1. 소프트웨어 공학 개요

학습목표

- 소프트웨어 유형과 특징
- 소프트웨어 공학 정의
- 소프트웨어 개발 공정
- 소프트웨어 개발 모델
- 소프트웨어 품질
- 소프트웨어 공학의 도전

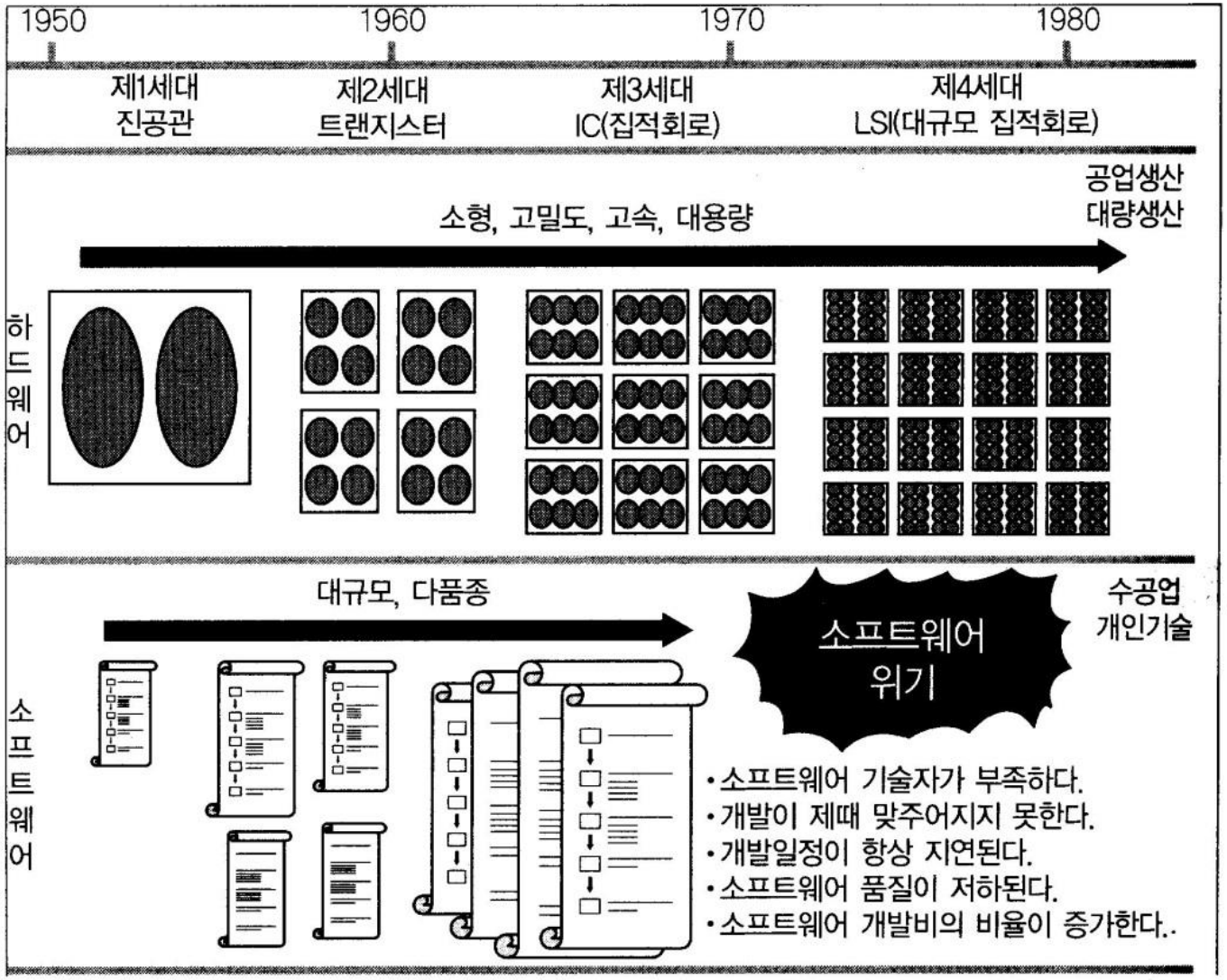
모든 곳에 사용되는 소프트웨어

소프트웨어 공학의 대두 배경

- 소프트웨어 위기(Software crisis)
  - 소프트웨어 수요 증가에 비해 공급 및 개발의 어려움

- 소프트웨어 위기의 해결
  - 다른 분야에서 사용했던 공학(Engineering) 패러다임을 이용하자는 결론
  - 1968년 NATO conference에서 소프트웨어 공학(Software Engineering) 제안됨

소프트웨어 위기



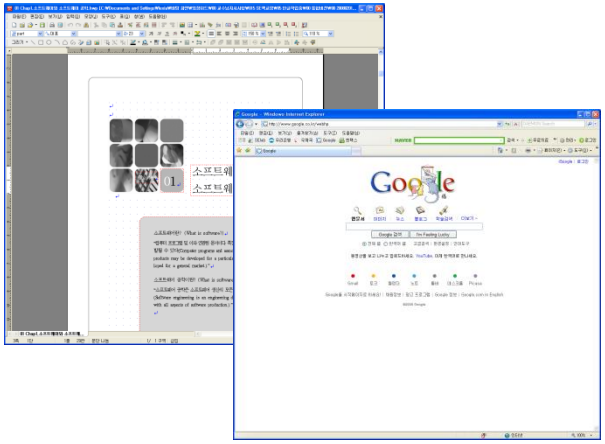
출처: 高橋 直久, 丸山 勝久, "ソフトウェア工学", 森北出版, 2010.

소프트웨어란?

