

# 2장. 프로세스와 방법론

---

# 목 차

---

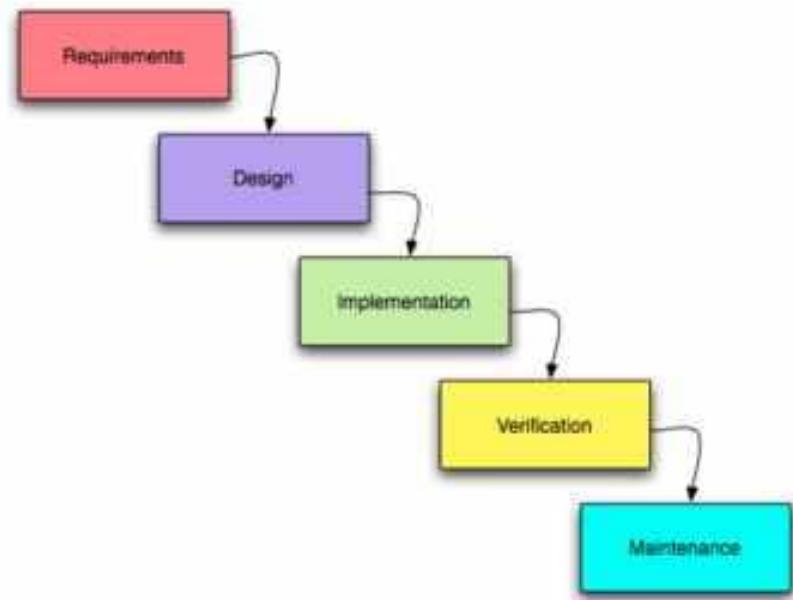
2.1 소프트웨어 생명주기

2.2 프로세스

2.3 프로세스 모델

2.4 지원 프로세스

2.5 방법론



# 여러분들의 지금까지의 개발방법

---

- Code-and-fix
  - 설계하는 작업의 중요성을 깨닫지 못함
  - 계획이 없어 작업 목표가 없음
  - 체계적인 테스트 작업이나 품질 보증 차원의 활동에 대한 필요성의 인식이 없음

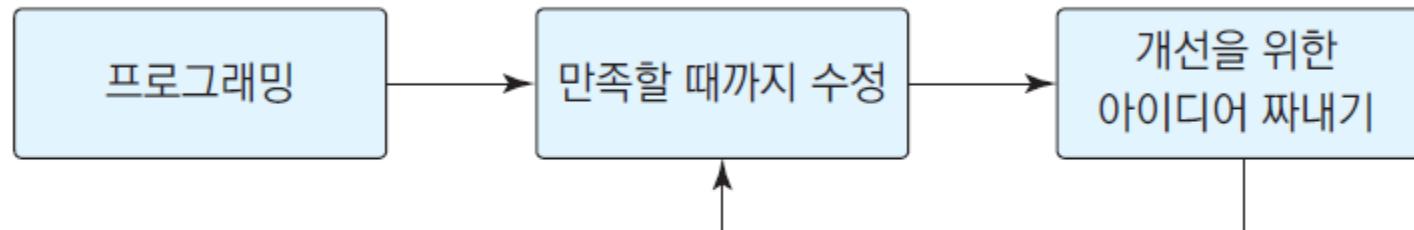


그림 2.1 프로세스 없는 소프트웨어 개발

# 프로세스와 방법론의 비교

	프로세스	방법론
특징	<ul style="list-style-type: none"><li>• 단계적인 작업의 틀을 정의한 것</li><li>• 무엇을 하는가에 중점</li><li>• 결과물이 표현에 대하여 언급 없음</li><li>• 패러다임에 독립적</li><li>• 각 단계가 다른 방법론으로도 실현 가능</li></ul>	<ul style="list-style-type: none"><li>• 프로세스의 구체적인 구현에 이름</li><li>• 어떻게 하는가에 중점</li><li>• 결과물을 어떻게 표현하는지 표시</li><li>• 패러다임에 종속적</li><li>• 각 단계의 절차, 기술, 가이드라인을 제시</li></ul>
사례	<ul style="list-style-type: none"><li>• 폭포수 프로세스</li><li>• 나선형 프로세스</li><li>• 프로토타이핑 프로세스</li><li>• Unified 프로세스</li><li>• 애자일 프로세스</li></ul>	<ul style="list-style-type: none"><li>• 구조적 분석, 설계 방법론</li><li>• 객체지향 방법론</li><li>• 컴포넌트 방법론</li><li>• 애자일 방법론</li></ul>

# 소프트웨어 프로젝트, 프로세스 명세와 프로세스 모델

---

- 소프트웨어 프로젝트
  - 수행할 작업을 조직화한 프로세스를 이용
  - 비용, 일정, 품질에 대한 목표를 성취하는 것
- 프로세스 명세
  - 프로젝트에서 수행하여야 하는 작업과 이들의 수행순서를 정의
  - 실행프로세스는 다를 수 있음
- 프로세스 모델
  - 일반적인 프로세스를 기술한 것
  - 작업의 단계와 순서
  - 각 단계 작업수행의 제약사항이나 조건을 모아 놓은 것

