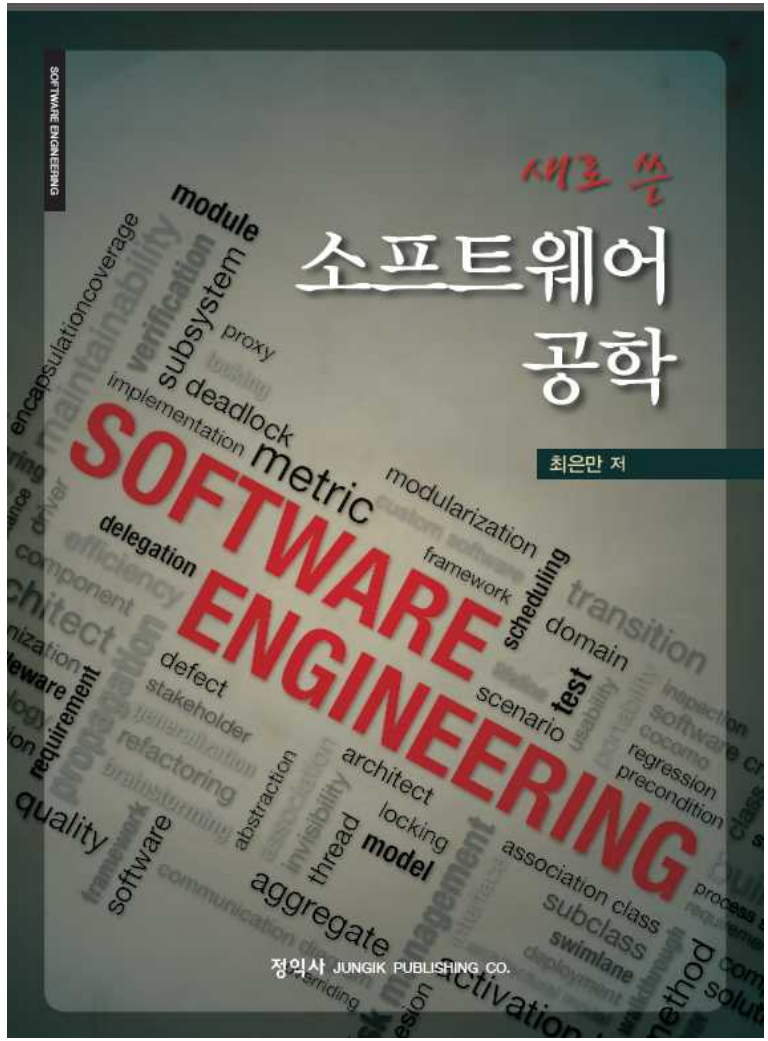
소프트웨어 공학

Lecture #2: 프로세스

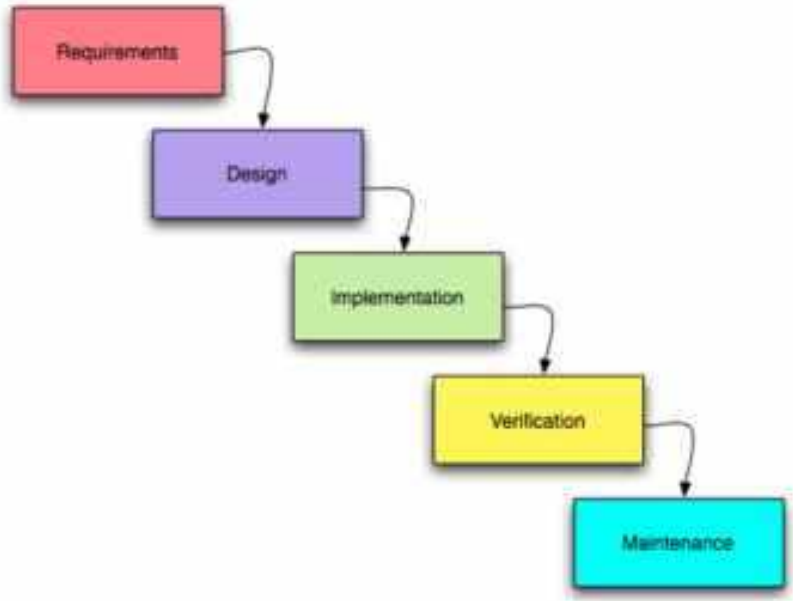
최은만 저

6차 개정판



학습 목표

- 소프트웨어 프로세스
- 바람직한 프로세스의 특징
- 소프트웨어 프로세스 모델
- 지원 프로세스



프로세스

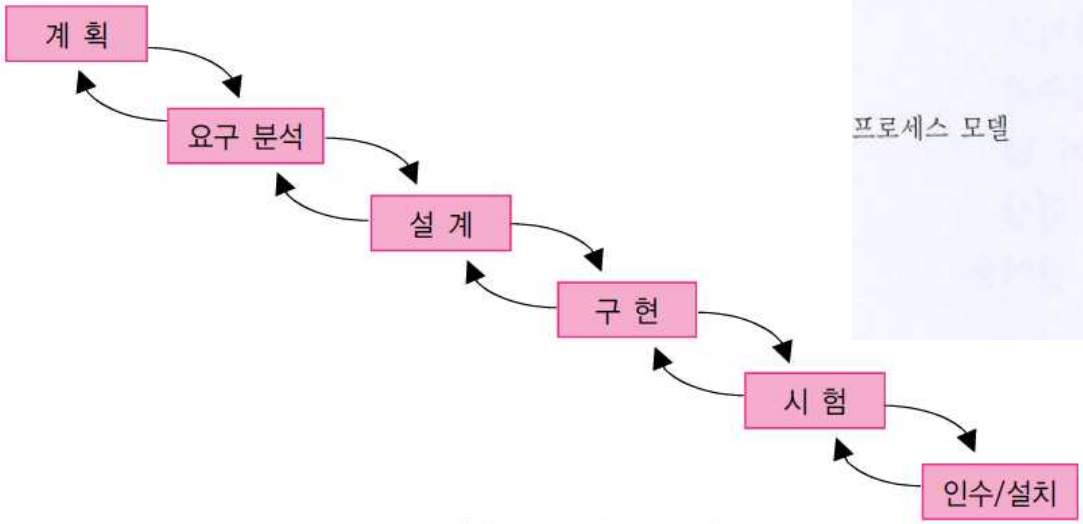
- 정의
  - 어떤 일을 하기 위한 특별한 방법으로 일반적으로 단계나 작업으로 구성됨(웹스터 영어 사전)
- 소프트웨어를 개발하는 과정, 즉 작업 순서
  - 순서제약이 있는 작업의 집합
  - 높은 품질과 생산성이 목표
- 프로세스가 없는 개발
  - Code-and-fix

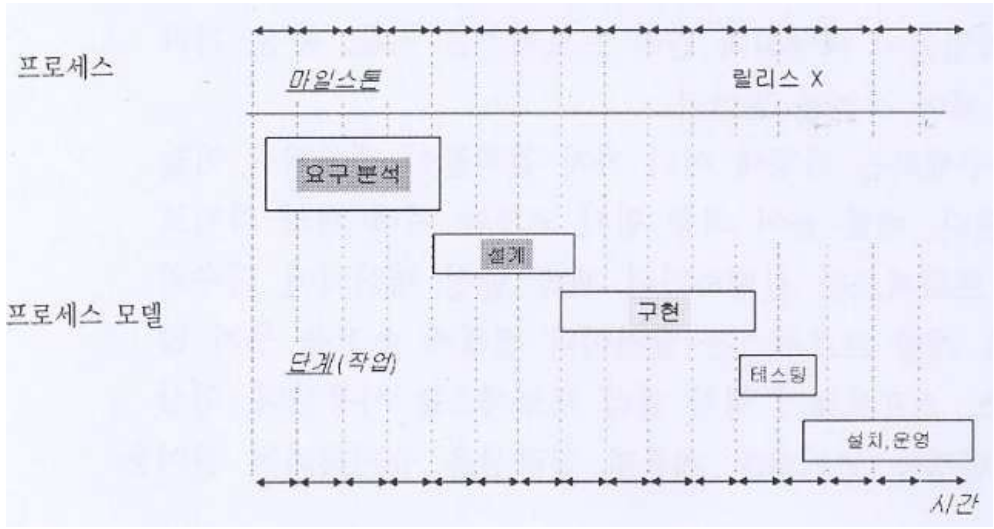
프로세스와 방법론의 비교

	프로세스	방법론
특징	• 단계적인 작업의 틀을 정의한 것 • 무엇을 하는가에 중점 • 결과물이 표현에 대하여 언급 없음 • 패러다임에 독립적 • 각 단계가 다른 방법론으로도 실현 가능	• 프로세스의 구체적인 구현에 이름 • 어떻게 하는가에 중점 • 결과물을 어떻게 표현하는지 표시 • 패러다임에 종속적 • 각 단계의 절차, 기술, 가이드라인을 제시
사례	• 폭포수 프로세스 • 나선형 프로세스 • 프로토타이핑 프로세스 • Unified 프로세스 • 애자일 프로세스	• 구조적 분석, 설계 방법론 • 객체지향 방법론 • 컴포넌트 • 애자일 방법론

