능이 생략되기도 함
  - 성능상의 문제점을 무시하기도 함
  - 성능이 좋지 않더라도 작성시간을 단축할 수 있는 언어 및 도구 사용되기도 함
  - 유저인터페이스의 요구사양 결정시, 화면기반의 목업(시제품) 도구가 사용되기도 함 (고객과 함께 평가)
- 높은 프로토타입 의존도의 간략한 요구사양서
  - 설계시에 프로토타입을 동작시켜서 사양을 확인하는데 많은 노력 필요
  - 프로토타입을 살펴보면 알수 있는 것도, 사양서에 명시적으로 기술 필요
  - 사전에 프로토타이핑의 목적을 명확하게 하여, 프로토타입으로 작성하는 기능에 대한 충분한 고려 필요
    - 프로토타입을 작성하더라도 추후에 프로토타입으로는 확인하지 못한 기능에서 재작업 발생

2.3 진화형 프로세스모델

2.3.2 시험형 프로토타이핑

- 나선형 모델(Spiral Model)
  - 앞서 살펴본 문제점을 해결하기 위하여 제안 (Barry W. Boehm)
    - 필수 기능 생략, 성능 무관, 실제 시스템과의 차이
  - 요구사항분석과 설계에서, 실패가능성 분석하여 위험요소 경감
    - 실패가능성: 개발이 실패하게 될 가능성이 있는 요인
      - 목표성능이 나오지 않을 위험성
      - 사용자가 사용하기 어려운 인터페이스가 될 위험성
  - 나선형 계단을 걸어올라가는 것처럼 단계적으로 소프트웨어개발
    - 다음단계에 대한 계획을 살펴보면서 설계 진행
    - 실패 가능성이 충분히 작아질때까지,
      아래 4단계의 반복을 계속 수행
      - 1) 목적/대체안/제약사항의 결정
      - 2) 대체안의 평가와 위험요소의 식별/해결
      - 3) 산출물의 개발/평가
      - 4) 다음 단계의 계획



(a) 기본개념

그림 2.6 | 나선형모델

2.3 진화형 프로세스모델

2.3.2 시험형 프로토타이핑

- 시험형 프로토타이핑을 폭포수모델과 조합한 경우 (좌측 그림)
  - 요구사항분석에서 프로토타입을 작성해서 한번에 요구사양을 결정
- 나선형 모델: 폭포수모델과 조합하는 경우 (우측 그림)
  - 위험요소의 분석결과에 따라, 기능 평가작업반복: '사전평가,시뮬레이션,프로토타입'에 의한 평가
  - 조금씩 줄여가는 위험요소가 충분히 줄어들게 되면, 이후 작업 이행 (예: 상세설계로부터 구현, 테스트 등)

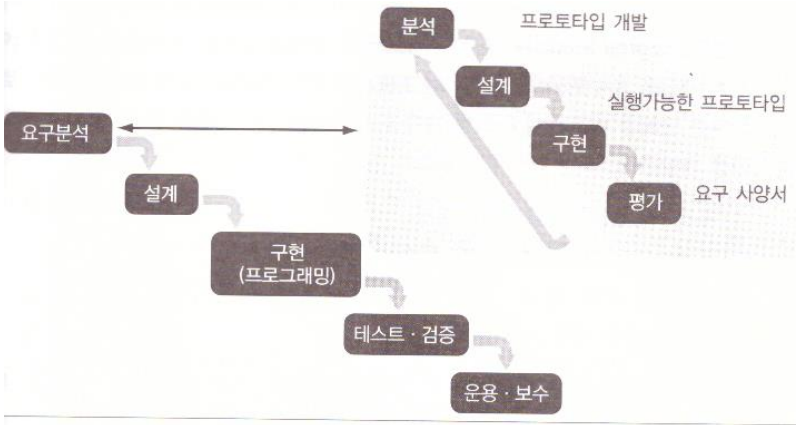
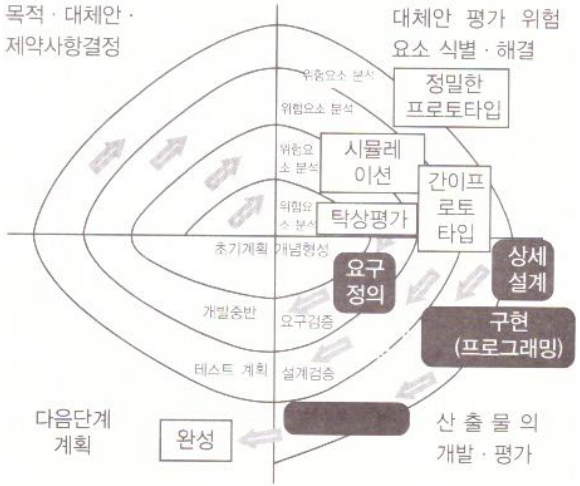