## 2.1 소프트웨어 생명주기

- SDLC : Software Development Life Cycle
- 소프트웨어 개발 프로세스는 소프트웨어가 탄생되어 운용되고 유지보수를 거쳐 폐기되기까지 전 과정을 의미
- 인간의 탄생과 죽음에 이르기까지의 과정과 유사

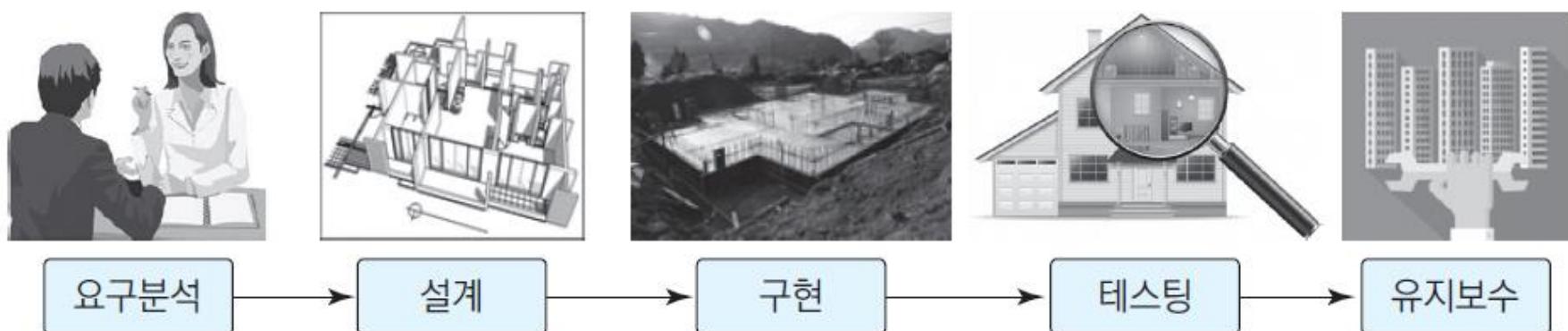
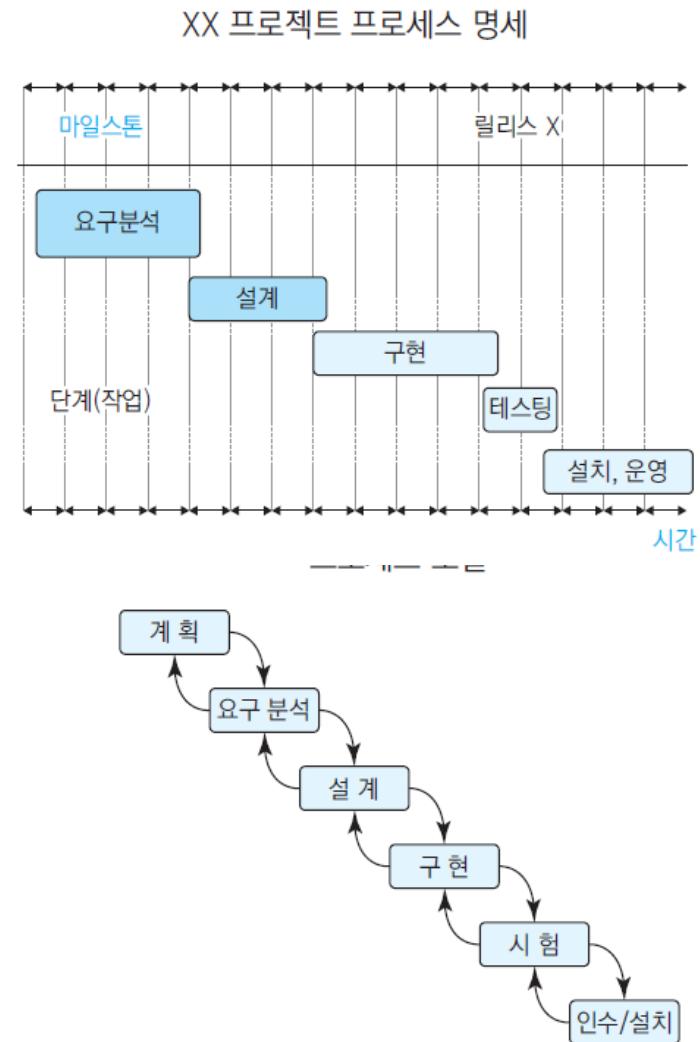


그림 2.2 소프트웨어 생명주기

## 2.2 프로세스

- **프로세스**

- 소프트웨어 시스템을 구축하기 위하여 수행되는 작업의 단계
- 소프트웨어 개발에 대한 기술적, 관리적 이슈를 다루는 작업
- 개발 모델별 컴포넌트 프로세스, 부프로세스 존재
- 서로 다른 목적
- 서로 협력하여 전체 목적을 만족