2.1 소프트웨어 프로세스

- 소프트웨어 개발에 대한 기술적, 관리적 이슈를 다루는 작업
  - 개발 모델별 컴포넌트 프로세스, 부프로세스 존재
  - 서로 다른 목적
  - 서로 협력하여 전체 목적을 만족



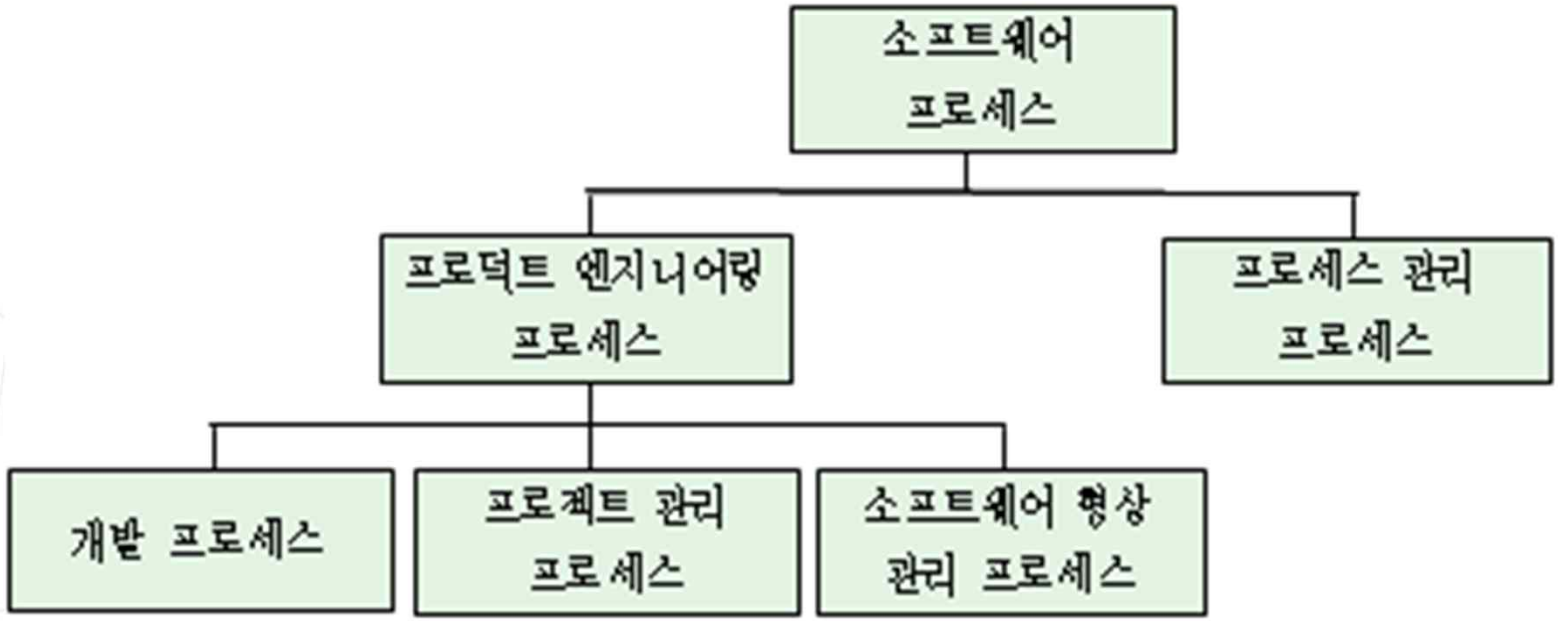


프로세스와 프로세스 모델

- 소프트웨어 프로젝트
  - 수행할 작업을 조직화한 프로세스를 이용
  - 비용, 일정, 품질에 대한 목표를 성취하는 것
- 프로세스 명세
  - 프로젝트에서 수행하여야 하는 작업과 이들의 수행 순서를 정의
  - 실행 프로세스는 다를 수 있음
- 프로세스 모델
  - 일반적인 프로세스를 기술한 것
  - 작업의 단계와 순서
  - 각 단계 작업 수행의 제약사항이나 조건 등을 모아 놓은 것

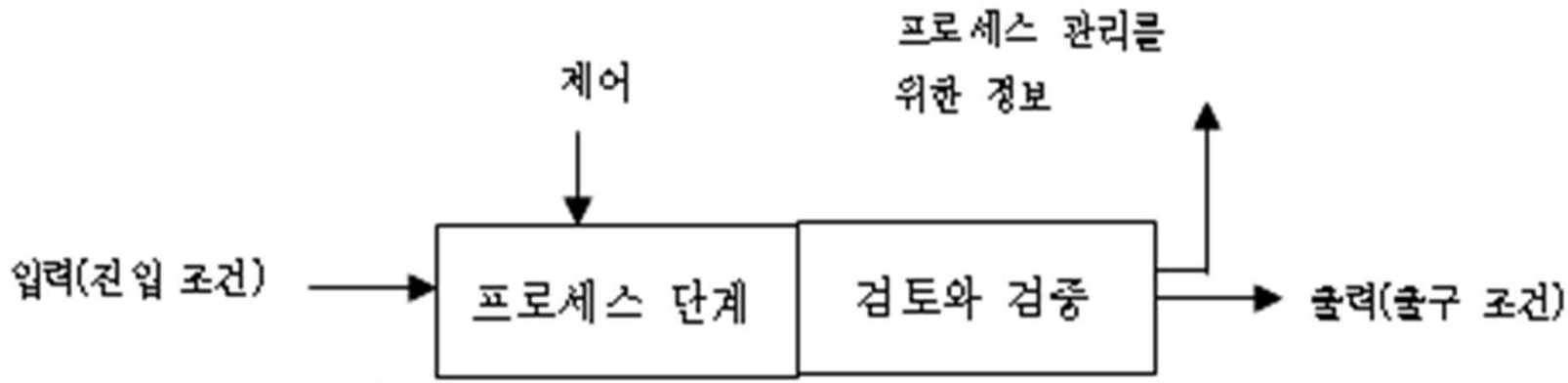
프로세스의 종류

- 프로젝트의 중심 프로세스
  - 개발 프로세스
  - 관리 프로세스
- 기타 프로세스
  - 형상 관리 프로세스
  - 프로세스 관리 프로세스