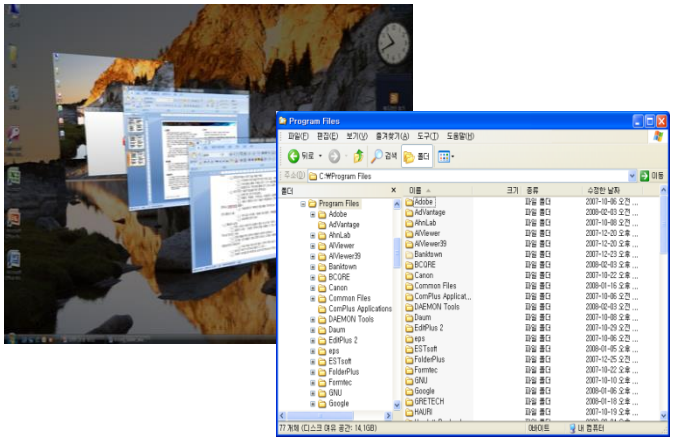
- 소프트웨어산업진흥법의 정의
  - 컴퓨터·통신·자동화 등의 장비와 그 주변 장치에 대하여 명령·제어·입력·처리·저장·출력·상호 작용이 가능하도록 하는 지시·명령의 집합과 이를 작성하기 위하여 사용된 기술서 기타 관련 자료
- 일반적인 정의
  - 단순한 프로그램 뿐 아니라 프로그램이 올바르게 작동하도록 하는데 필요한 관련된 모든 문서와 설치 데이터를 포함

소프트웨어의 유형 (1/2)

- 용도에 따른 분류
  - 응용 소프트웨어
    - 사용자의 원하는 목적에 맞게 개발된 소프트웨어
      - 예) 워드 프로세서(Word Processor), 스프레드 시트(Spread Sheet), 브라우저(Browser) 등
  - 시스템 소프트웨어
    - 하드웨어를 관리하고 응용 소프트웨어를 지원하는 소프트웨어
      - 예) 운영 체제(Operating System), 네트워크 관리 프로그램, 파일 관리 프로그램 등



워드프로세서와 브라우저



운영체제와 파일 관리 프로그램

소프트웨어의 유형 (2/2)

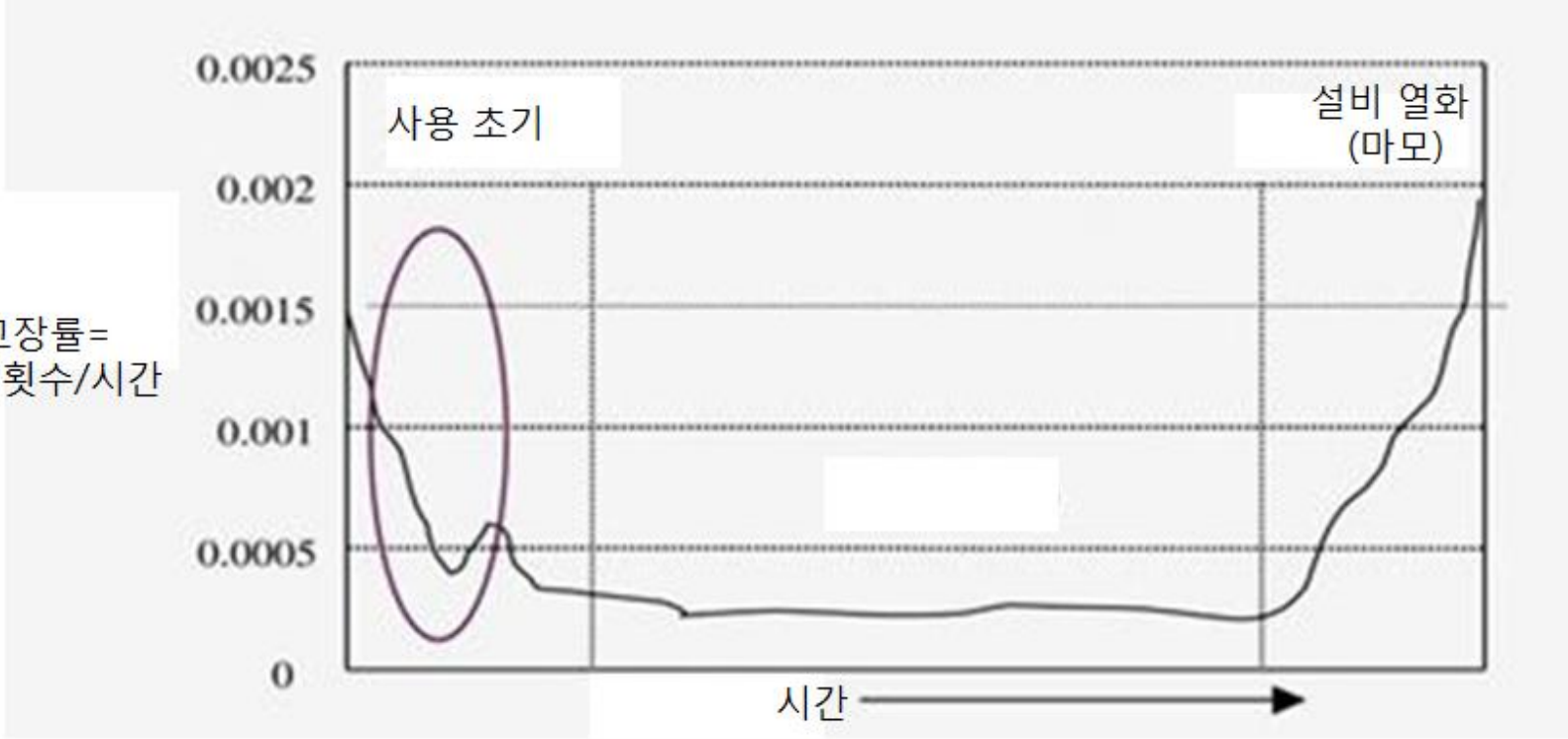
- 일반적인 제품
  - 전문 소프트웨어 개발사에 의해서 생산된 상용 제품
  - 소프트웨어 전문 유통 시장에서 적정 비용을 지급하여 구입
  - 예) 데이터베이스, 문서 편집기, 프로젝트 관리 도구, 그래픽 패키지 등과 같은 PC용 소프트웨어와 다양한 스마트 기기를 위한 앱 등
- 맞춤형 제품
  - 특정 고객을 위해 개발된 제품
  - 소프트웨어 개발자는 해당 고객의 요구사항을 만족하는 소프트웨어를 개발
  - 예) 전자 장치를 위한 제어 소프트웨어나 기업의 업무를 지원하기 위한 ERP 시스템, 공항의 항공관제 시스템, 대학의 학사 관리 시스템, 특정 국가의 전자정부 시스템 등