# 프로세스

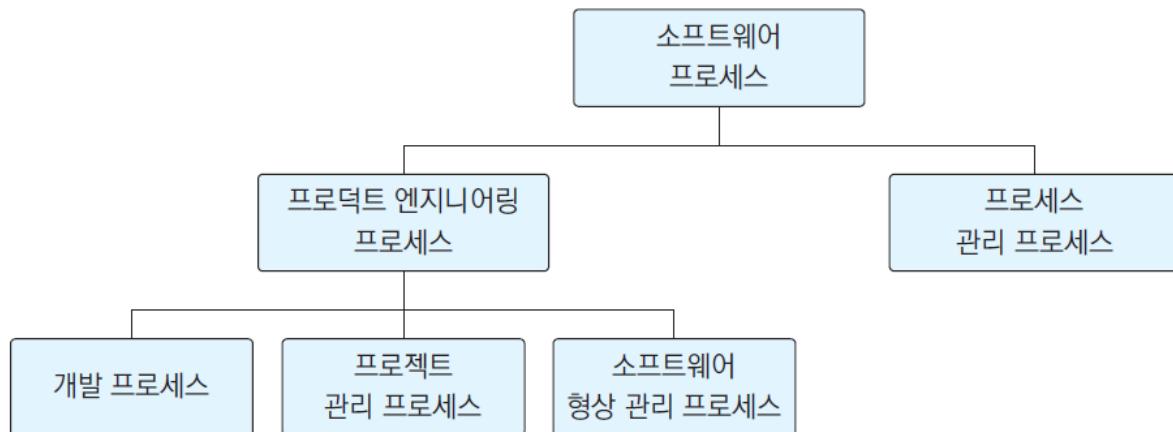
- 어떤 일을 하기 위한 특별한 방법으로 단계나 작업으로 구성됨
  - 작업 결과와 검증 조건을 명확히 정의하여야 함
  - 작업 방법
  - 진입 조건, 출구 조건