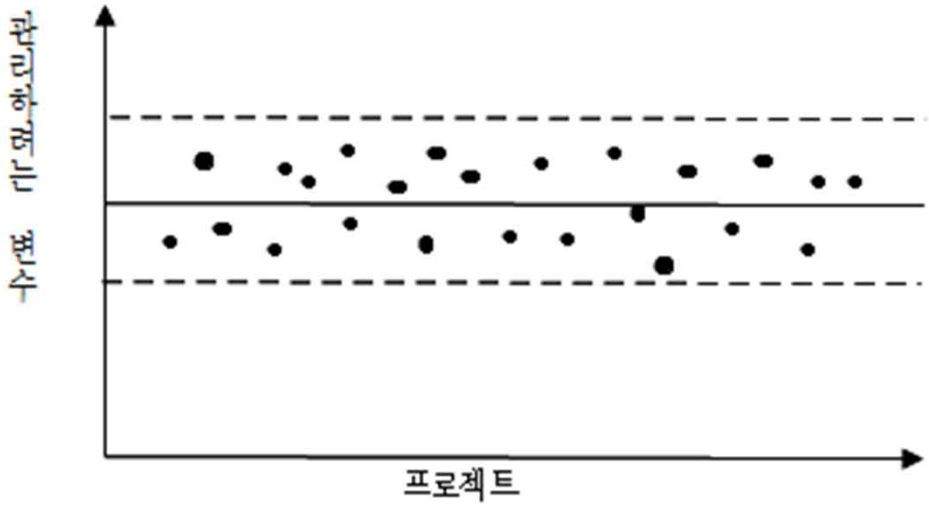
프로세스의 정의

- 작업 결과와 검증 조건을 명확히 정의하여야 함
- 작업 방법
- 진입 조건, 출구 조건

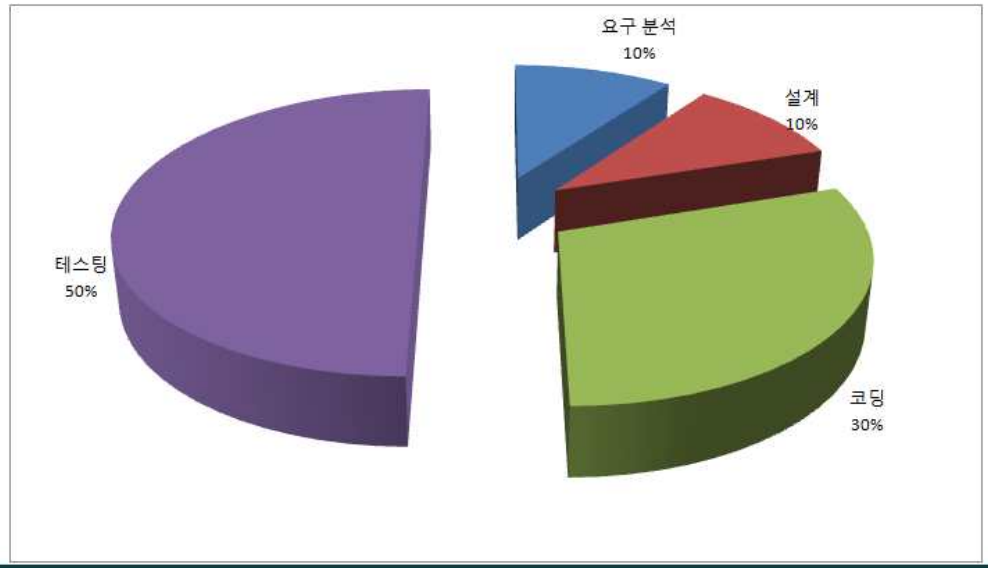


2.2 바람직한 프로세스의 특징(1)

- 예측 가능성



- 테스팅과 유지보수 지원



바람직한 프로세스의 특징(2)

- 변경 지원 – 변경을 쉽게 다룰 수 있는 프로세스

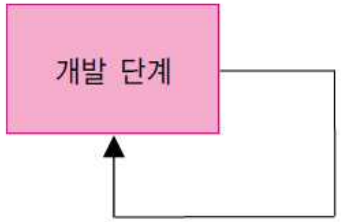


그림 2.7 ▶ 변경을 위한 피드백

- 결함 제거

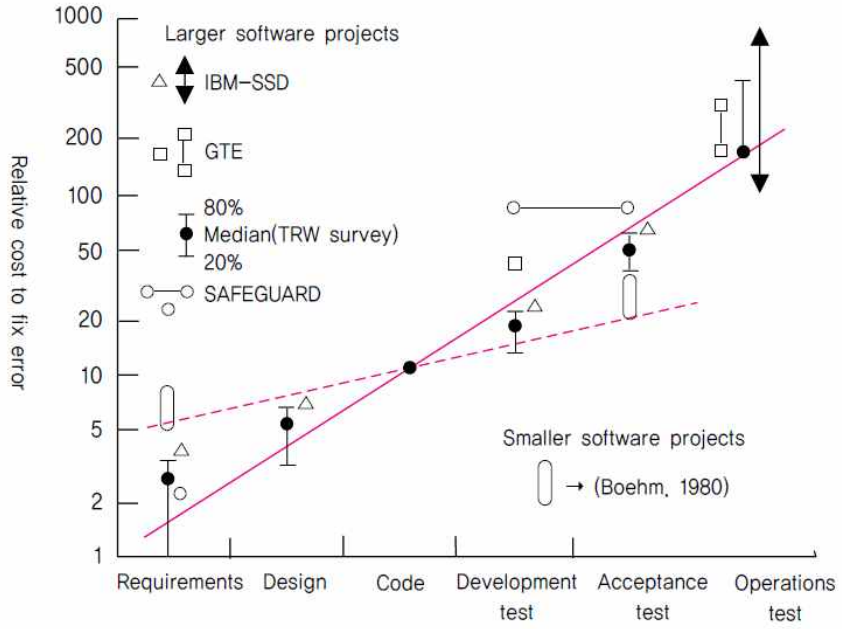


그림 2.8 ▶ 오류 수정 비용

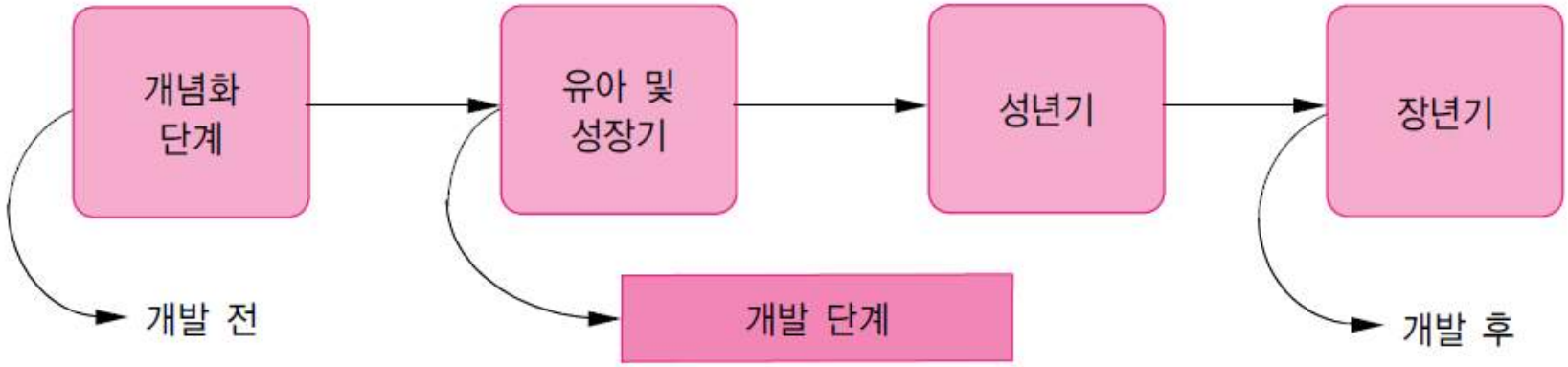
2.3 소프트웨어 개발 프로세스

- 프로세스 모델
  - 일반적인 모델이 될만한 프로세스를 기술한 것

- 대표적인 프로세스 모델
  - 폭포수 모델
  - 프로토타이핑 모델
  - 점증적 모델
  - V 모형
  - 일정 중심 설계 모델
  - 진화적 출시 모델
  - 애자일 모델