그림 2.5 | 시험형 프로토타이핑을 조합한 폭포수모델의 예



(b) 폭포수 모델에 적용

그림 2.6 | 나선형모델

2.3.3 진화형 프로토타이핑

- 진화형과 시험형 프로토타이핑 비교
  - 유사점: 실제로 동작하는 시스템을 사용자에게 최대한 빨리 제공한다라는 점
  - 차이점
    - 시험형 프로토타이핑
      - 프로토타입의 역할이 끝나면 폐기
      - 프로토타입작성에 이용할 수 있는 언어 및 도구, 우선적으로 만들어야하는 기능 등이 서로 다름
    - 진화형 프로토타이핑
      - 최종적인 시스템과 같은 언어 사용 필요
        - 점진적으로 수정해가는 프로토타입이 완성되면, 최종 완성 시스템으로서 이용
        - 최종 시스템으로서 품질을 갖춘 프로그램(프로토타입) 개발 필요
      - 개발순서
        - 우선 개발: 기능 명확, 개발효과 확실, 기술적으로 불확정적인 부분이 적은 것
        - 이후 서서히 개발: 새롭게 도전적인 기능 추가
      - 재작업 발생 방지: 모든 사양과 성능을 확인하면서 개발 진행
        - 개발의 최종단계에서 설계의 변경으로 인한 대폭적인 재작업 발생 방지

2.3.3 진화형 프로토타이핑

- 진화형 프로토타이핑(반복형 개발)의 분류
  - 분석, 설계, 구현, 테스트와 같은 공정을 반복
  - 프로토타입의 개설방법의 차이에 따른 분류
    - 1) 반복적 개발프로세스모델
    - 2) 점증적 개발프로세스모델

2.3 진화형 프로세스모델

2.3.3 진화형 프로토타이핑

- 1) 반복적 개발프로세스모델
  - 요구사양을 초기에 명확하게 지정하는 것이 어려운 일부 (첨단)시스템 개발에 사용
    - 예) 인공지능시스템, 유저인터페이스시스템
  - 프로토타입의 개발과 평가를 반복해서 만족할 수 있는 시스템이 되면 완료하는 모델
    - 조금씩 구현해서 사용자에게 제공 및 의견 청취 후 프로토타입을 반복하여 수정
    - 사용자의 요구사항에 걸맞는 시스템을 제공 가능

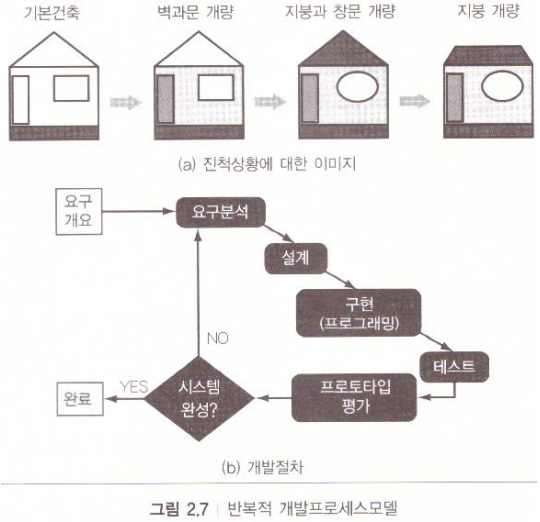


그림 2.7 | 반복적 개발프로세스모델

2.3 진화형 프로세스모델

2.3.3 진화형 프로토타이핑

- 1) 반복적 개발프로세스모델
  - 시스템전체를 한번에 개발하고 개발종료 후에 문제가 발생하면 **수정 작업 반복**
    - 장점:
      - 개발초기에는 애매모호했었던 사용자의 요구사항을 명확하게 함
      - 새로 발생한 요구사항에 대해서도 유연하게 대처
      - 시스템의 완성도 향상
    - 단점:
      - (수정 작업의 반복으로 인하여)
        - 프로토타입의 최종적인 구조를 파악하는 것이 어려움
        - 개발한 기능 등에 관한 문서화 불충분
        - 개발 후의 유지보수작업의 어려움
      - (폭포수모델 대비)
        - 고객은 시스템이 완성될 때까지 어떤 시스템이 되는지 알기 어려움
        - 시스템개발 전, 개발계약 체결이 어려운 경향