그림 2.5 개발 프로세스의 정의

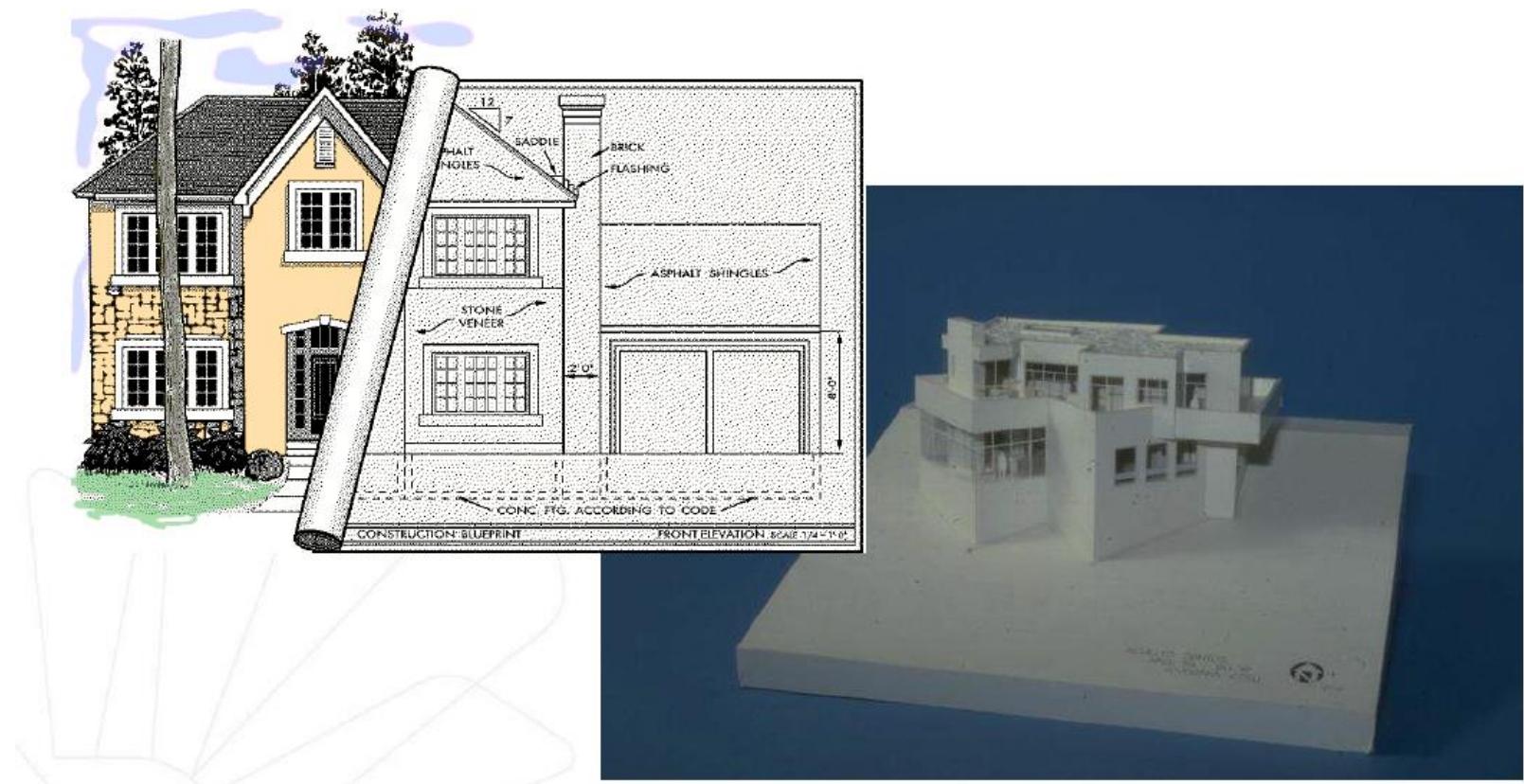
# 프로세스의 종류

- 프로젝트의 중심 프로세스
  - 개발 프로세스
  - 관리 프로세스
- 기타 프로세스
  - 형상 관리 프로세스 : 품질보증팀(QA)
  - 프로세스 관리 프로세스 : 프로세스관리그룹(SEPG)



# 소프트웨어 공학과 유사한 작업

- 건물의 건축



# 소프트웨어 개발 프로세스

---

- 계획
- 요구분석
- 설계
- 구현
- 통합과 테스트
- 설치와 유지보수

# 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

# 요구분석

---

- 요구 – 시스템이 가져야 할 능력(capability)과 조건(condition)
- What의 단계
- 응용분야(도메인)에 집중
- 가장 중요하고도 어려운 단계
  - 조그만 차이가 큰 오류로 변함
- 결과물 : 요구분석서  
(SRS: Software Requirement Specification)