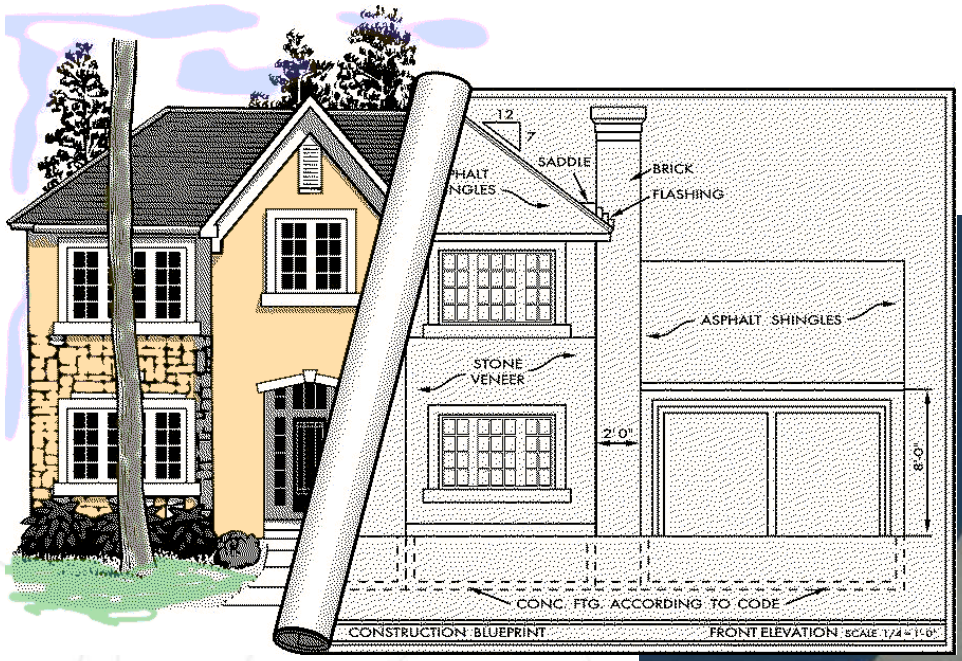
소프트웨어 생명주기

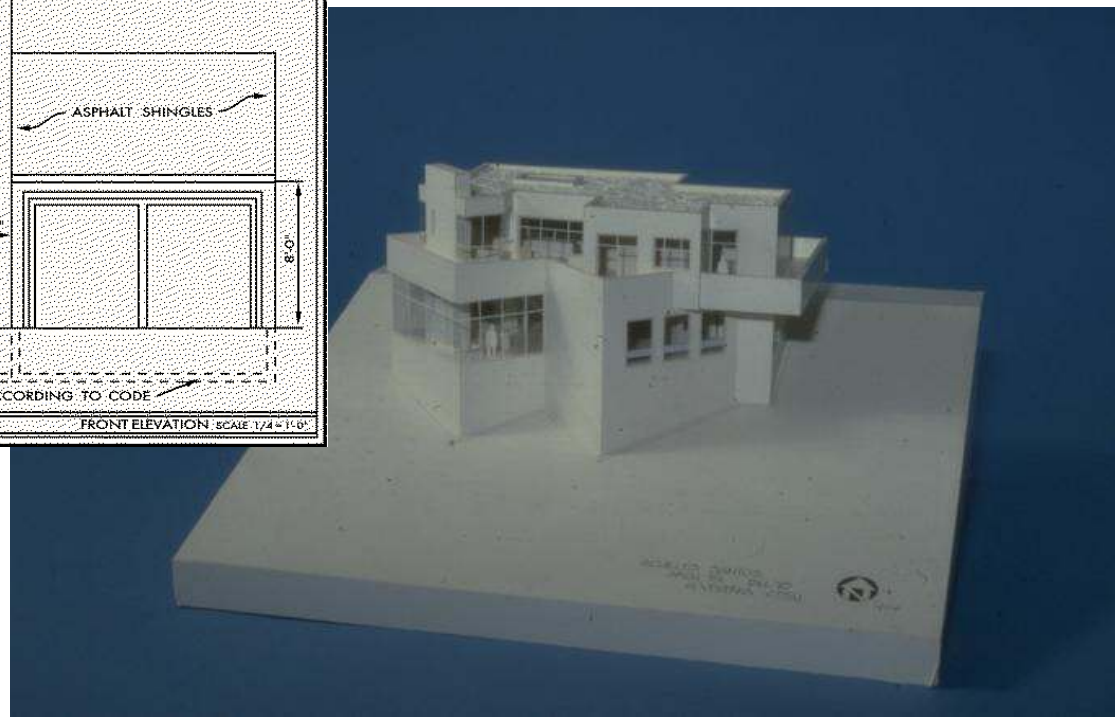
- **소프트웨어 생명주기**



SE와 유사한 작업들

- 건물의 건축





계획

- 다음 질문의 대답을 찾는 단계
  - How much will it cost?
  - How long will it take?
  - How many people will it take?
  - What might go wrong?

- 범위 정하기
- 산정(Estimation)
- 리스크 분석
- 일정 계획
- 관리 전략 수립

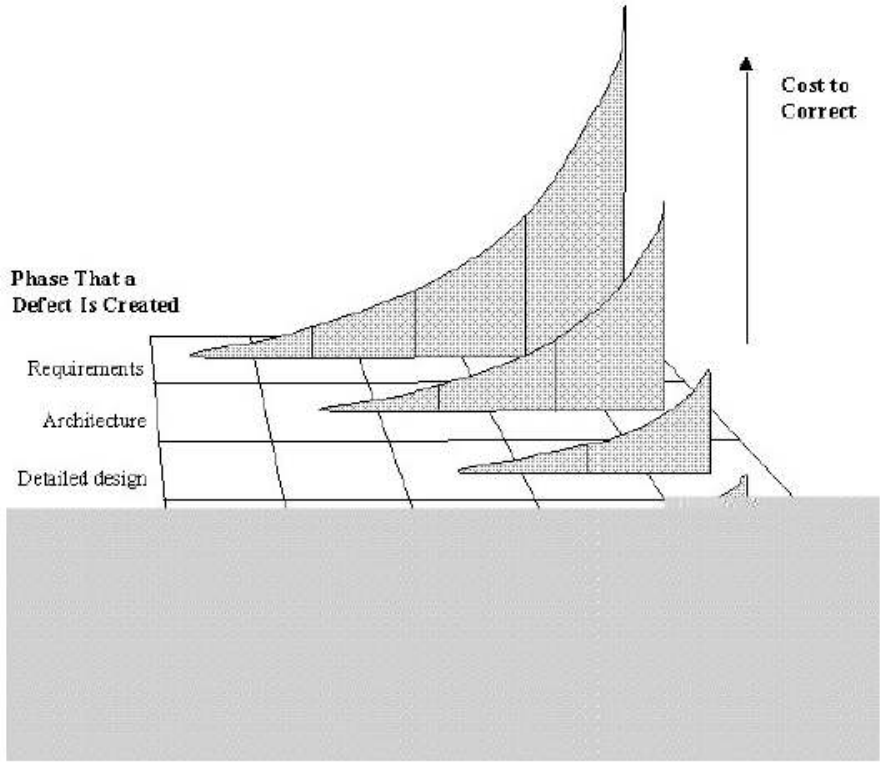
Why 단계

ROI

Concept 정립

요구 분석

- 요구 – 시스템이 가져야 할 능력(capability)과 조건(condition)
- What 의 단계
- 응용 분야(도메인)에 집중
- 가장 중요하고도 어려운 단계
  - 조그만 차이가 큰 오류로 변함
- 결과물: 요구분석서(SRS)



설계

- How의 단계
- 솔루션에 집중
- 아키텍처 설계
- 데이터베이스 설계
- UI 설계
- 상세 설계
- 결과물: 설계서(SD)

구현

- ‘Do it’ 단계
- 코딩과 단위 테스트
- 설계 또는 통합 단계와 겹치기도 함
  - 전체 일정을 줄이기 위하여
  - 협력 작업이 필요한 경우
- 특징
  - 압력 증가
  - 최고의 인력 투입
- 이슈
  - Last minute change
  - Communication overhead
  - 하청 관리

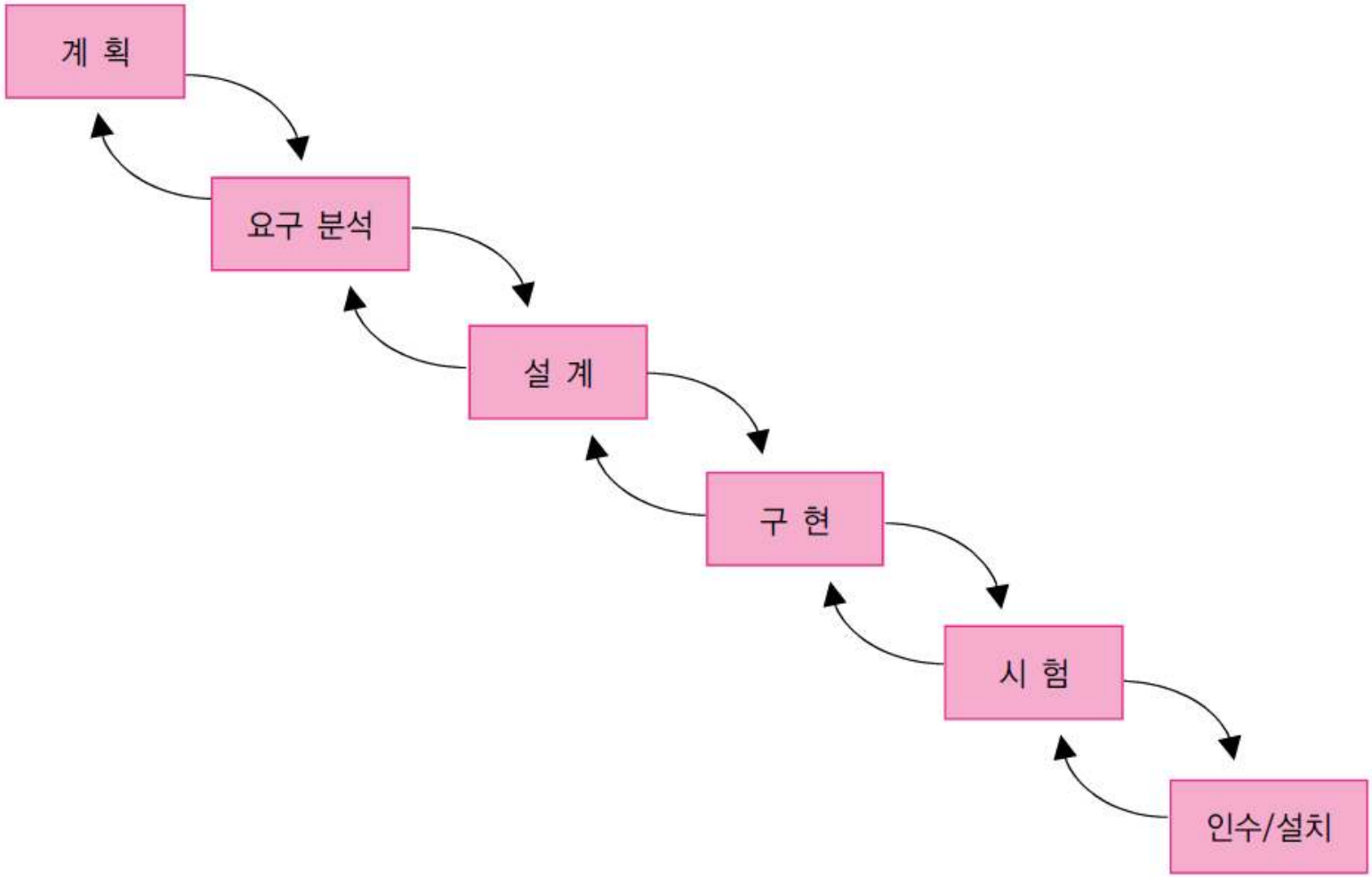
통합과 테스트

- 병행
  - 통합해 나가면서 테스트 시작
- 모듈의 통합으로 시작
- 점차 완성된 모듈을 추가
- 통합은 개발자가 주로 담당
- 테스트는 QA 팀이 주로 담당
- 단계적인 테스트
  - 단위, 통합, 시스템
- 목적 중심 테스트
  - 스트레스 테스트, 성능 테스트, 베타 테스트, Acceptance 테스트, Usability 테스트

