

소프트웨어공학

한성대학교 컴퓨터공학부



INDEX



소프트웨어의 위기와 소프트웨어 공학의 필요성
소프트웨어 개발 모델

소프트웨어 위기의 등장

- 소프트웨어 크라이시스(Crisis)

- 1968년 NATO 소프트웨어공학 학회에서 처음 등장
- 컴퓨터에 의한 계산 용량과 문제의 복잡도가 급증하면서, 새로운 소프트웨어 개발 방법의 필요성을 인식
- 본질적으로는 정확하고 이해할 수 있고, 검증 가능한 컴퓨터 프로그램을 작성하는 것이 얼마나 어려운가를 의미

- 소프트웨어 위기의 원인

- 소프트웨어 규모의 대형화 및 복잡도 증가에 따른 개발 비용 증대
- 소프트웨어 유지보수의 어려움과 개발 정체 현상 발생
- 소프트웨어 개발 프로젝트 기간 및 소요 예산에 대한 정확한 예측의 어려움
- 신기술에 대한 교육 및 훈련의 부족
- 사용자의 소프트웨어에 대한 기대치 증가
- 소프트웨어에 대한 사용자 요구사항의 빈번한 변경 및 반영

☞ 예산 초과, 일정 지연, 낮은 품질 또는 비효율적인 소프트웨어 개발, 사용자 요구사항 불만족, 산출물 관리의 어려움과 같은 증상

소프트웨어 위기의 등장

- 이런 증상이 누적되면 결국 소프트웨어 개발 실패, 사용자에게 서비스하지 못하는 현상 발생
- 이 소프트웨어 위기를 해결하기 위한 다양한 공학적 접근 방법과 기법 개발
- 하지만 소프트웨어 공학적 기술의 적용이 프로젝트의 성공과 실패를 명확히 결정할 수는 없을 것
- 1980년대 소프트웨어 개발 프로젝트는 전체 프로젝트 중에서 15%만 성공, 30%는 사용자에게 배포되지 못함