소프트웨어의 특징

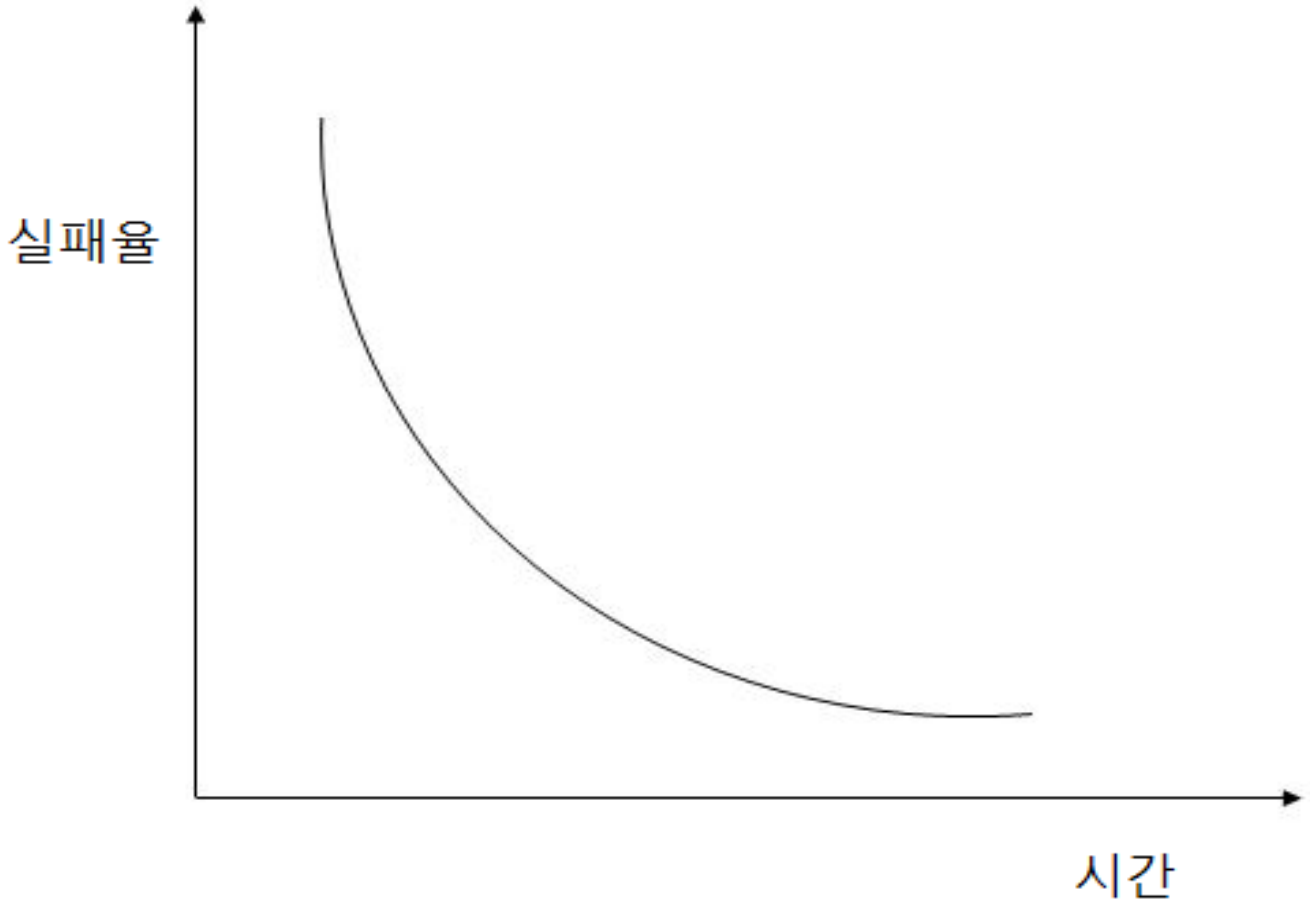
- 소프트웨어의 비가시성(Invisibility)
  - 소프트웨어 완제품의 구조가 개발된 코드 안에 숨어 있어 파악하기 힘든 특징

- 프레스만(Pressman)이 정의한 소프트웨어의 특징
  - 제조되지 않는 소프트웨어
    - 소프트웨어는 고전적인 의미의 '제조(Manufacture)'가 아니라, '개발(Development)'되는 것이다.
  - 마모되지 않는 소프트웨어
    - 소프트웨어는 닳지 않지만, 요구사항의 변경과 주변 환경의 변화에 따라 수정되고 진화한다.