# 설계

---

- How의 단계
- 솔루션에 집중
- 아키텍처 설계
- 데이터베이스 설계
- UI 설계
- 상세 설계
- 결과물: 설계명세서

(SDD/SDS:Software Design Document/Specification)

# 구현

---

- 'Do it' 단계
- 코딩과 단위 테스트
- 설계 또는 통합 단계와 겹치기도 함
  - 전체 일정을 줄이기 위하여
  - 협력 작업이 필요한 경우
- 특징
  - 스트레스 증가
  - 최고의 인력 투입

# 구현

---

- 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

# 좋은 프로세스의 특성

- 예측 가능성

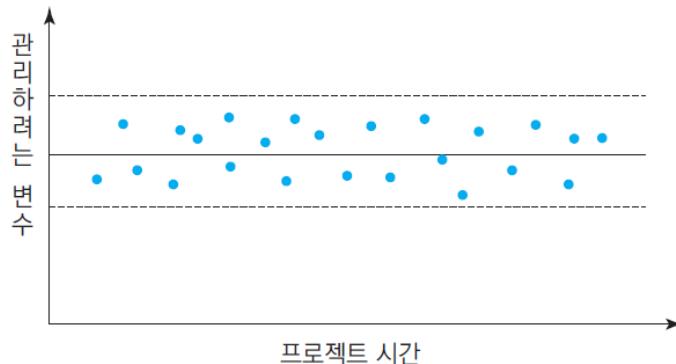


그림 2.6 통계에 근거한 프로세스 제어

- 테스팅과 유지보수 용이성

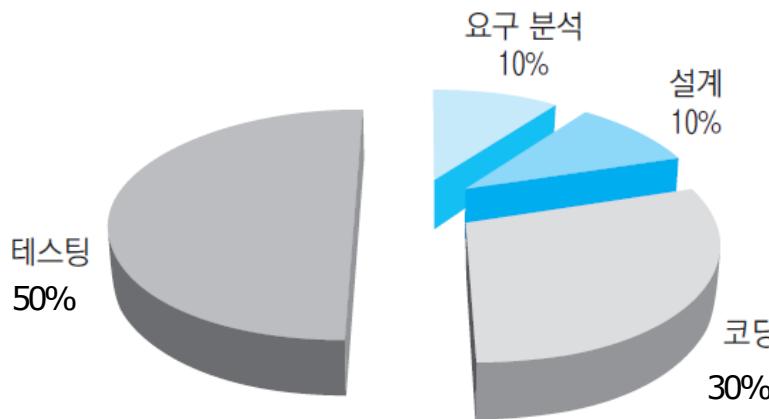
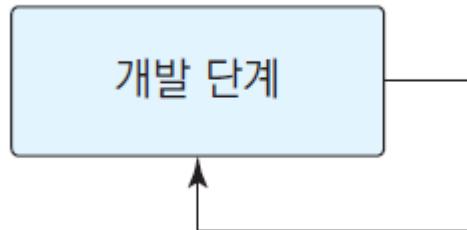