소프트웨어 개발 프로젝트 통계

■ 소프트웨어 개발 프로젝트의 성공과 실패율

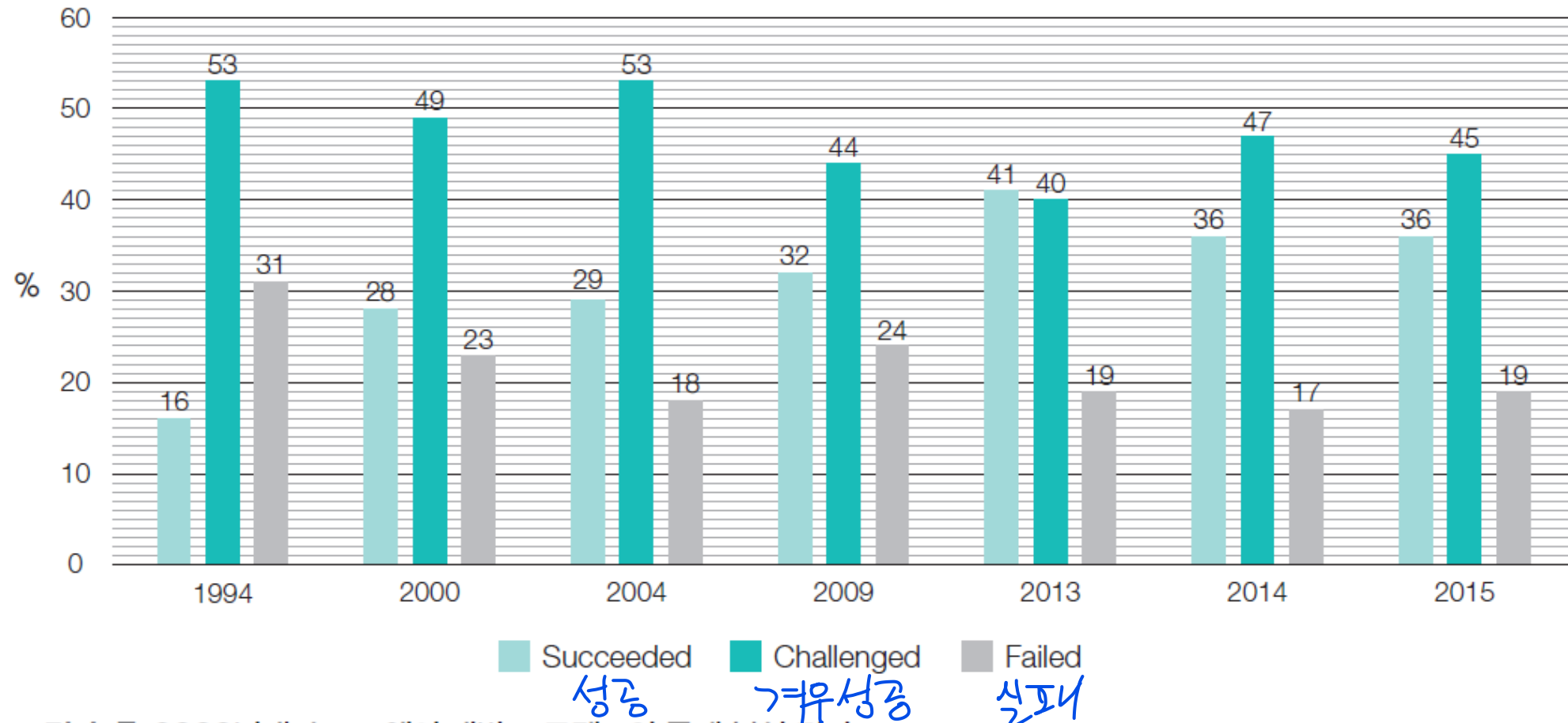



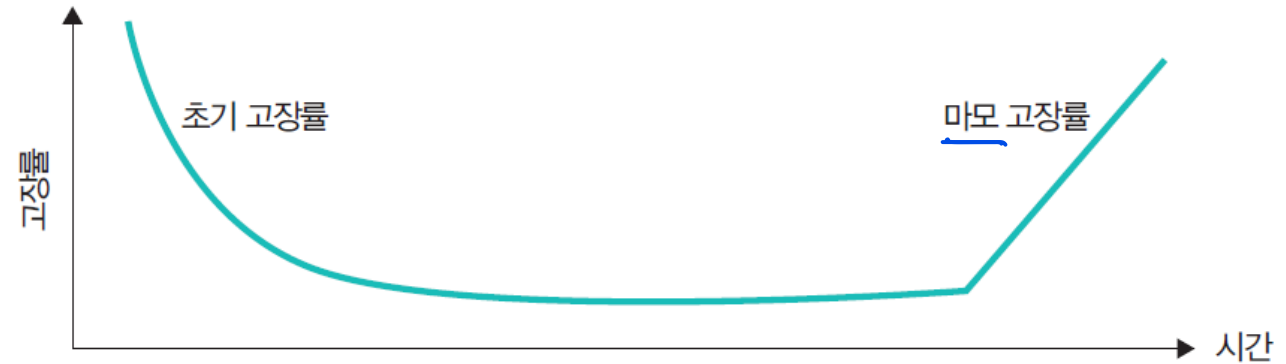
그림 1-7 2000년대 소프트웨어 개발 프로젝트의 통계 분석 결과(출처: Standish Group, CHAOS Report, 2015, USA)

소프트웨어 특성 정보

- 소프트웨어 개발 비용
 - 분석 및 설계 40%, 코드 개발 20%, 테스트 40%
- 소프트웨어 개발 및 유지보수
 - 개발 비용 33%, 운영 및 유지 비율 67%
- 소프트웨어 개발 시 오류의 근원 
 - 문서화 및 기타 오류 35%, 코드 작성 오류 30%, 로직 설계의 오류 20%, 기능의 잘못된 이해 15%