설치와 유지보수

- 시스템의 타입에 따라 다른 설치 방법
  - Web-based, CD-ROM, in-house, etc.
- 이전(Migration) 정책
- 시스템의 사용을 시작하게 하는 방법
  - 병행 운용
- 설치는 개발 프로젝트의 일부, 유지보수는 별개
- 유지보수
  - 결함을 고침
  - 새 기능 추가
  - 성능 추가

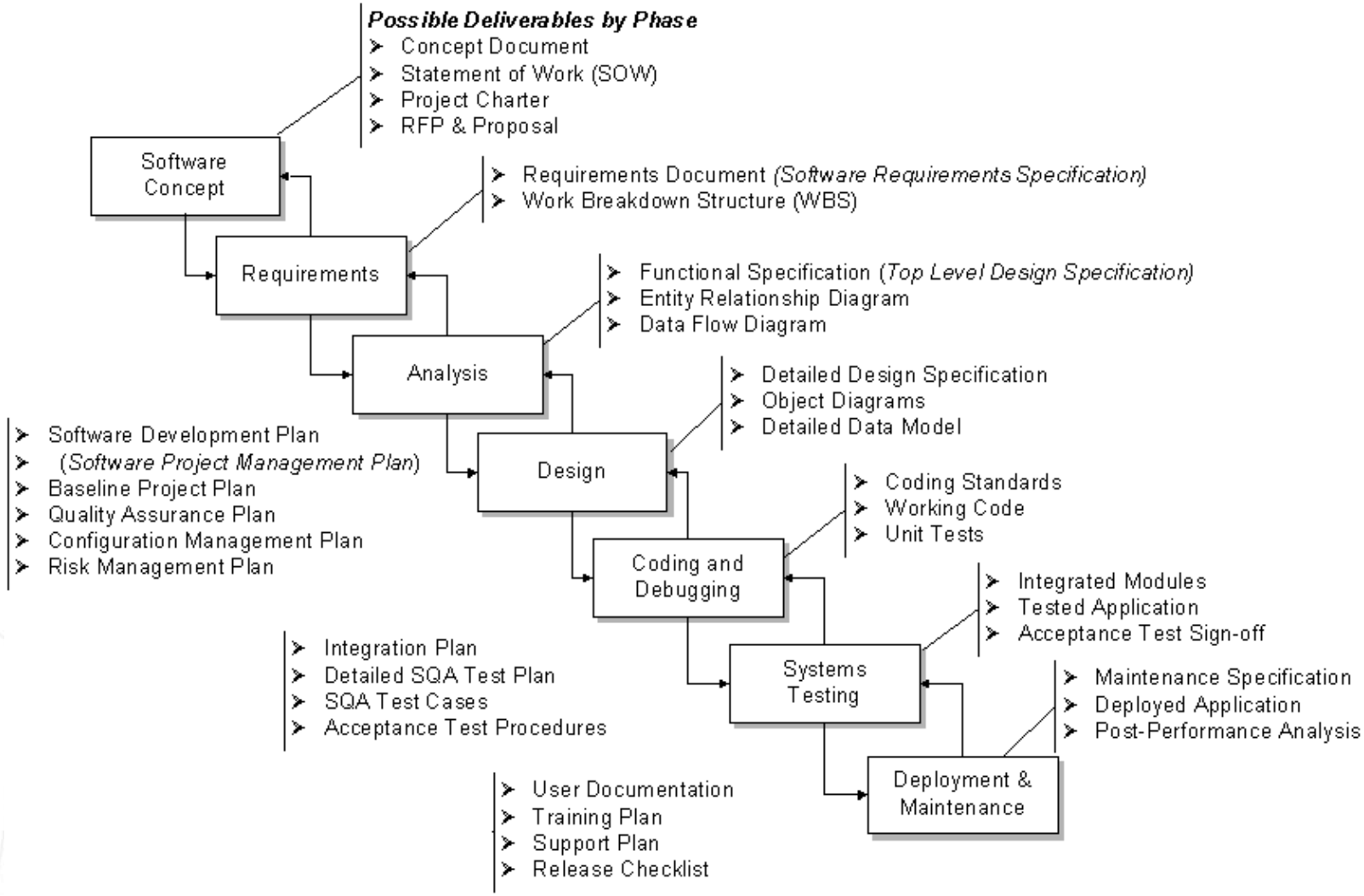
(1) 폭포수(waterfall) 모델



폭포수(waterfall) 모델

- 1970년대 소개
  - 항공 방위 소프트웨어 개발 경험으로 습득
- 각 단계가 다음 단계 시작 전에 끝나야 함
  - 순서적 - 각 단계 사이에 중복이나 상호작용이 없음
  - 각 단계의 결과는 다음 단계가 시작 되기 전에 점검
  - 바로 전단계로 피드백
- 단순하거나 응용 분야를 잘 알고 있는 경우 적합
  - 한 번의 과정, 비전문가가 사용할 시스템 개발에 적합
- 결과물 정의가 중요
- Method vs. Methodology

폭포수 모델의 단계별 결과물

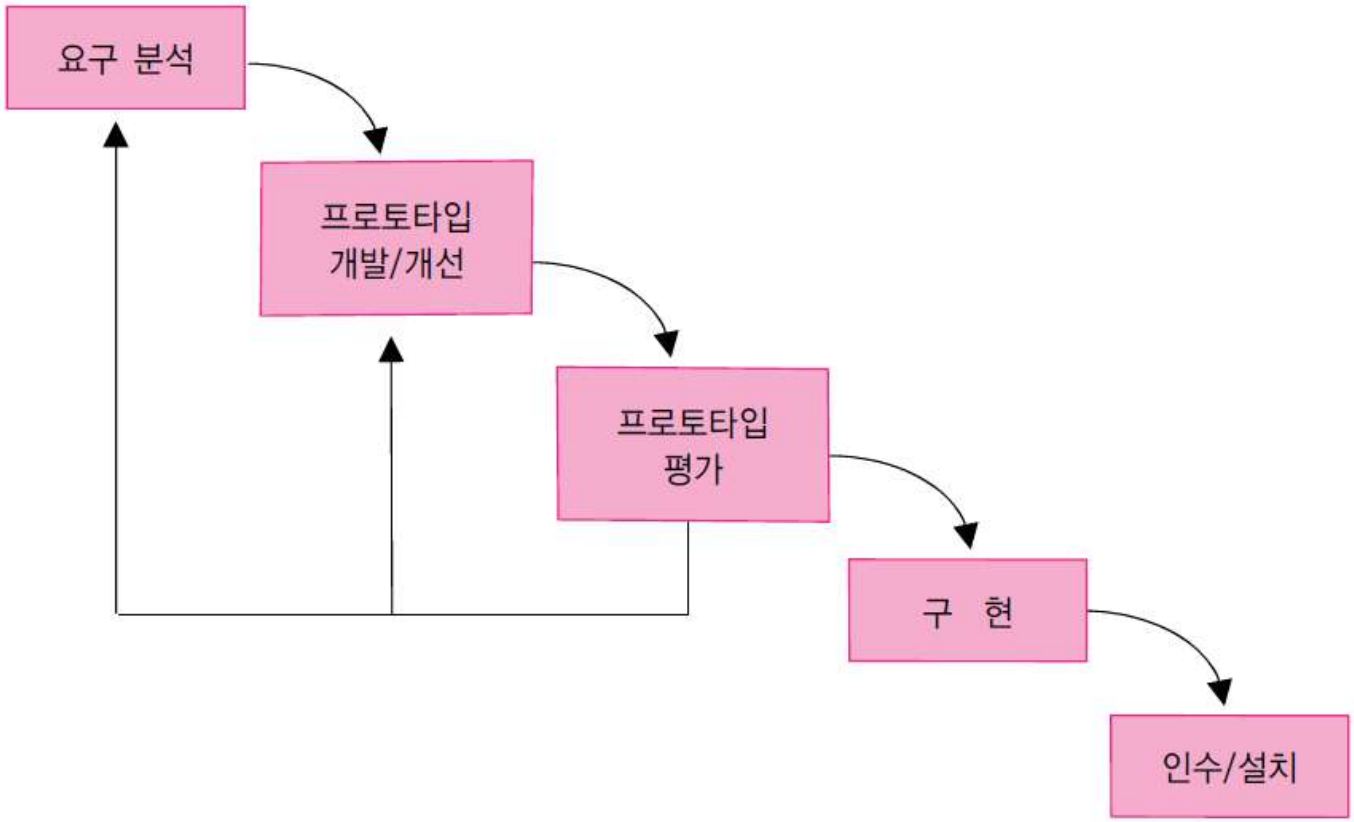


폭포수 모형의 장단점

- 장점
  - 프로세스가 단순하여 초보자가 쉽게 적용 가능
  - 중간 산출물이 명확, 관리하기 좋음
  - 코드 생성 전 충분한 연구와 분석 단계
- 단점
  - 처음 단계의 지나치게 강조하면 코딩, 테스트가 지연
  - 각 단계의 전환에 많은 노력
  - 프로토타입과 재사용의 기회가 줄어듦
  - 소용 없는 다종의 문서를 생산할 가능성 있음
- 적용
  - 이미 잘 알고 있는 문제나 연구 중심 문제에 적합
  - 변화가 적은 프로젝트에 적합