2.3 진화형 프로세스모델

2.3.3 진화형 프로토타이핑

- 2) 점증적 개발프로세스모델
  - 점증적(incremental)이란, 조금씩 증가시킨다라는 의미
  - 시스템전체를 독립성이 높은 서브시스템들로 분할하고 각 서브시스템들을 개발하는 모델.
    - 최초 개발 완료된 서브시스템과의 인터페이스를 유지하면서, 다음번 서브시스템 개발 작업 반복
  - 반복적 개발프로세스모델의 문제점들을 해결하기 위하여,
    - 아래 공정 절차 반복: 반복할때마다 '계획과 문서' 기록되어, 관리 측면의 문제점들이 해소
    - 1) 최종 구조 파악 용이하도록, 기본구조 설계
    - 2) 증가시키는 부분을 명확히 표기
    - 3) 해당부분에 대한 구현과 검증 수행
    - 4) 이미 개발완료된 부분과 통합
    - 5) 전체적인 검증 수행

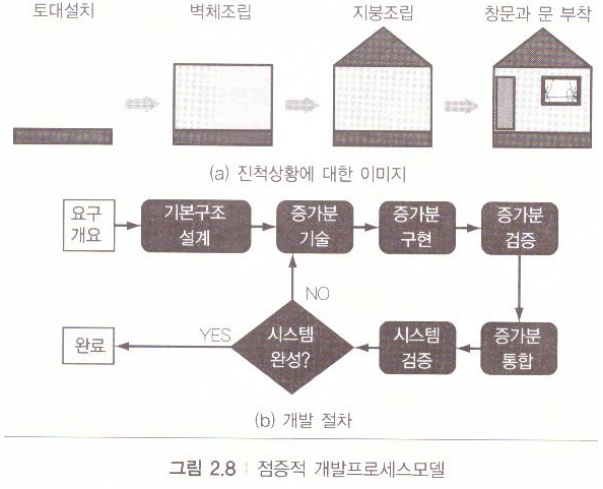


그림 2.8 | 점증적 개발프로세스모델

2.3.4 애자일프로세스(경쾌한 프로세스 또는 경량프로세스)모델

- 애자일(agile)이란, "재빠르고 수월하게 움직일 수 있다."라는 의미
- 변화에 민첩하게 대응할 목적
  - 개발대상을 여러 개의 작은 '기능'들로 분할
  - 각각의 작은 기능을 단기간내에 개발하는 작업 반복
  - 각 기능의 개발에서
    - 요구사항분석, 설계, 구현, 테스트, 문서화 등과 같은 일반적인 소프트웨어개발공정을 한단계씩 수행
    - 지금까지 개발완료된 성과물에 새로운 기능을 한가지씩 추가해 감
- 애자일프로세스개발
  - 고객의 만족도 향상이 최우선
    - 소프트웨어를 부분적으로도 가능한한 조기단계에서 지속적으로 사용자에게 인도
    - 동작하는 소프트웨어를 가능한한 짧은 시간간격으로 반복하여 인도
  - 고객과 개발자 및 개발자조직 내부에서의 대화 중시

2.3 진화형 프로세스모델

2.3.4 애자일프로세스모델

- 애자일소프트웨어개발기법
  - eXtreme Programming(XP)
    - 개발팀의 실천 사항 규정
      - "요구사항은 항상 변화한다"
      - "고객은 동작하는 시스템을 가능한한 하루빨리 인수하고 싶어한다."
    - 단기간(1~2주간 정도)마다 실행가능한 코드를 제공하는 것을 목표
      - 그 사이에 설계, 구현, 테스트 수행
      - 설계를 가능한한 단순화
      - 초기에 테스트를 작성하여
        이를 통과할 수 있도록 구현작업을 수행

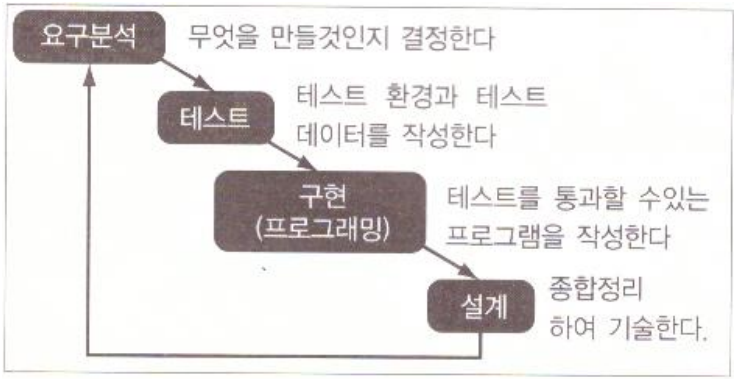


그림 2.9 XP(eXtreme Programming)