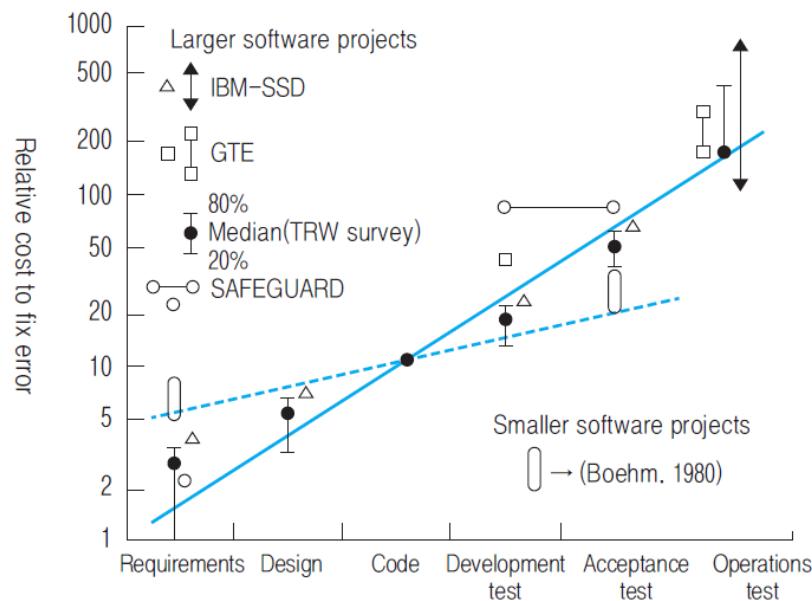
그림 2.7 프로젝트 각 단계의 비용 분포

# 좋은 프로세스의 특성

- 변경 지원 - 변경을 쉽게 다룰 수 있는 프로세스



- 결함 제거

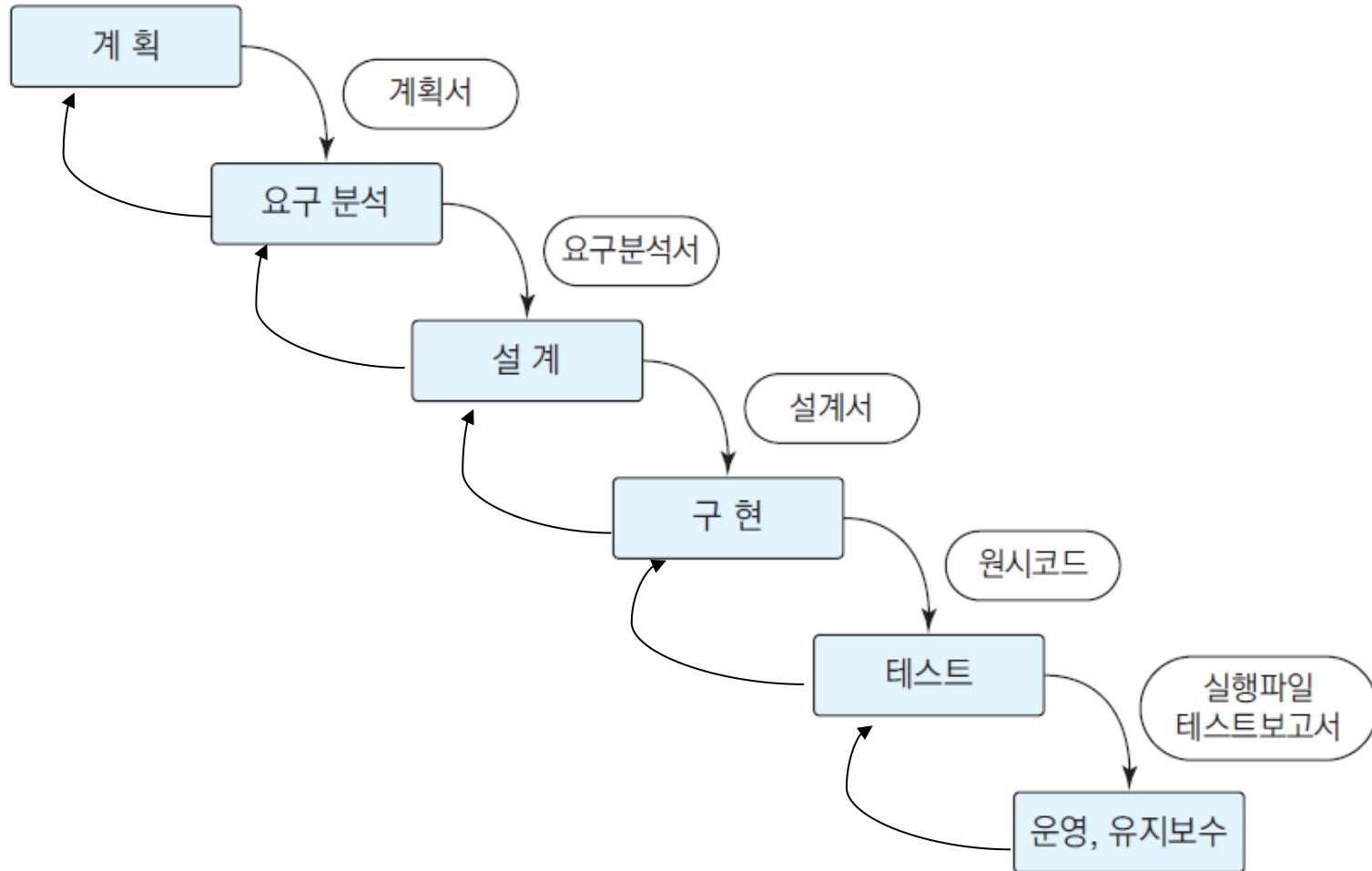


## 2.3 프로세스 모델

---

- 프로세스 모델
  - 일반적인 모델이 될만한 프로세스를 기술한 것
- 대표적인 프로세스 모델
  - 폭포수 모델
  - 프로토타이핑 모델
  - 나선형 모델
  - 진화적 모델
  - Unified Process
  - 애자일 프로세스

# 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

# 폭포수 모형의 장단점

---

- 장점
  - 프로세스가 단순하여 초보자가 쉽게 적용 가능
  - 중간 산출물이 명확, 관리하기 좋음
  - 코드 생성 전 충분한 연구와 분석 단계
- 단점
  - 소용 없는 다종의 문서를 생산할 가능성 있음
  - 애매한 부분이 남아 있거나 프로세스 진행 과정에 변경될 수 있는데 이를 수용할 수 없다.
  - 테스트 작업이 프로젝트 후반, 즉 시스템이 완성된 후에 시작됨
- 적용
  - 이미 잘 알고 있는 문제나 연구 중심 문제에 적합
  - 변화가 적은 프로젝트에 적합