소프트웨어 고장 그래프

- 하드웨어 고장 그래프



- 소프트웨어 고장 그래프

그림 1-11 하드웨어 수명주기에 따른 고장률 그래프

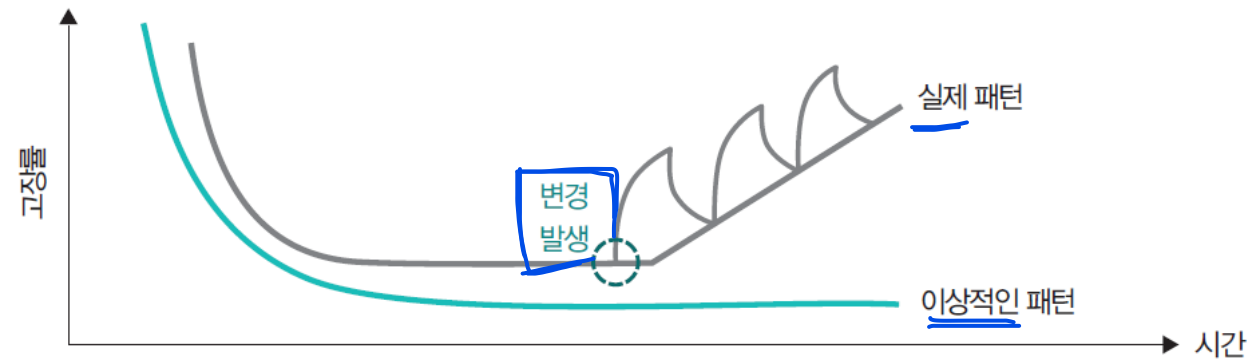
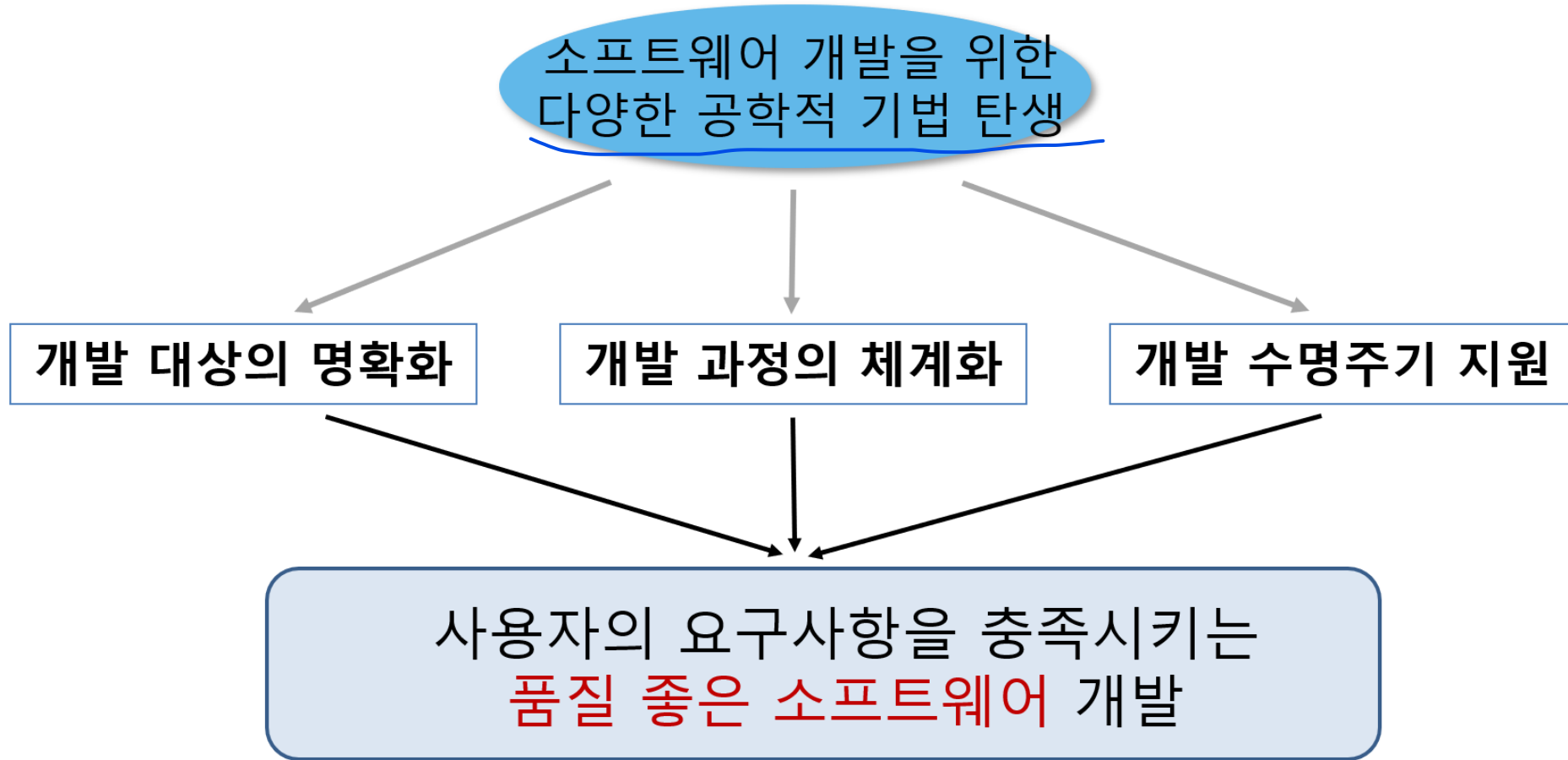