2.3.4 애자일프로세스모델

- 짝 프로그래밍(pair programming)
  - XP의 특징들 중 하나
  - 한 팀으로 한 개의 프로그램을 개발
    - 드라이버: 실제로 키보드를 사용하여 코드 작성
    - 네비게이터: 드라이버의 옆에서 코드 체크 및 오류 지적
  - 좋은 점
    - 2명이 작업하기 때문에 (아래는 가능성임)
      - 작업효율 향상
      - 프로그램의 품질 향상 가능
      - 프로그램에 대한 지식 공유

그림 2.10 Pair Programming

2.4 프로세스프로그래밍

개요

- 프로세스프로그래밍(Process Programming)
  - 소프트웨어프로세스에서 소프트웨어개발공정에서의 수행 작업을 명확하게 정의할 때, 이러한 정의를 '프로세스기술언어'를 사용하여 기술하는 것을 말함
  - 1987년 논문에서 제안된 개념 (Leon Osterweil, "software processes are software too.(소프트웨어프로세스도 소프트웨어다.)")
  - 모든 프로젝트에 적절하게 적용가능한 통일된 소프트웨어프로세스는 존재하기 어려움
    - 문화, 기술자의 기술력, 개발전략 등이 각각 서로 다르기 때문에
  - 대안으로서,
    - 1) 기본적인 프로세스를 정의
    - 2) 각 프로젝트의 특징에 따라, 각 프로젝트에 적합한 형태로 변경
    - 3) 소프트웨어개발환경을 특정한 프로젝트에 적합한 형태로 조정 필요
    - 4) 소프트웨어의 개발공정을 절차적인 프로그램으로서 기술
    - 5) 해당 프로그램의 실행을 통해, 소프트웨어 개발 체계 생성

2.4 프로세스프로그래밍

프로세스프로그래밍

- 프로세스프로그래밍의 목표
  - 1) 소프트웨어프로세스를 엄밀하게 기술함
    - 해당 소프트웨어프로세스의 성질 및 성능 평가
  - 2) 기술된 프로그램을 컴퓨터에서 실행
    - 개발프로세스 자동화
  - 3) 소프트웨어프로세스의 분석, 평가, 개선, 자동화를 진행
    - 소프트웨어개발의 생산성 향상

- 프로세서프로그래밍에 대한 연구들
  - 1) 인간의 행동이 주요요소가 되는 개발과정을 관찰, 분석하여 개선방법을 제시하는 연구
  - 2) 프로세스를 형식적으로 기술하는 언어와 시스템을 연구개발하여, 실제로 프로세스를 기술하는 연구
  - 3) 프로세스라는 개념을 핵심으로 하는 소프트웨어개발환경을 개발하는 연구

프로세스프로그래밍

- 프로세스프로그래밍의 한계
  - 실용적인 기술로 발전하는데 어려움 있음

  - 프로그램을 실행하여 실제로 동작하는 것은 인간이기 때문에
    - 프로그램에 기술된 것 그대로는 되지 않음
    - 프로그램으로 기술할 수 있는 범위의 제약 (포괄성 측면)
      - 소프트웨어개발전체에서 살펴보면 일부분에 지나지 않음
    - 이 외, 여러가지 문제점이 내포되어 있어 어려움

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 1. 소프트웨어개발에 있어서 각 작업공정을 무엇이라고 부르는가? ( )
  (1) 요구분석 (2) 설계 (3) 프로세스 (4) 단위시험 (5) 정답없음

- 2. 소프트웨어 공정으로 적합하지 않은 것은 무엇인가? ( )
  (1) 회의 (2) 요구사항분석 (3) 설계 (4) 구현 (5) 테스트/검증
  (6) 운용/유지보수

- 3. 설계 공정에 해당하는 것을 모두 고르시오. ( )
  (1) 기본설계 (2) 중급설계 (3) 고급설계 (4) 상세설계 (5) 간략설계

- 4. 테스트/검증 공정에서 수행되는 테스트가 아닌 것은? ( )
  (1) 단위테스트 (2) 통합테스트 (3) 시스템테스트 (4) 인수테스트
  (5) 정답없음

- 5. 소프트웨어프로세스모델에 해당되지 않는 것은? ( )
  (1) 폭포수 (2) 진화형 (3) 시험형 (4) 애자일 (5) 페어플레이

객관식 문제 (답안지)

- 1. (3) → 프로세스
- 2. (1) → 기획
- 3. (1,4)
  → 기본설계, 상세설계
- 4. (5) → 정답없음
- 5. (5) → 모델아님

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 6. 폭포수모델에서는 개발후기단계에서 요구사항을 결정하는 것이 필요하다. ( )
  (1) 맞다. (2) 틀리다.