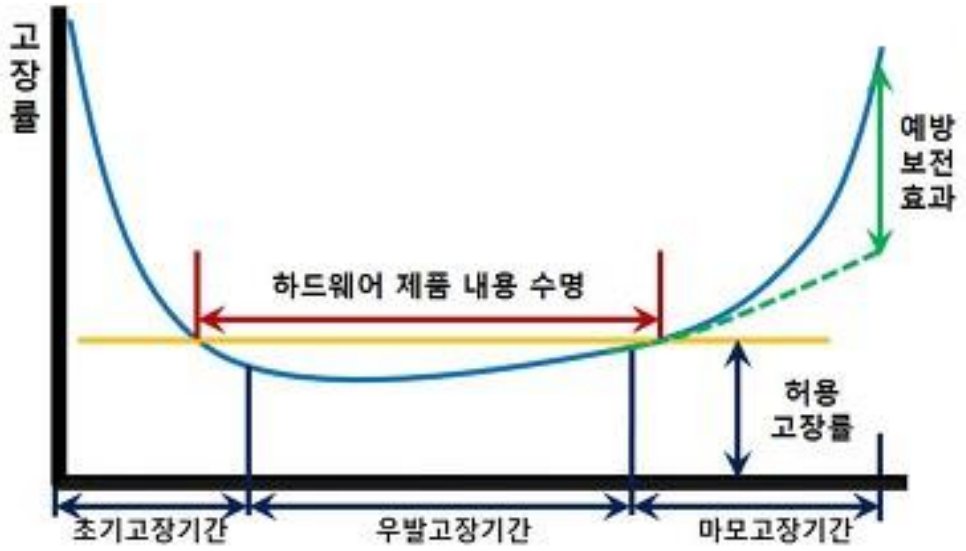
하드웨어 고장 곡선

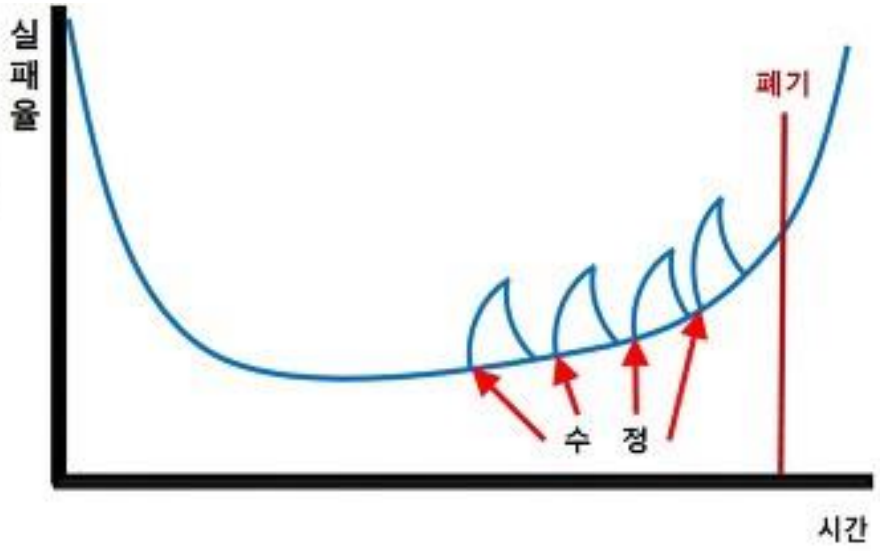


이상적인 소프트웨어 고장 곡선



하드웨어 고장 곡선과 소프트웨어 실패 곡선 비교

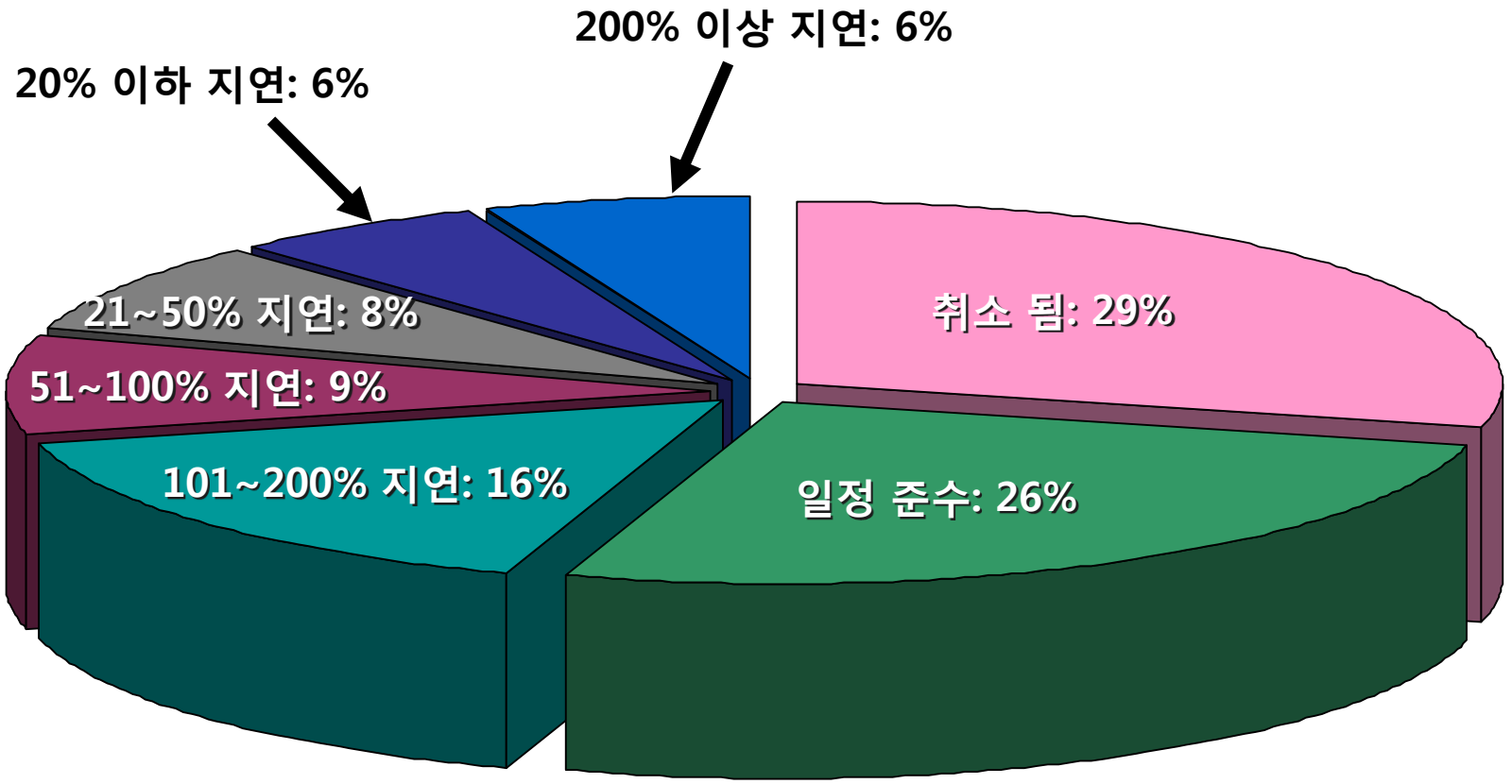




소프트웨어의 특성으로 인한 개발의 어려움

- 소프트웨어는,
  - 물리적인 형태가 없는 무형의 논리적인 요소
    - 개발 과정에 대해 정확하게 이해하기 어려움
    - 개발 진행 상황을 파악하기도 어려움
  - 최종 산출물이 개발 과정에서 확인되지 않음