# V 모델

- 검증을 강화하는 관점에서 폭포수 모델을 확장한 모델

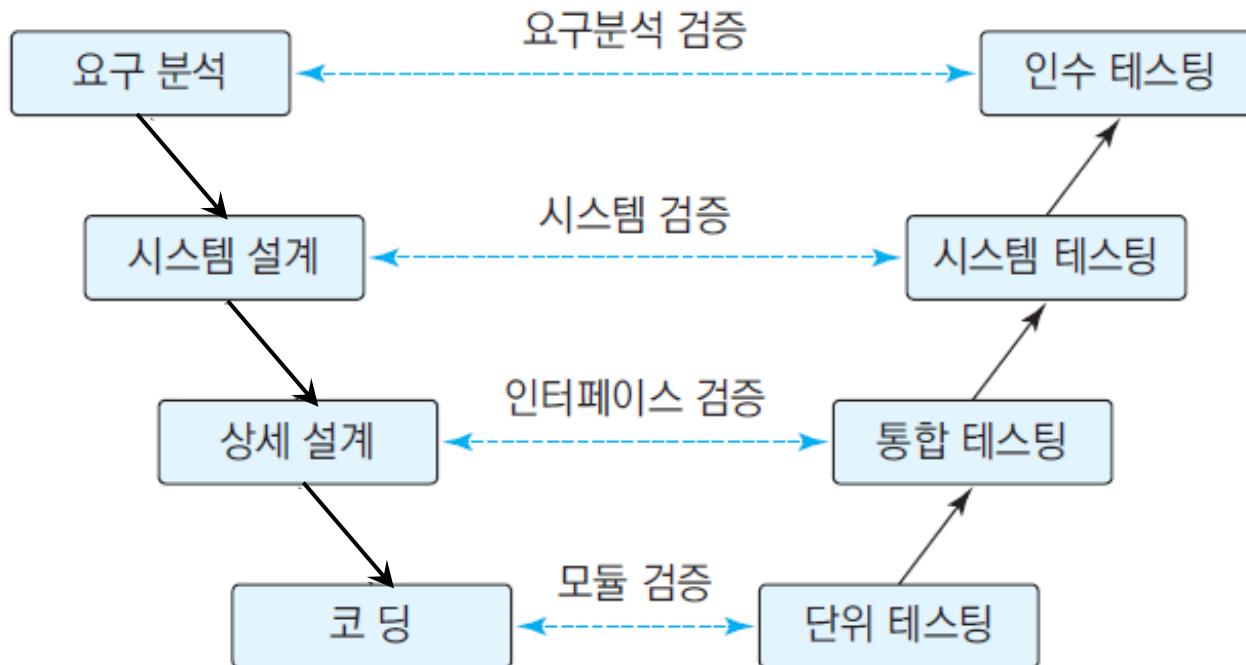


그림 2.11 V 모델

# 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

# 프로토타이핑 모델

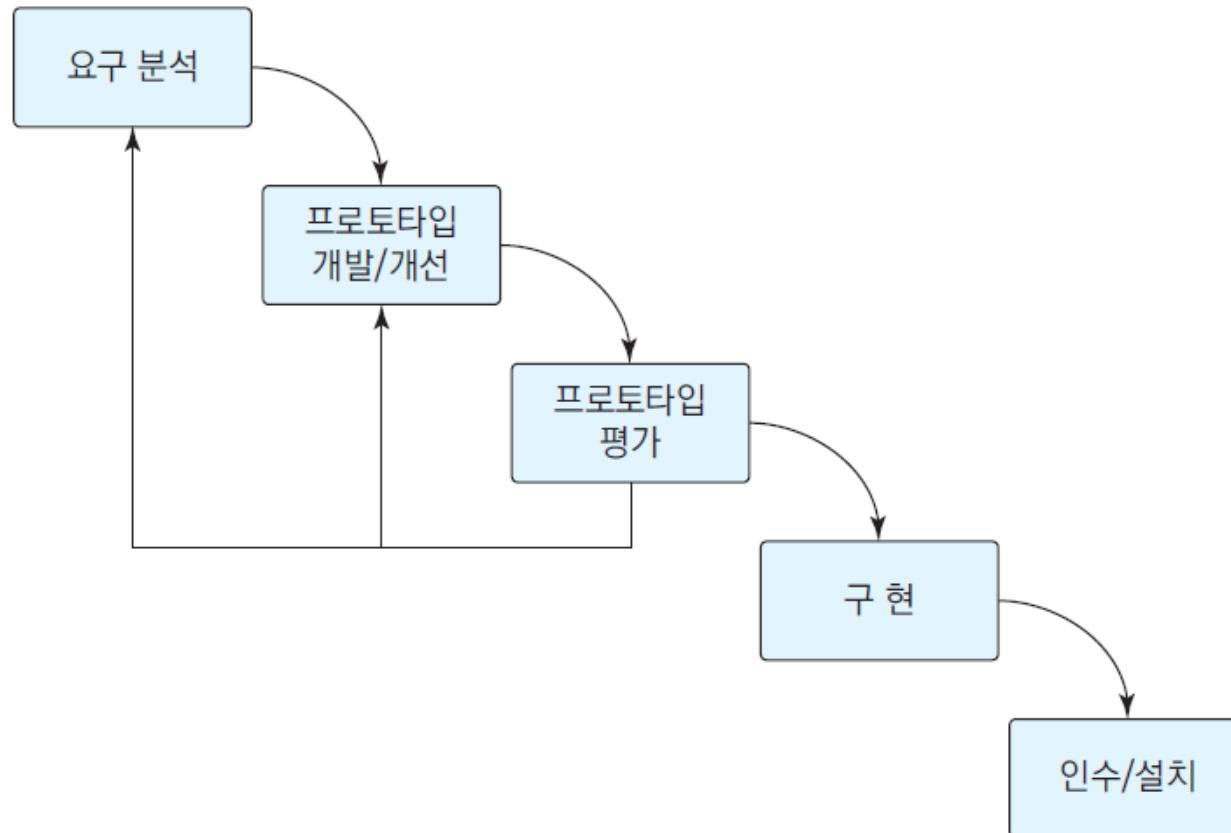


그림 2.13 프로토타이핑 모델

# 프로토타이핑 모델

---

- 프로토타이핑
  - 요구 사항에 대한 피드백을 받기 위해 시스템을 실험적으로 만들어 사용자에게 보여주고 평가하게 하는 방법
- 프로토타이핑 도구
  - 시스템의 작동을 시뮬레이션 하여 사용자가 볼 수 있는 반응을 보여줌
- 공동의 참조 모델
  - 사용자와 개발자의 의사소통을 도와주는 좋은 매개체
- 프로토타입의 목적
  - 단순한 요구 추출 – 만들고 버림
  - 제작 가능성 탐진 - 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

# 진화적 모델

- 개발 사이클이 짧은 환경
  - 빠른 시간 안에 시장에 출시하여야 이윤에 직결
  - 개발 시간을 줄이는 법 – 시스템을 나누어 릴리스
- 릴리스 구성 방법
  - 점증적 방법 – 기능별로 릴리스
  - 반복적 방법 – 릴리스 할 때마다 기능의 완성도를 높임