그림 1-12 소프트웨어 수명주기에 따른 고장률 그래프

소프트웨어 공학의 목표



소프트웨어 개발 모델

- 소프트웨어 라이프 사이클/소프트웨어 수명(생명) 주기

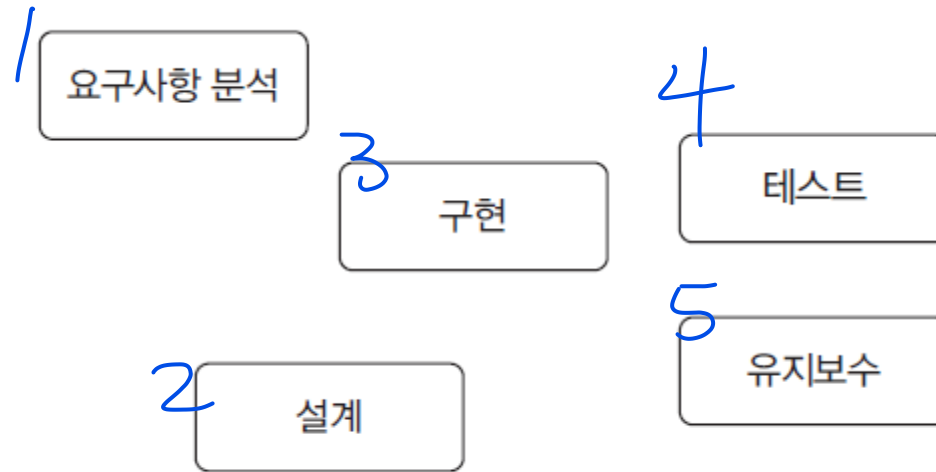


그림 1-1 소프트웨어 생명주기

소프트웨어 개발 단계

- 요구사항 분석: 개발하고자 하는 소프트웨어에 대한 요구사항을 고객으로부터 수집하고 분석하며 명세하는 단계
- 설계: 고객의 요구사항을 만족하기 위한 여러 해결책을 제시하고 이 중에서 가장 최적화된 해결책을 선정하는 단계
- 구현: 고객의 요구사항을 실제 서비스의 형태로 제공할 수 있도록 프로그래밍 언어를 사용하여 개발하는 단계
- 테스트: 개발된 프로그램이 고객의 요구대로 동작이 되는지를 시험하는 단계

소프트웨어 개발 모델

Code-and-fix 모델

테스트와 디버깅의 차이를 주지 않는다

우연히 발견된 오류를 수정하는 디버깅에 중점을 두며
프로그램의 오류를 찾기 위한 별도의 기법을 적용하지는 않는다.

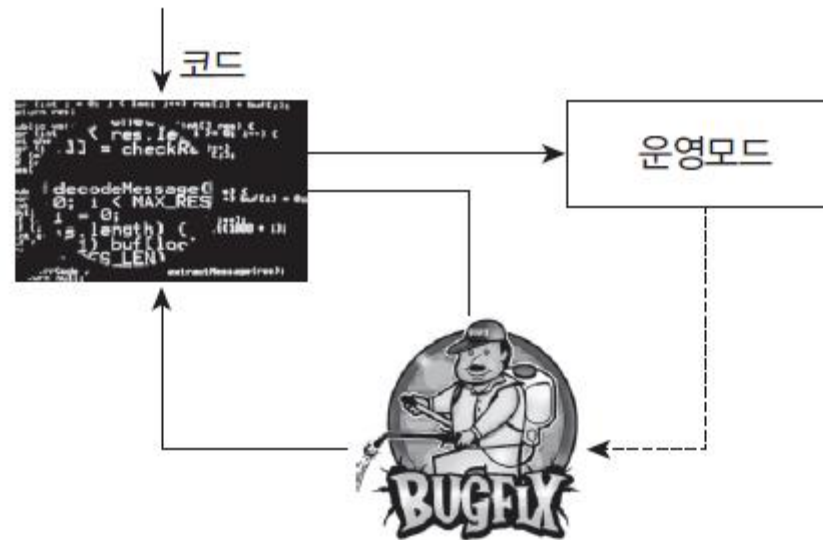
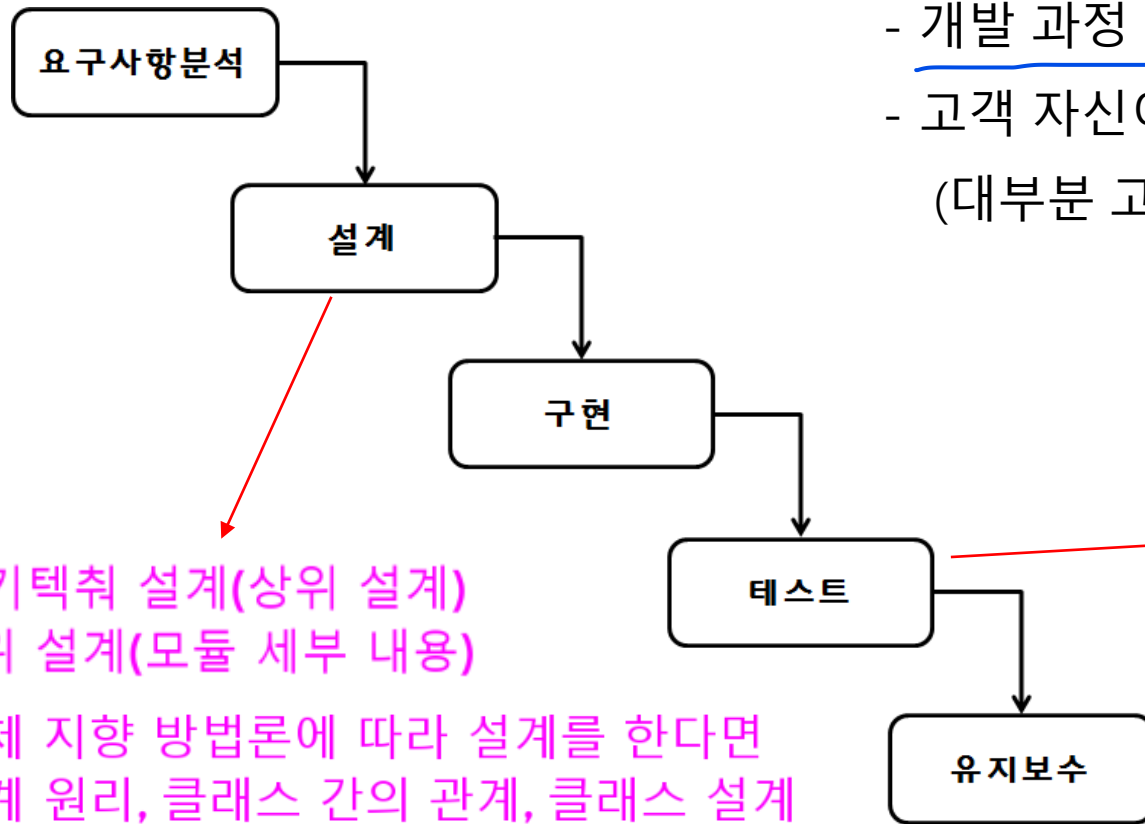