    - 오류를 발견해야 할 시기를 놓치거나,
    - 오류에 대한 해결책을 못 찾는 경우가 발생
  - 프로젝트의 지연 및 예산 범위 초과로 인한 프로젝트 실패 가능성이 높음

미국 소프트웨어 프로젝트 결과

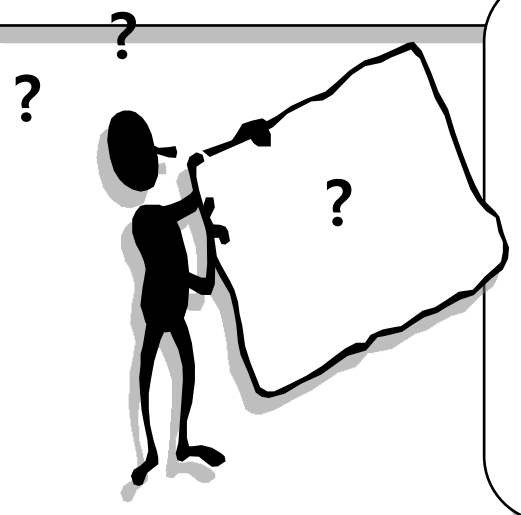


출처: Software Industry Benchmarking Study 2001

과거의 소프트웨어 개발

소프트웨어 프로그래밍 = 예술

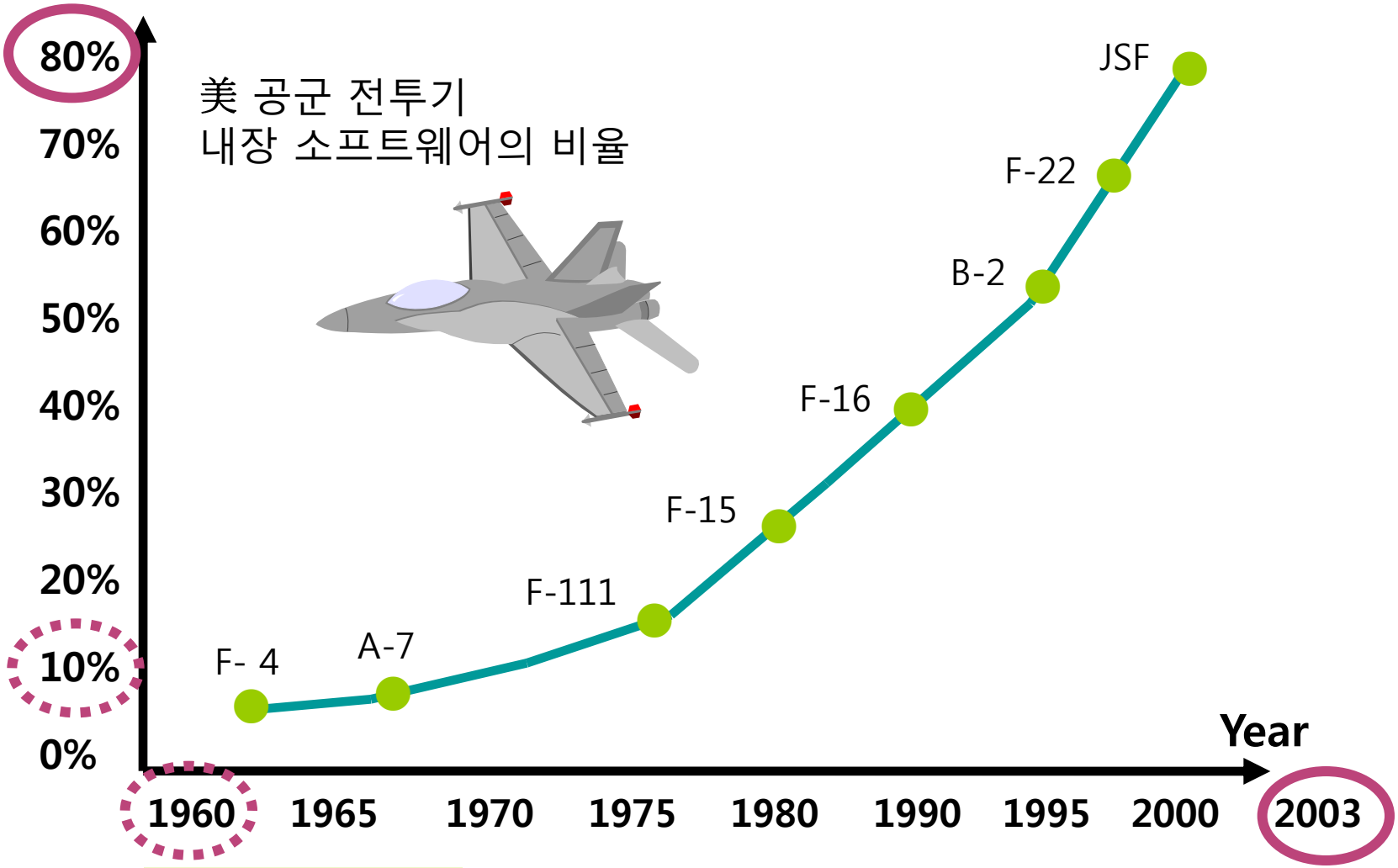
- 개발자에 따라 다양한 방식이 존재
- 사용자 = 프로그래머 = 유지보수 담당자