- 7. 폭포수모델의 문제점이 아닌 것을 고르시오. ( )
  (1) 재작업 발생 (2) 개발지연 (3) 수정 비용증가 (4) 관리 어려움

- 8. 진화형 프로세스모델에서 해결하려고 하는 '폭포수모델의 문제점'을 모두 고르시오. ( )
  (1) 복수 개의 공정 혼재 (2) 재작업 발생 (3) 리뷰 불충분 (4) 관리 소홀

- 9. 시험형 프로토타이핑의 장점으로 알맞은 것은? ( )
  (1) 재작업 증가 (2) 재작업 감소 (3) 문제점 증대 (4) 정답없음

객관식 문제 (답안지)

- 6. (2) → 개발초기단계
- 7. (4) → 관리 수월
- 8. (1,2) → 복수 개의 공정 혼재, 재작업 발생
- 9. (2) → 재작업 감소

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 1. 소프트웨어개발을 어떻게 진행할 것인가를 규정한 것을 ( ) 이라고 한다.
- 2. 소프트웨어개발에 있어서 각 작업공정은 ( )라고 부른다.
- 3. 소프트웨어개발에 있어서 각 작업공정은 프로세스라고 부르며, 개발공정전체를 ( )라고 부른다.
- 4. 소프트웨어프로세스를 규정하면, 각 프로세스의 작업과 그다음 작업으로 연결하기 위한 ( )이 결정된다.
- 5. ( )은 어떤 소프트웨어를 작성해야되는가를 명확하게 하여 요구사양을 결정하는 공정이다.

주관식 문제 (답안지)

- 1. 소프트웨어개발공정
- 2. 개발프로세스 (또는 프로세스)
- 3. 소프트웨어프로세스
- 4. 산출물
- 5. 요구사항 분석

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 6. (　　)은 프로그래밍언어를 사용하여 요구사양 및 설계사양에 대응하는 프로그램 기술(코딩) 공정이다.
- 7. (　　)은 프로그램이 올바르게 동작하는지를 검사하기 위하여 코딩완료된 프로그램을 테스트하는 공정이다.
- 8. (　　)는 소프트웨어를 운용하고 유지관리하는 공정이다.
- 9. (　　)는 각 공정에서 작성된 사양서의 재검토 작업을 말한다.
- 10. 진화형 프로세스모델의 문제점은 (　　)이다.

주관식 문제 (답안지)

- 6. 구현
- 7. 테스트/검증
- 8. 운용/유지보수
- 9. 리뷰 (review)
- 10. 소프트웨어개발공정을 여러번 반복하게 되므로, 개발에 필요한 공수 및 비용의 산정이 어려움

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 11. (　　)은 진화형 프로세스모델에서 도중에 작성하는 프로그램이다.
- 12. (　　)은 프로토타입 작성이 수반되는 소프트웨어개발공정이다.
- 13. (　　)은 실제 시스템 개발 전, 문제점 추출 목적으로 시험적으로 프로토타입을 작성하는 공정
- 14. (　　)은 프로토타입의 시험결과를 토대로 기능 변경 및 추가 확장하여 실용적인 프로그램으로 발전시키는 공정
- 15. (　　)은 요구사항분석과 설계에서 실패가능성을 분석하여 위험요소를 경감하는 방안 중 하나이다.

주관식 문제 (답안지)

- 11. 프로토타입(prototype)
- 12. 프로토타이핑(Prototyping)
- 13. 시험형 프로토타이핑
- 14. 진화형 프로토타이핑
- 15. 나선형모델(Spiral model)

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 16. ( )과 ( ) 프로토타이핑의 유사점은 실제로 동작하는 시스템을 사용자에게 최대한 빨리 제공한다라는 점이다.
- 17. ( ) 개발프로세스모델의 장점은 사용자의 요구사항을 명확히 하여, 새로 발생한 요구사항에 대해서도 유연하게 대처할 수 있지만, 고객은 시스템이 완성될 때까지 어떤 시스템이 되는지 알기 어려운 측면이 있다.
- 18. ( ) 개발프로세스모델은 ( ) 개발프로세스모델의 문제점을 해결하기 위하여, 최종 구조 파악이 용이하도록 기본구조 설계를 하는 등의 공정 절차를 반복하며, 반복할때마다 '계획과 문서'가 기록되어, 관리 측면의 문제점들이 해소될 수 있다.

주관식 문제 (답안지)

- 16. 진화형(또는 반복형), 시험형
- 17. 반복적
- 18. 점증적, 반복적

Talktime

Any questions?