그림 1-2 Code-and-Fix 모델

요구사항 분석 단계와 설계 단계가 없다. 따라서 무엇을 개발할 지가 아주 명확하거나 개발될 소프트웨어가 매우 소규모이면서 매우 간단한 경우에 사용된다. 또한 요구사항에 대한 명세를 따로 작성하지 않기 때문에 매우 짧은 기간 동안만 사용되는 경우에 이용

폭포수(waterfall) 모델

- 가장 전통적인 모델
- 다른 모델의 기반
- 선형적



아키텍처 설계(상위 설계)
하위 설계(모듈 세부 내용)

객체 지향 방법론에 따라 설계를 한다면

- 설계 원리, 클래스 간의 관계, 클래스 설계 원칙 등 고려

폭포수 모델은

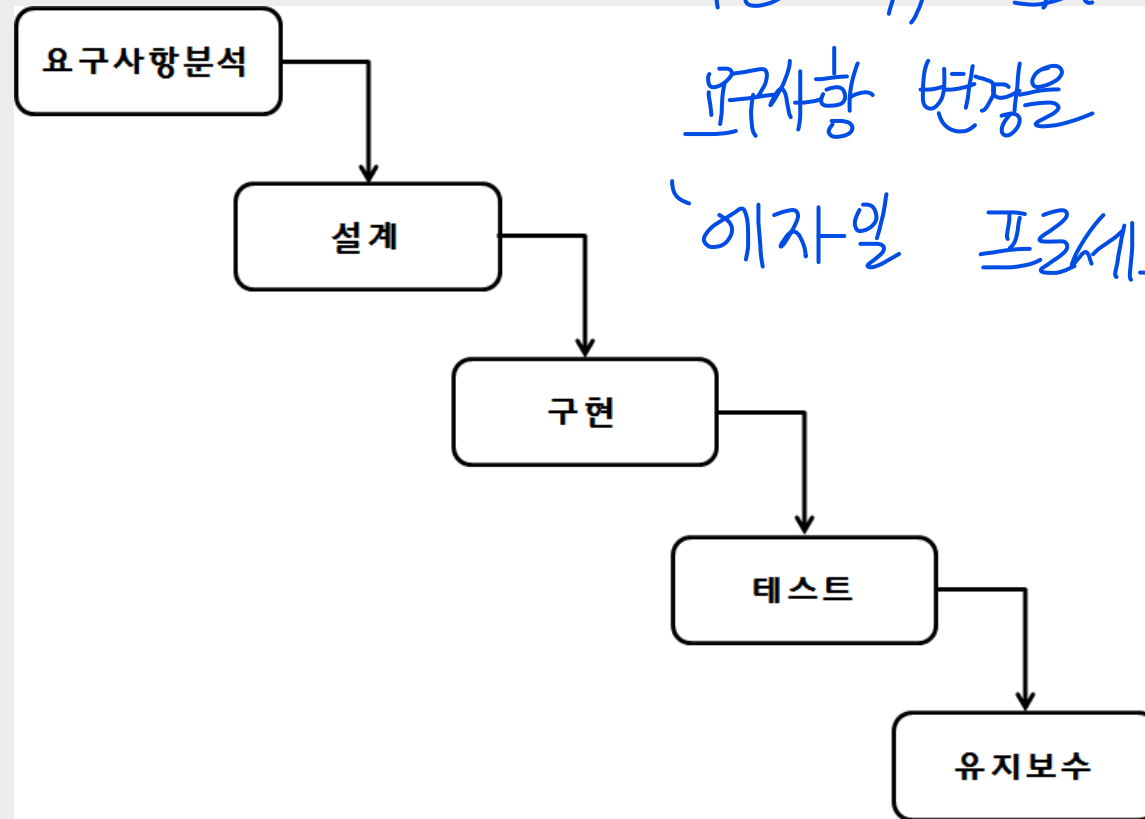
- 요구사항 단계에서 고객의 모든 요구사항이 완벽하게 수집되어야 한다.
- 하지만 고객의 요구사항을 불완전하게 반영하거나 틀리게 정의 하는 경우도 존재(초기 개발 단계)
- 개발 과정 중에 고객이 요구사항을 변경하는 경우도 존재
- 고객 자신이 원하는 서비스를 고객 자신도 모르는 경우도 존재
(대부분 고객의 피드백은 실제 동작하는 소프트웨어의 존재 필요)