(2) 프로토타이핑 모델

- **Rapid Prototyping Model(RAD)**



프로토타이핑 모델

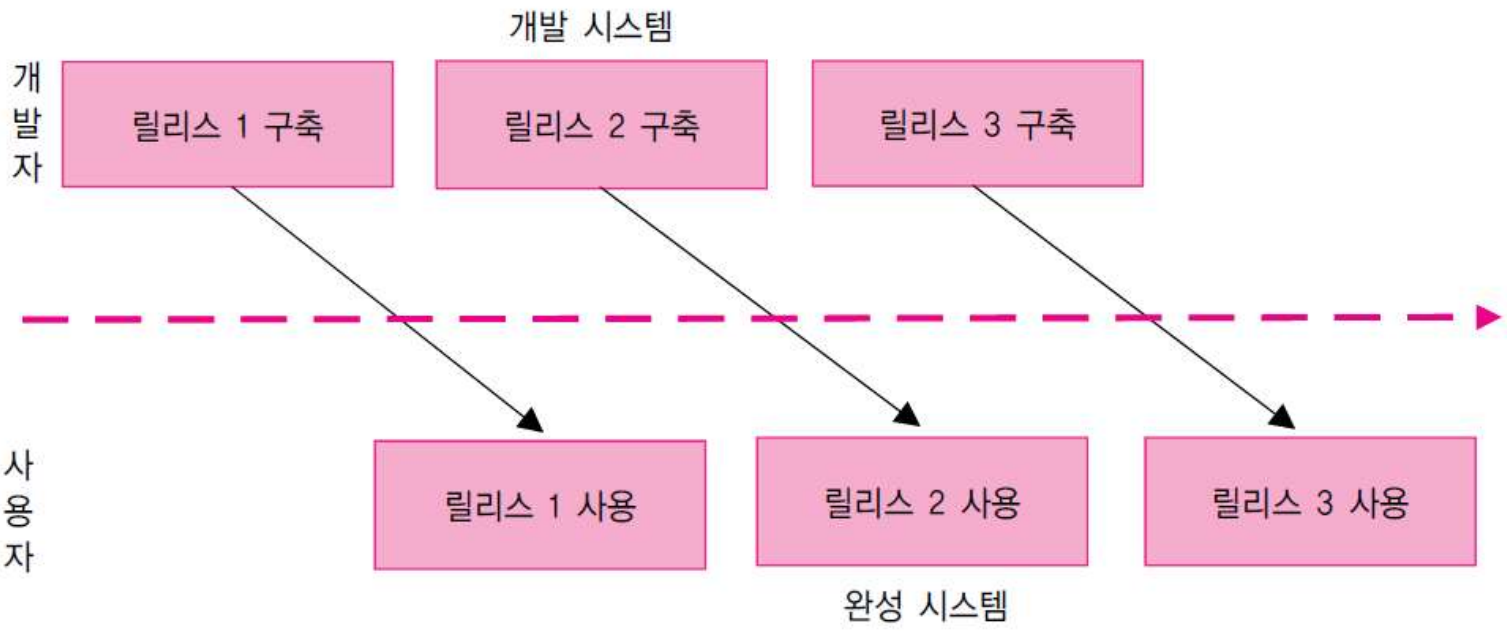
- 프로토타입(시범 시스템)의 적용
  - 사용자의 요구를 더 정확히 추출
  - 알고리즘의 타당성, 운영체제와의 조화, 인터페이스의 시험 제작
- 프로토타이핑 도구
  - 화면 생성기
  - 비주얼 프로그래밍, 4세대 언어 등
- 공동의 참조 모델
  - 사용자와 개발자의 의사소통을 도와주는 좋은 매개체
- 프로토타입의 목적
  - 단순한 요구 추출 – 만들고 버림
  - 제작 가능성 타진 - 개발 단계에서 유지보수가 이루어짐

프로토타이핑 모델의 장단점

- 장점
  - 사용자의 의견 반영이 잘 됨
  - 사용자가 더 관심을 가지고 참여할 수 있고 개발자는 요구를 더 정확히 도출할 수 있음
- 단점
  - 오해, 기대심리 유발
  - 관리가 어려움(중간 산출물 정의가 난해)
- 적용
  - 개발 착수 시점에 요구가 불투명할 때
  - 실험적으로 실현 가능성을 타진해 보고 싶을 때
  - 혁신적인 기술을 사용해 보고 싶을 때

(3) 진화적 모델

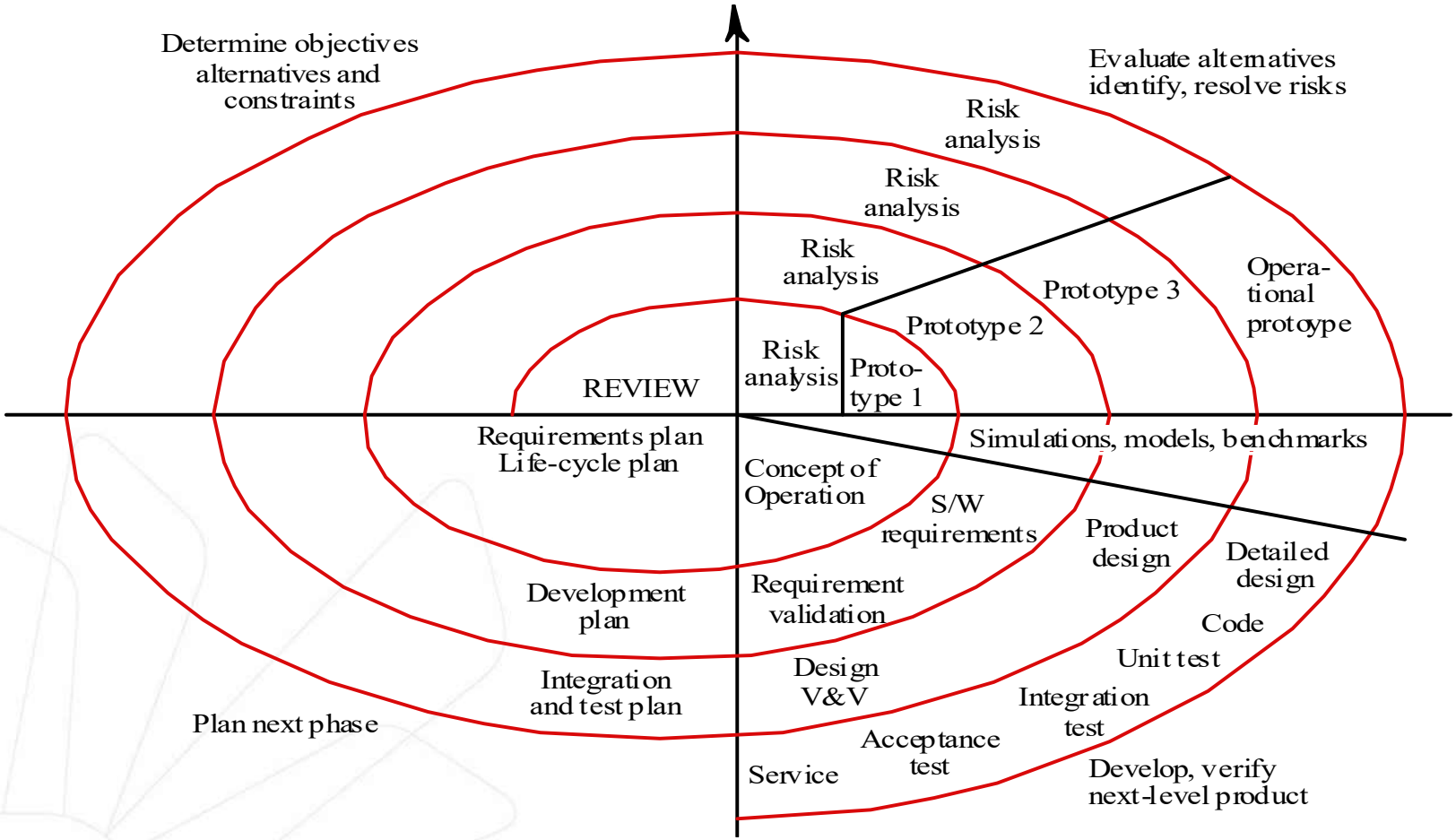
- 개발 사이클이 짧은 환경
  - 빠른 시간 안에 시장에 출시하여야 이윤에 직결
  - 개발 시간을 줄이는 법 – 시스템을 나누어 릴리스