체계적 방법의 부재

- 정형적인 방법론이 거의 없고, 그것을 사용할 수 있는 사람도 거의 없음
- 프로그래머는 시행착오에 의해 기술을 습득함

점점 더 중요해지는 소프트웨어

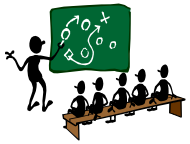


Source: U.S Airforce

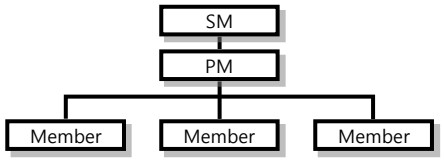
대규모 프로젝트의 어려움

수백 명의 개발자

- 의사소통 및 상호 협력의 어려움

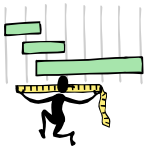


- 조직 및 팀 구조



오랜 개발 시간

- 프로젝트 관리



- 비용 및 일정 산정

모호하고 복잡한 요구사항

- 수백 페이지의 요구사항

- 빈번한 요구사항의 변화

소프트웨어 공학

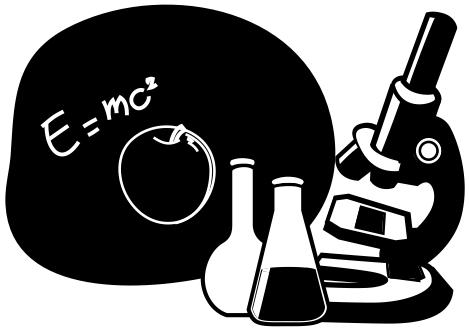
소프트웨어 공학의 다양한 정의

- Boehm
  - 과학적인 지식을 컴퓨터 프로그램 설계와 제작에 실제 적용하는 것이며, 이를 개발하고 운영하고 유지보수 하는데 필요한 문서화 과정을 포함
- Fairley
  - 허용하는 비용과 기간 내에서 소프트웨어 제품을 체계적으로 생산하고 유지보수 하는데 관련된 기술적이면서 관리적인 원리
- Humphrey
  - 양질의 소프트웨어의 효율적인 생산을 위한 공학, 과학, 수학적 원리와 방법들의 체계적인 적용

소프트웨어 공학이란?

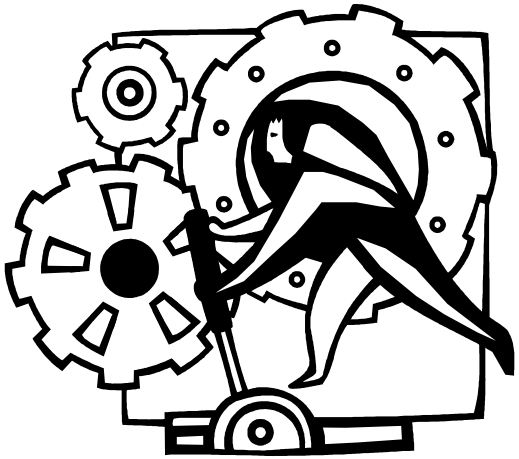
- IEEE의 정의
  - 소프트웨어의 개발, 운용, 유지보수 및 폐기에 대한 체계적인 접근 방법
- 특징
  - 소프트웨어 개발 전 과정에 걸쳐 필요한 이론, 개념 및 기술을 다룸
  - 소프트웨어 개발 과정에서 생성되는 모든 산출물이 그 대상이 됨
- 목표
  - 소프트웨어 개발이 체계적이고 공학적인 방법으로 이루어져 추정된 비용과 기간에 고객이 원하는 품질 높은 소프트웨어를 개발하는 것