이와 같이 폭포수 모델은 요구사항에 변화에 대해 기민하게 대처하지 못함

개발자 또는 사용자 시각에 따른 분류
사용되는 목적에 따른 분류
프로그램의 실행 요구 여부에 따른 분류
품질 특성에 따른 분류
소프트웨어 개발 단계에 따른 분류

폭포수(waterfall) 모델

- 요구사항 단계에서 고객의 모든 요구사항이 완벽하게 수집되어야 하며 일단 요구사항이 수집되고 분석되어 명세가 완료되었으면 더 이상 요구사항은 변경 되지 않아야 한다



비현실적, 요구사항은 바뀔 수도 있다.

요구사항 변경을 압축해서
'이메일 프로세스' 등장

폭포수(waterfall) 모델

- 폭포수 모델은 레퍼런스 없이 최초로 개발하는 경우나 난이도가 낮고 요구사항이 명확한 프로젝트를 수행하는 경우에 적합
- 장점
 - 관리가 용이하다.
 - 체계적으로 문서화할 수 있다.
 - 요구사항의 변화가 적은 프로젝트에 적합하다.
- 단점
 - 요구사항의 변화에 기민하게 대처하지 못함
 - 폭포가 위에서 아래로 흐르듯이 각 단계는 앞 단계가 완료되어야 수행할 수 있다.
 - 각 단계마다 작성된 결과물이 완벽한 수준으로 작성되어야 다음 단계에 오류를 넘겨주지 않는다.
 - 사용자가 중간에 가시적인 결과를 볼 수 없어 답답해할 수 있다.