진화적 모델

- 릴리스 구성 방법
  - 점증적 방법 – 기능별로 릴리스
  - 반복적 방법 – 릴리스 할 때마다 기능의 완성도를 높임
- 단계적 개발
  - 기능이 부족하더라도 초기에 사용 교육 가능
  - 처음 시장에 내놓는 소프트웨어는 시장을 빨리 형성시킬 수 있음
  - 자주 릴리스 하면 가동 중인 시스템에서 일어나는 예상하지 못했던 문제를 신속 꾸준히 고쳐나갈 수 있음.
  - 개발 팀이 릴리스마다 다른 전문 영역에 초점 둘 수 있음.

(4) 나선형(spiral) 모델



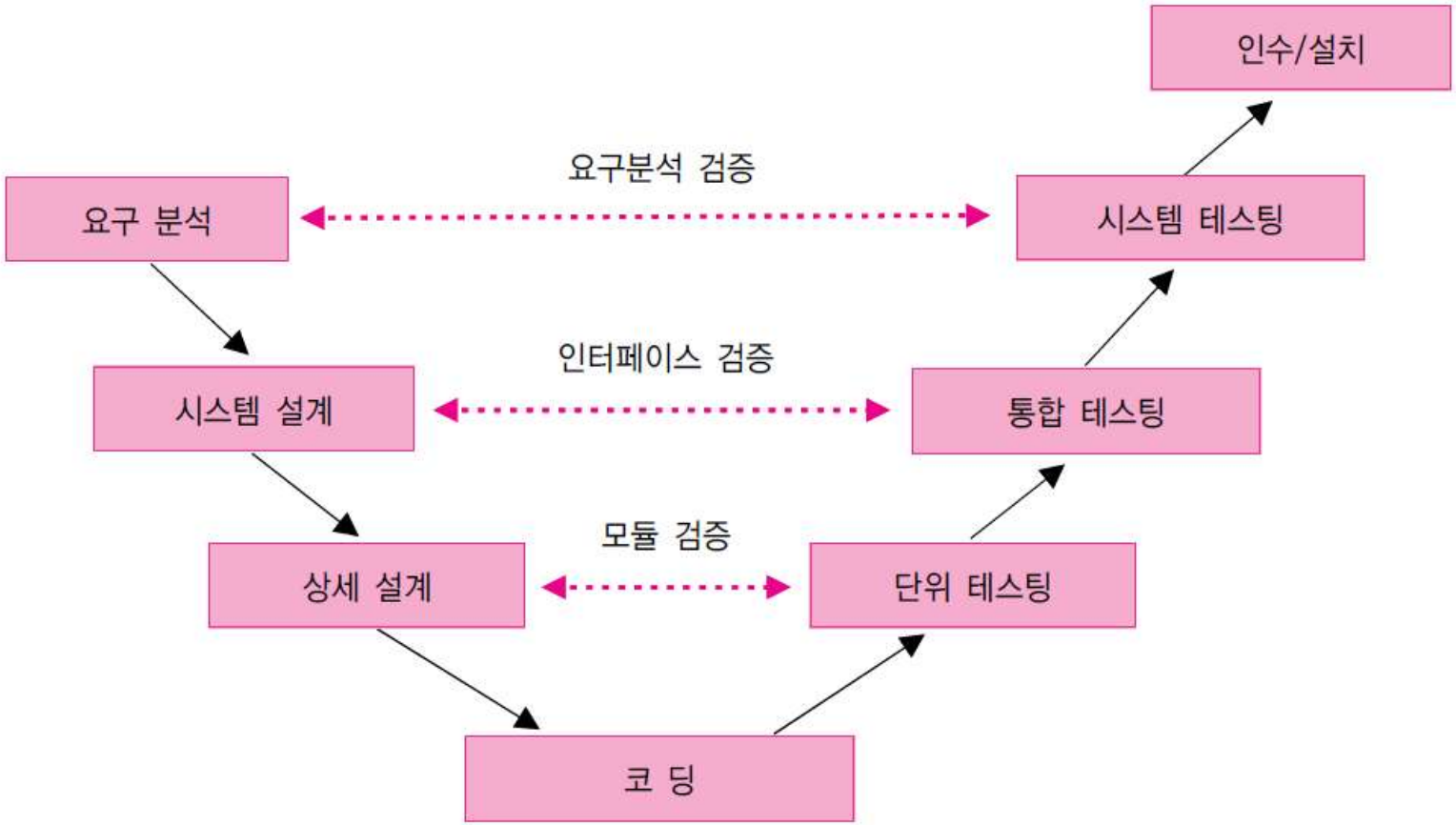
나선형(spiral) 모델

- 소프트웨어의 기능을 나누어 점증적으로 개발
  - 실패의 위험을 줄임
  - 테스트 용이
  - 피드백
- 여러 번의 점증적인 릴리스(incremental releases)
- Boehm이 제안
- 진화 단계
  - 계획 수립(planning): 목표, 기능 선택, 제약 조건의 결정
  - 위험 분석(risk analysis): 기능 선택의 우선순위, 위험요소의 분석
  - 개발(engineering): 선택된 기능의 개발
  - 평가(evaluation): 개발 결과의 평가

나선형(spiral) 모델의 장단점

- 장점
  - 대규모 시스템 개발에 적합 - risk reduction mechanism
  - 반복적인 개발 및 테스트 - 강인성 향상
  - 한 사이클에 추가 못한 기능은 다음 단계에 추가 가능
- 단점
  - 관리가 중요
  - 위험 분석이 중요
  - 새로운 모형
- 적용
  - 재정적 또는 기술적으로 위험 부담이 큰 경우
  - 요구 사항이나 아키텍처 이해에 어려운 경우