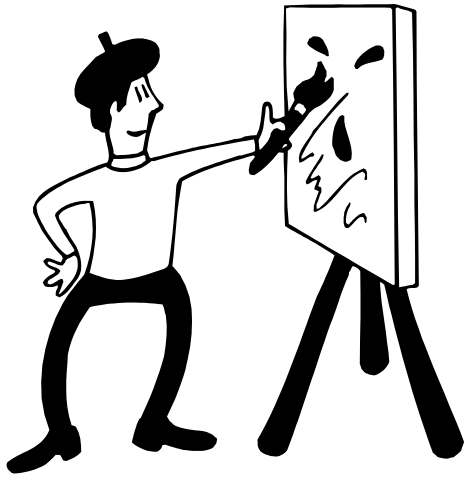
과학, 공학, 예술의 차이



과학

공학





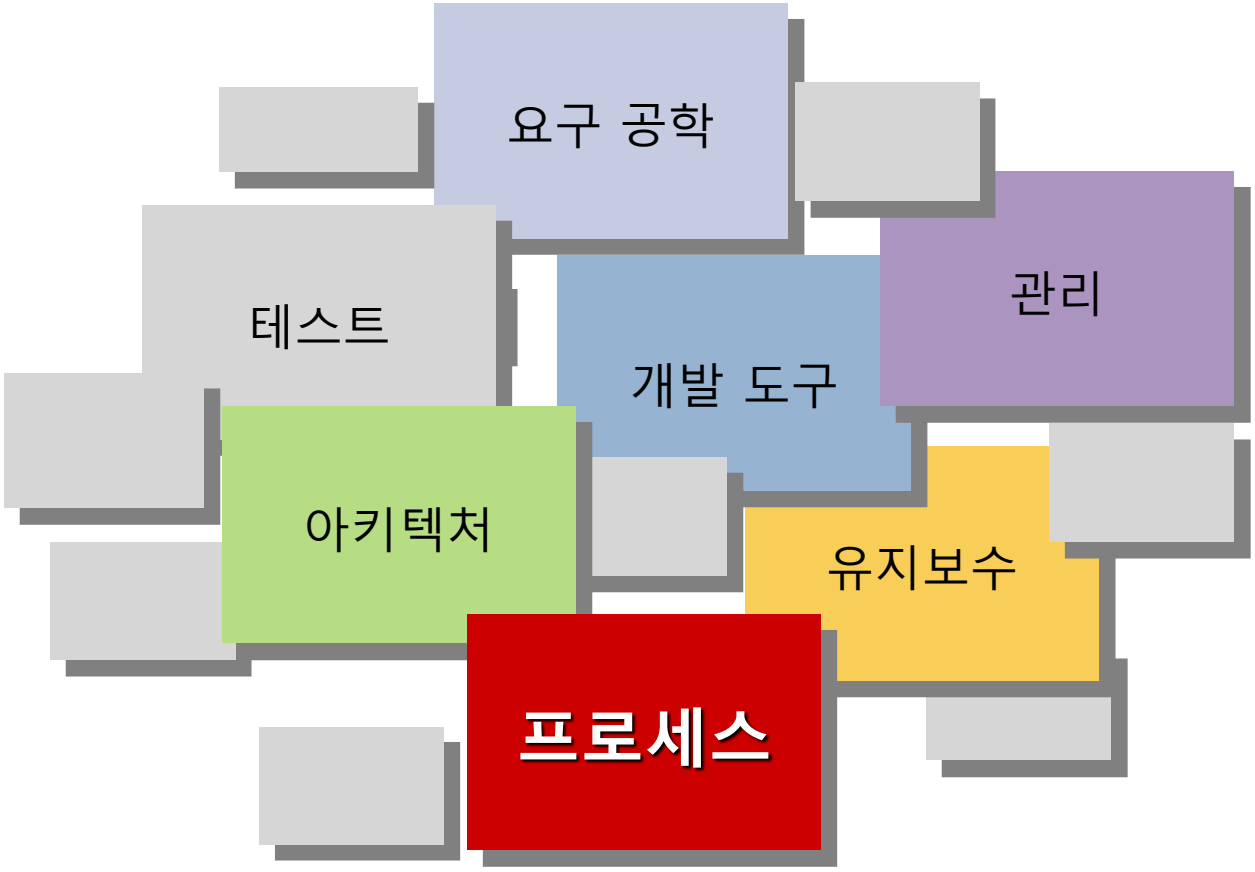
예술

공학이란?

- 의미
  - 실제적 문제(Practical Problem)를 해결하거나
  - 실제적인 산출물을 생산해내기 위해
  - 자원과 비용을 효과적으로 활용하면서
  - 과학적 지식을 적용하는 것

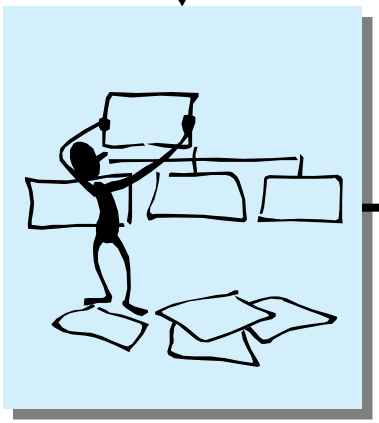
- 공학과 소프트웨어 공학
  - 공학
    - 업무분야에서 문제 발생 시, 실무자가 적절한 해답을 찾을 수 있도록 체계적으로 정리된 기술적 지식을 제공
  - 소프트웨어 공학
    - 소프트웨어 개발 기술, 절차 및 도구의 우수한 사례(Best Practice)들을 정리하여 소프트웨어 개발 시, 누구나 당면한 문제를 해결할 수 있도록 체계적인 기술적 지식을 제공

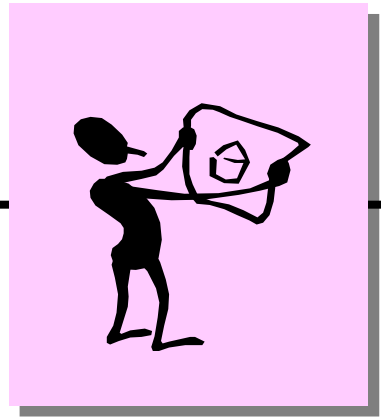
소프트웨어 공학의 주요 영역들

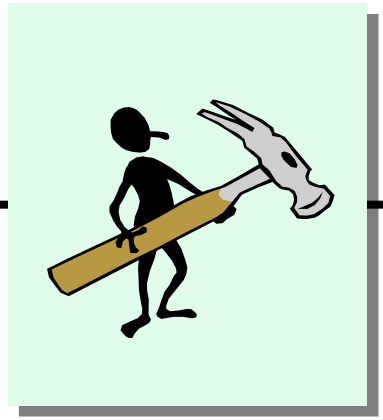


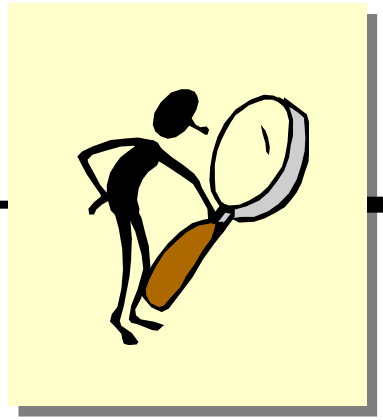
소프트웨어 개발 공정(프로세스)