애자일 프로세스

- 반복적이고 점진적인 개발 방법 (IID, Iterative and Incremental Development)에 기반

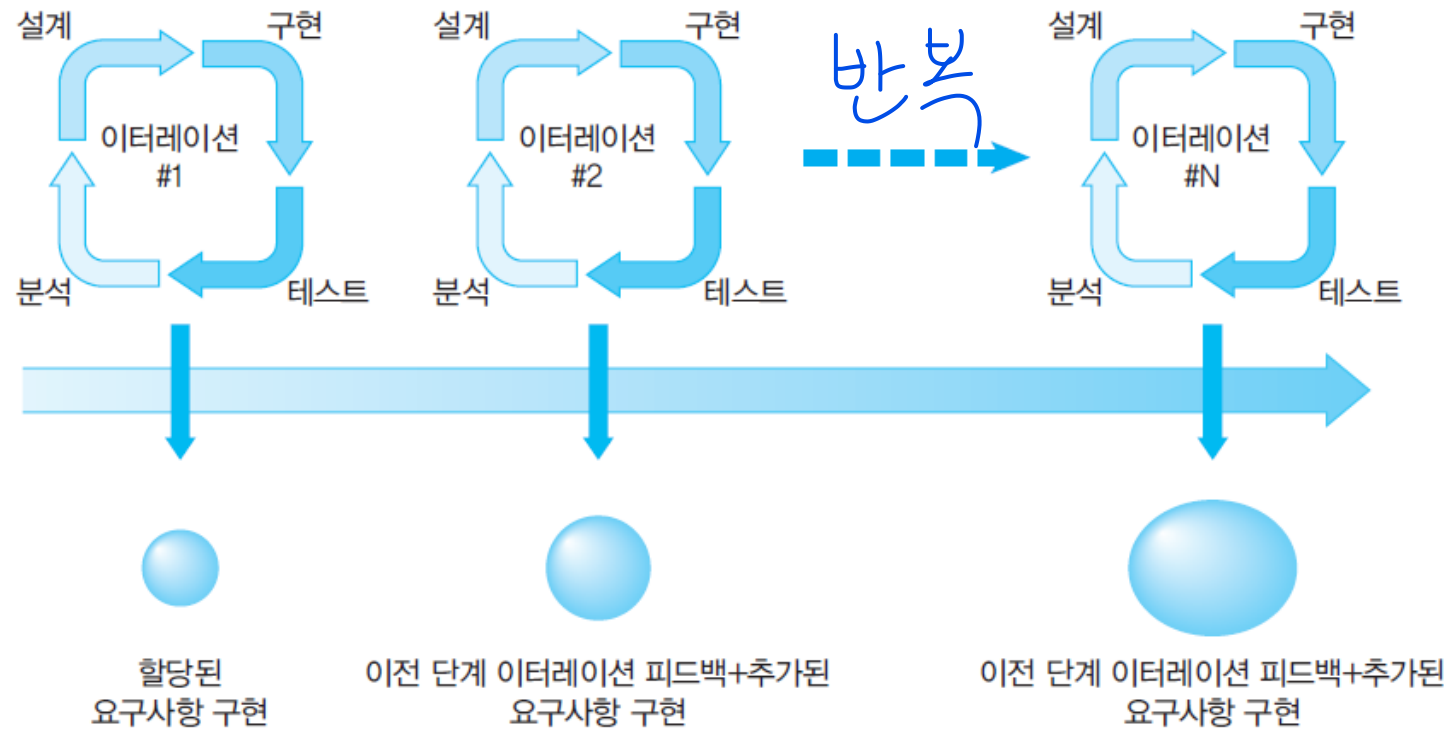


그림 1-6 애자일 프로세스

애자일 프로세스

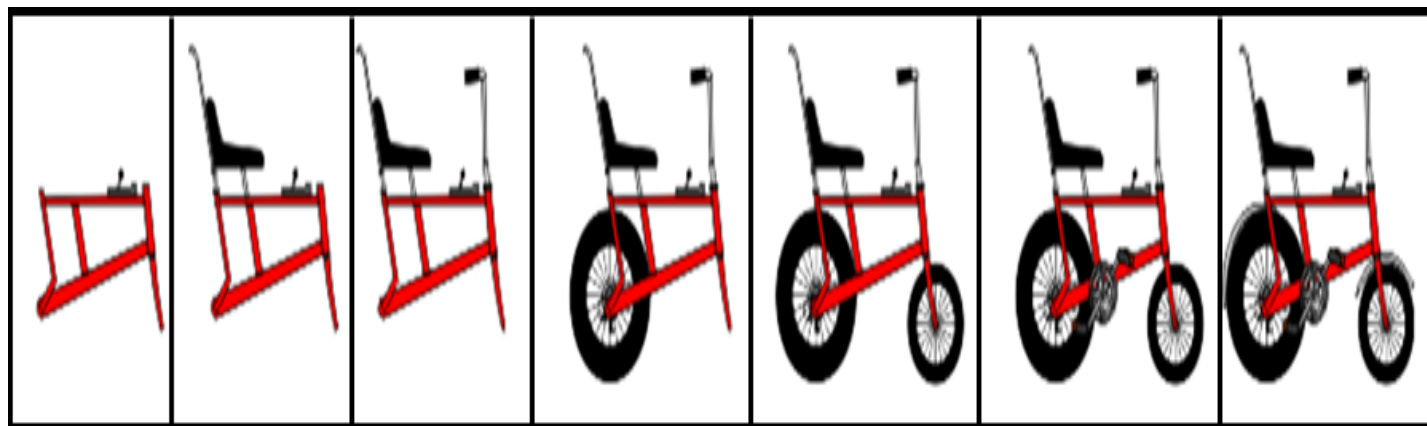
- 소프트웨어 개발 주기를 여러 개의 반복(Iteration)주기로 구분.
- 각 반복주기에서는 요구분석, 설계, 구현 및 테스트와 같은 활동들로 구성된 소규모 프로젝트
- 각 반복주기가 종료되면 부분적으로 완성된 시스템이 산출된다.
- 한 단계의 이터레이션 결과를 보고 다음 단계의 이터레이션에서 요구사항을 추가하거나 수정하는 등의 고객의 피드백을 반영한다.
- 소프트웨어 개발 주기를 구성하는 각 반복 주기는 보통 1주에서 4주이며 반복마다 새로운 요구사항이 추가되어 개발(Incremental development)

애자일 개발



그림 1-7 애자일 방법에서의 반복적이면서 점진적 개발 방식

전통적 개발



애자일 선언

- 2001년 켄트 벡, 워드 커닝햄, 알리스테어 코번, 마틴 파울러 등 생각이 비슷한 사람이 모여 폭포수 모델러로 대표되는 고전적 개발 방식으로 독립을 선언하듯 소위 애자일 선언을 발표

애자일이 추구하는 4가지 가치

- 프로세스나 도구보다는 개개인과 그들 상호 협력에 보다 많은 가치를 둔다.
- 포괄적인 문서화보다는 동작하는 소프트웨어에 보다 많은 가치를 둔다.
- 계약 협상보다는 고객과의 협력에 보다 많은 가치를 둔다.
- 계획에 따르기보다는 변화에 대한 대응에 보다 많은 가치를 둔다.