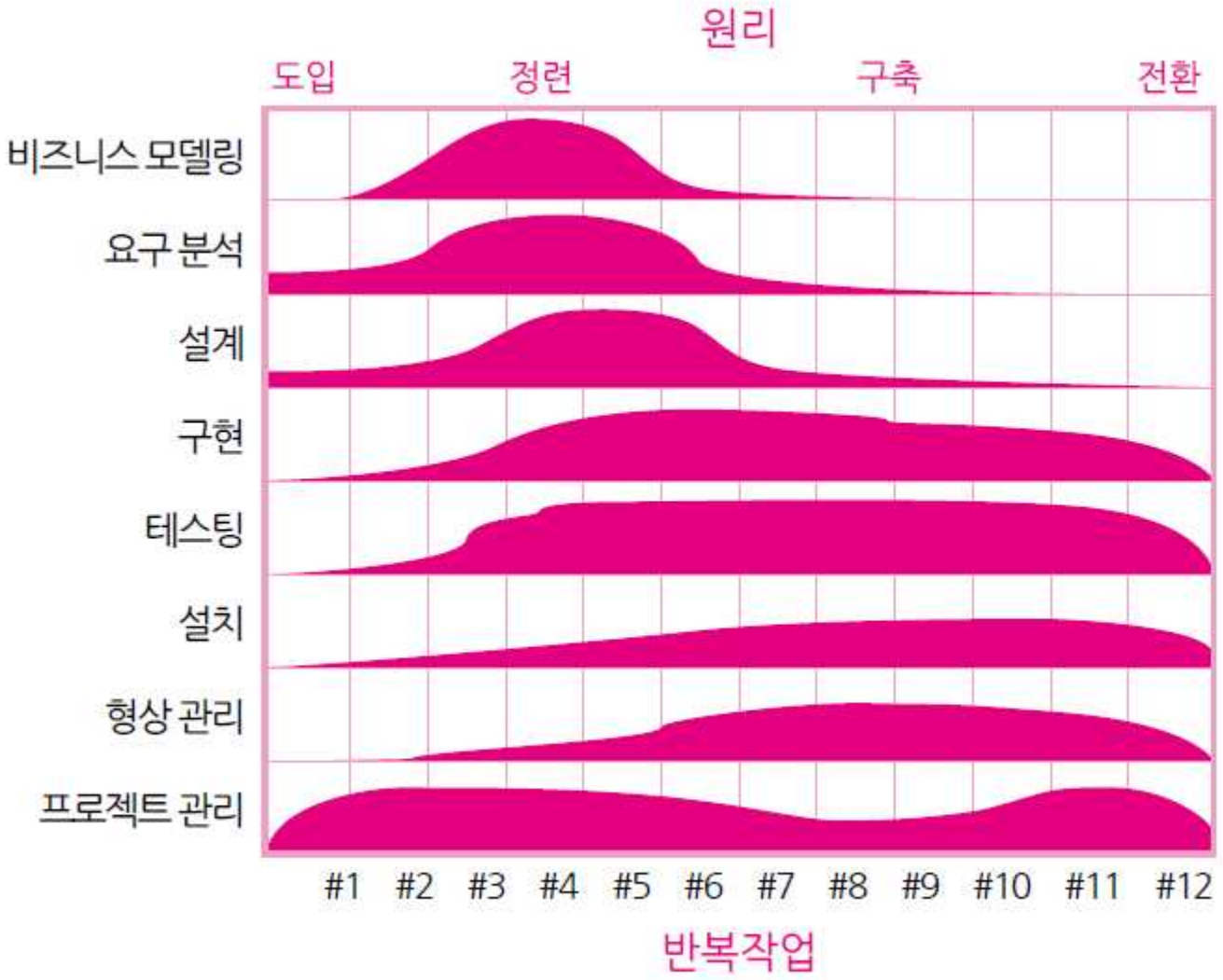
(5) V 모델


인수/설치
요구분석 검증
요구 분석
시스템 테스팅
인터페이스 검증
시스템 설계
통합 테스팅
모듈 검증
상세 설계
단위 테스팅
코 딩

V 모델

- 폭포수 모형의 변형
  - 감추어진 반복과 재 작업을 드러냄
  - 작업과 결과의 검증에 초점
- 장점
  - 오류를 줄일 수 있음
- 단점
  - 반복이 없어 변경을 다루기가 쉽지 않음
- 적용
  - 신뢰성이 높이 요구되는 분야

(6) Unified 프로세스



Unified 프로세스

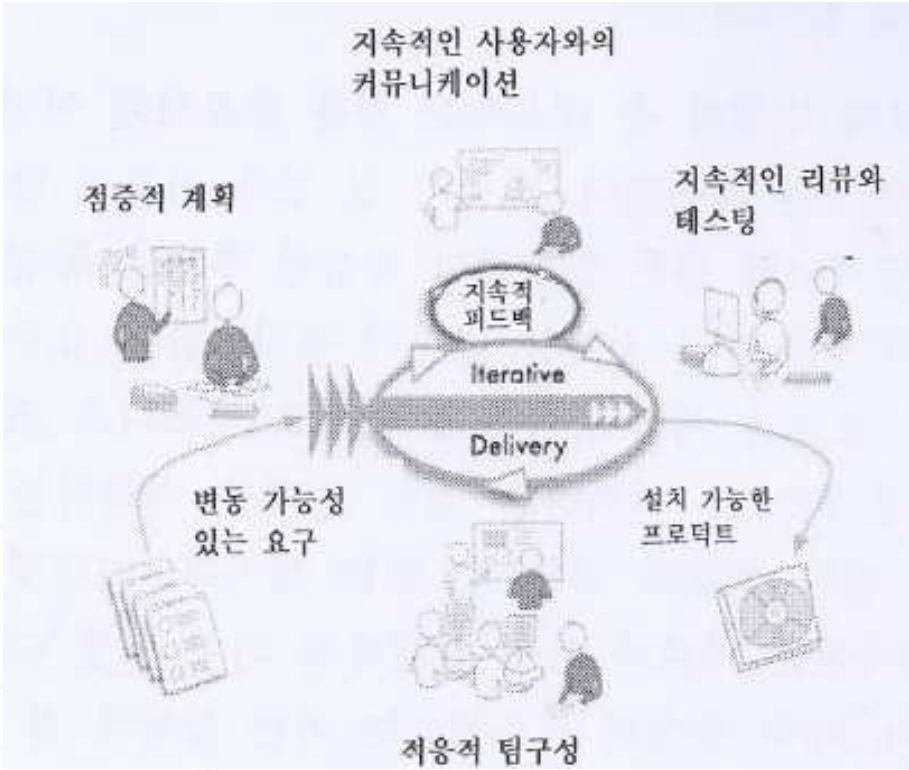
- 사용 사례 중심의 프로세스
- 시스템 개발 초기에 아키텍처와 전체적인 구조를 확정
- 아키텍처 중심
- 반복적이고 점증적

(7) 애자일 프로세스

- 폭포수 프로세스의 단점을 해결
- 절차와 도구보다 개인과 소통을 중요시 한다
- 잘 쓴 문서보다는 실행되는 소프트웨어에 더 가치를 둔다
- 계약 절충보다는 고객 협력을 더 중요하게 여긴다
- 계획을 따라 하는 것보다 변경에 잘 대응하는 것을 중요하게 여긴다

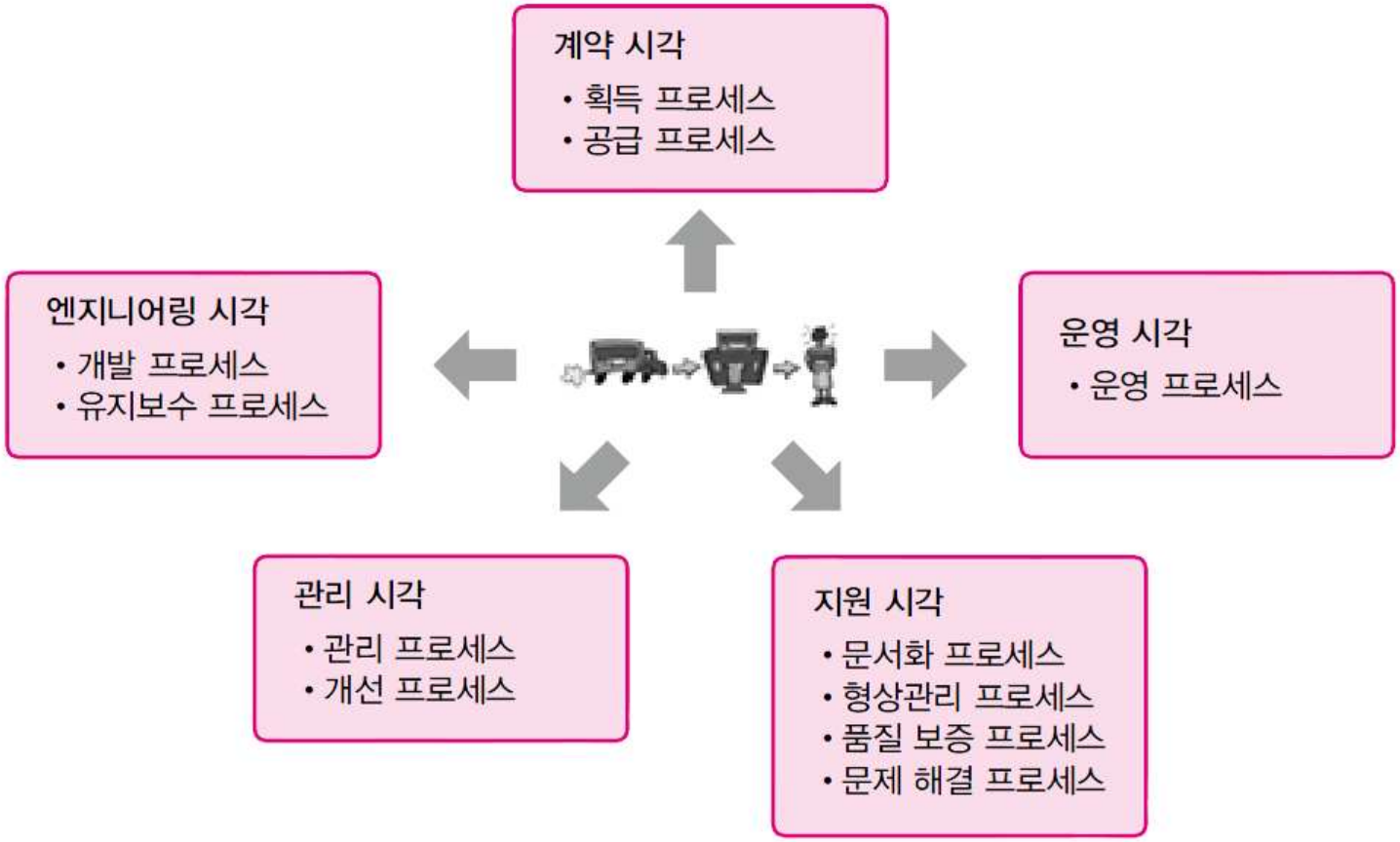
애자일 프로세스

- 사용사례 또는 사용자 스토리나 피처 단위
- 테스트 중심 개발(Test Driven Development)



2.4 지원 프로세스

- **ISO/IEC 12207에서의 프로세스 그룹**



Questions?