고객의 요구



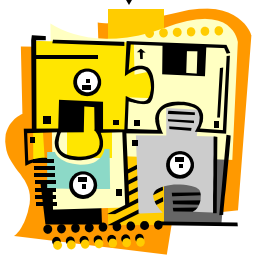






요구사항 분석 — 설계 — 구현 — 테스트

명세화 — *개발* — *확인*



S/W 제품

진화

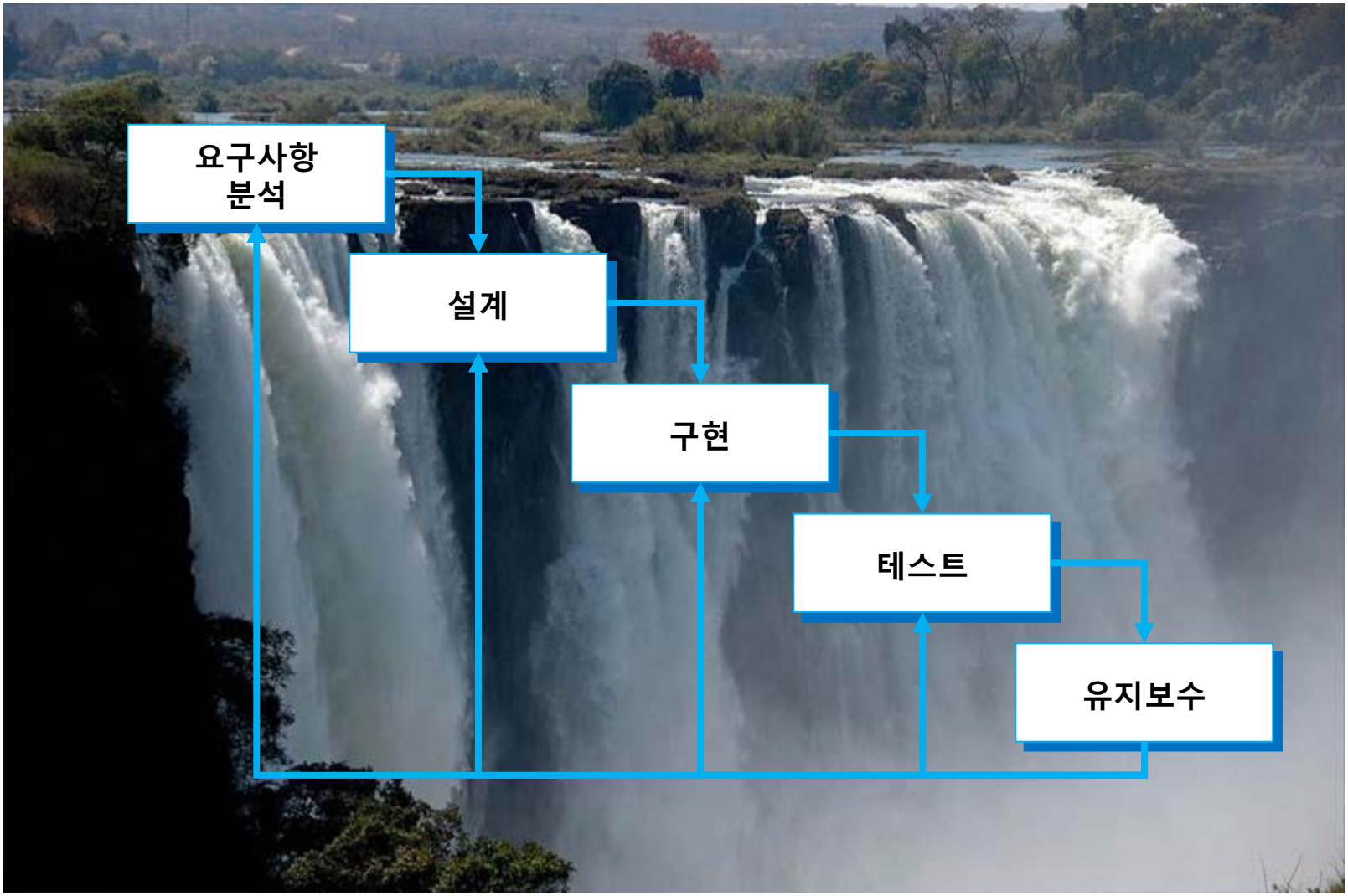
유지보수

소프트웨어 개발 모델

- 폭포수 모델: 요구 분석, 소프트웨어 설계, 구현, 테스트와 같이 분리된 프로세스 단계로 구성되며, 한 단계가 완료되어야 다음 단계로 진행된다.

- 반복적인 개발: 요구 분석, 소프트웨어 설계, 구현, 테스트 활동이 서로 겹친다. 초기 시스템은 매우 추상적인 명세서로부터 빠르게 개발된다. 추후에 고객의 요구사항을 만족시키는 시스템을 생산하기 위해서 고객의 요구사항을 기반으로 다듬어져서 고객에게 인도될 수 있다.

소프트웨어 개발 모델 : 폭포수 모델



소프트웨어 품질

[표 1-1] 좋은 소프트웨어가 가져야 할 품질 특성

품질 특성	의미
유지보수성	소프트웨어는 고객의 요구사항 변경을 쉽게 수용할 수 있는 방법으로 작성되어야 한다. 비즈니스 환경의 변화로 인해 소프트웨어는 끊임없이 변하기 때문에 본질적이며 중요한 속성이다.
확실성(dependability)	소프트웨어 확실성은 신뢰성, 보안성, 안전성을 포함하는 포괄적인 특성이다. 믿을 수 있는 소프트웨어는 장애가 발생하여도 물리적 혹은 경제적 손실을 입히지 않는다.
효율성	소프트웨어는 메모리, 프로세서와 같은 시스템 자원을 낭비하지 말아야 한다. 효율성은 반응시간, 처리시간, 메모리 활용성 등을 포함한다.
사용용이성	소프트웨어는 사용자가 사용하기 쉬어야 한다. 이것은 적절한 사용자 인터페이스와 문서를 가지고 있어야 한다는 것을 의미한다.

소프트웨어 공학의 도전

- 다양한 환경 – 이질성
  - 일반적으로 SW 시스템은 다수의 이기종 컴퓨터를 포함하는 네트워크 상에서 작동하는 분산 시스템에서 운영
- 짧은 개발 시간
  - 현대의 비즈니스 환경은 매우 빠르게 변화하고 경쟁이 치열하기 때문에, 과거와 달리 SW 개발이 신속하게 이루어져야 함
  - 빠른 개발과 신속한 인도에도 불구하고 품질은 유지되어야 함
- 확실성(dependability)
  - 우리의 모든 실생활과 관련되기 때문에 신뢰할 수 있는 SW를 개발해야 함