전통적 개발 방식은 제품을 만들기 위한 작업 단계들에 중점을 두고 있는 반면 애자일 방법은 제품에 의해 고객에게 전달되는 가치에 중점을 두고 있음

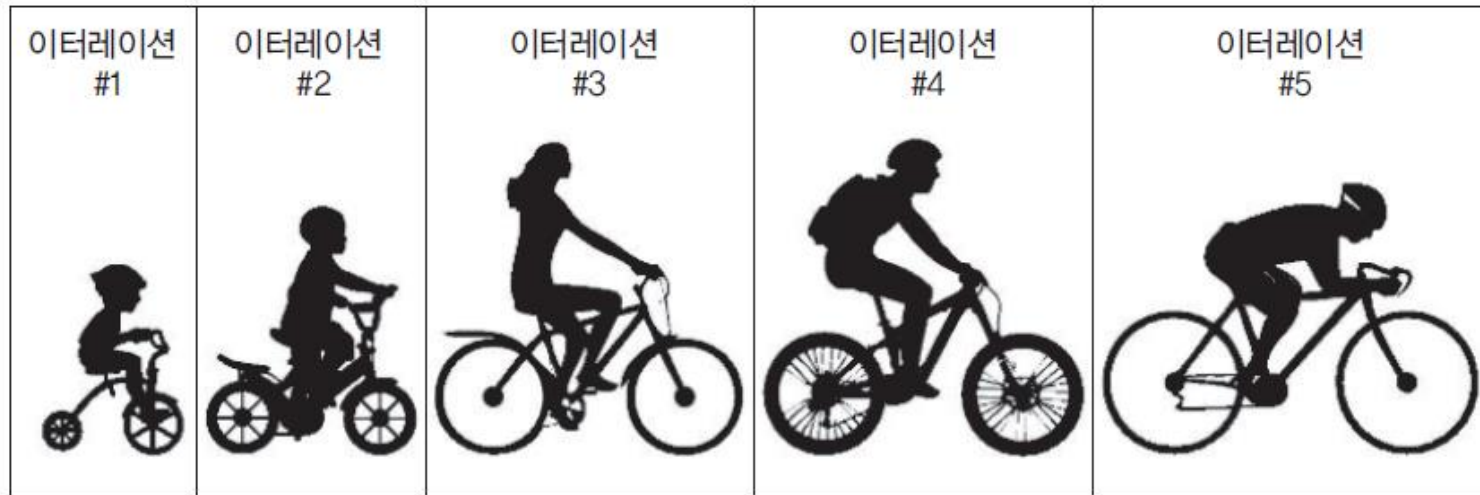
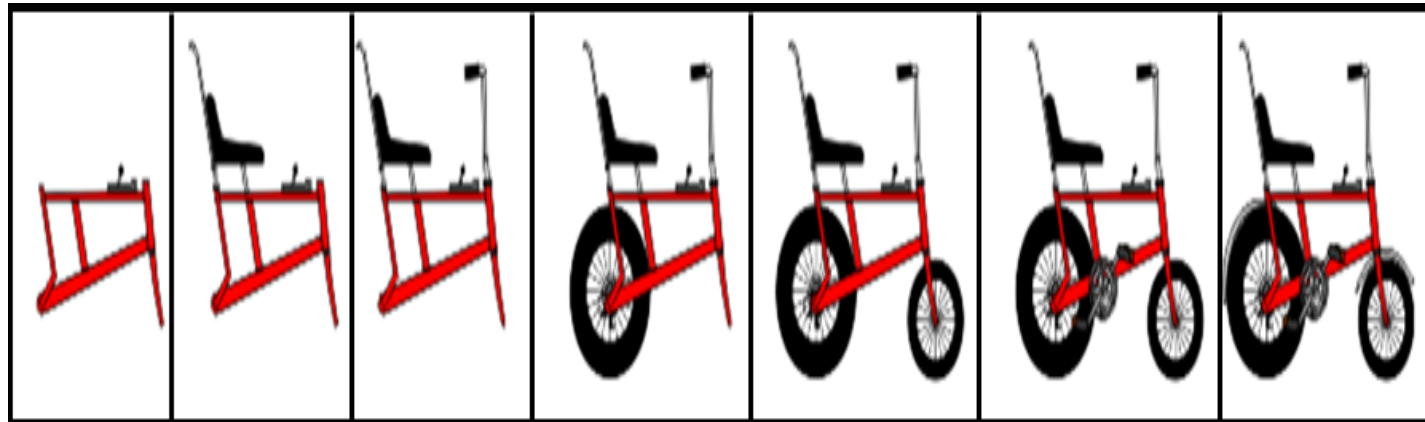


그림 1-7 애자일 방법에서의 반복적이면서 점진적 개발 방식



참고문헌

- 오픈 소스 소프트웨어로 실습하는 소프트웨어 공학, 정인상 교수님, 생능출판사
- 쉽게 배우는 소프트웨어 공학, 김치수, 한빛아카데미
- 소프트웨어 공학 이론과 실제, 홍장의, 한빛아카데미
- 소프트웨어공학의 모든 것, 최은만, 생능출판사
- 소프트웨어 공학, 조민호, 인피니트박스
- 소프트웨어 공학 에센셜, 윤청, 생능출판사



 **T h a n k y o u**

TECHNOLOGY

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Velit ex
plicabo ipsum, labore sed tempora ratione asperiores des
cenderat bore sed tempora rati jgert one bore sed tem!