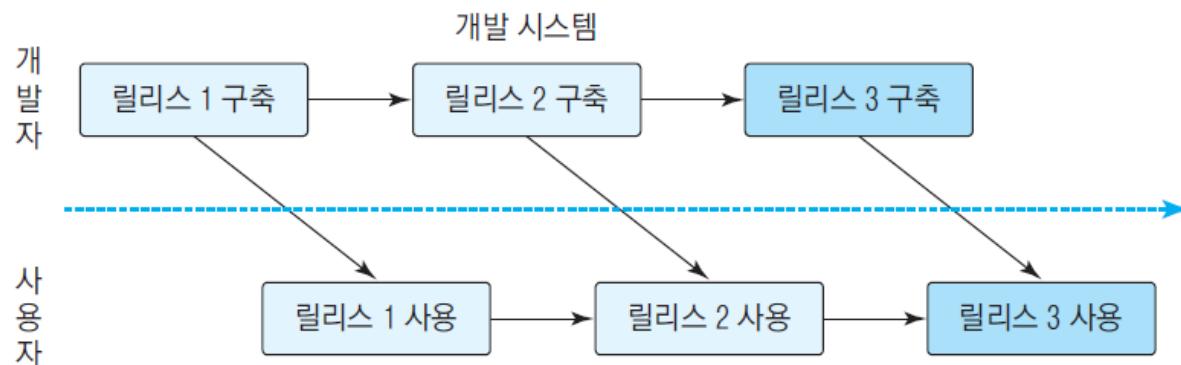


그림 2.15 진화적 모델

# 진화적 모델의 장단점

---

- 장점
  - 몇 가지 기능이 부족하더라도 초기에 사용 교육 가능
  - 사용자의 요구를 빠르게 반영
  - 새로운 기능을 가진 소프트웨어에 대한 시장을 빨리 형성
  - 가동 중인 시스템에서 일어나는 예상하지 못했던 문제를 신속하고 꾸준하게 고쳐 나갈 수 있음
- 단점
  - 프로젝트 관리가 복잡해지기 때문에 작은 프로젝트 부적합
  - 끝이 안보일 수 있어 실패의 위험이 커짐
  - 프로젝트의 진행이 위험 분석에 크게 의존

# 나선형(spiral) 모델

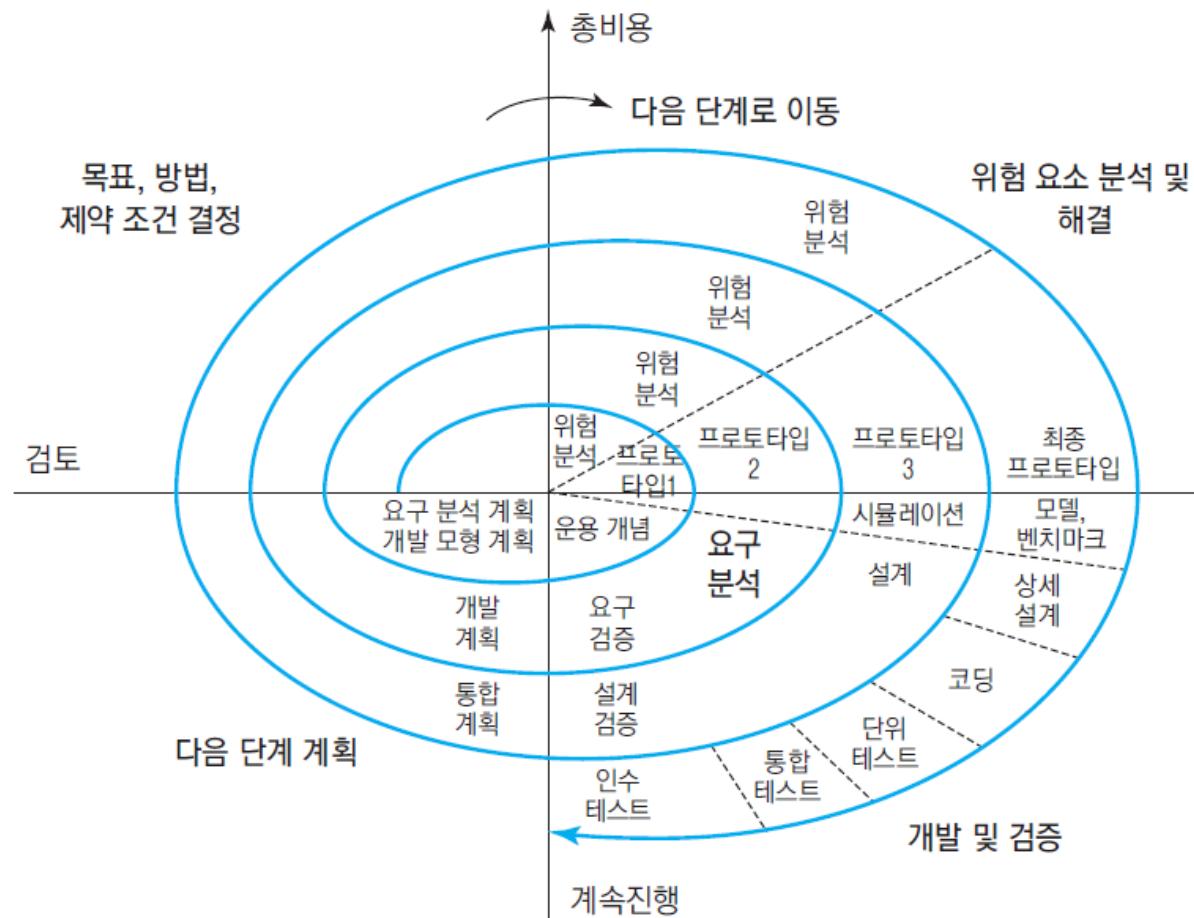


그림 2.14 나선형 모델

# 나선형(spiral) 모델

---

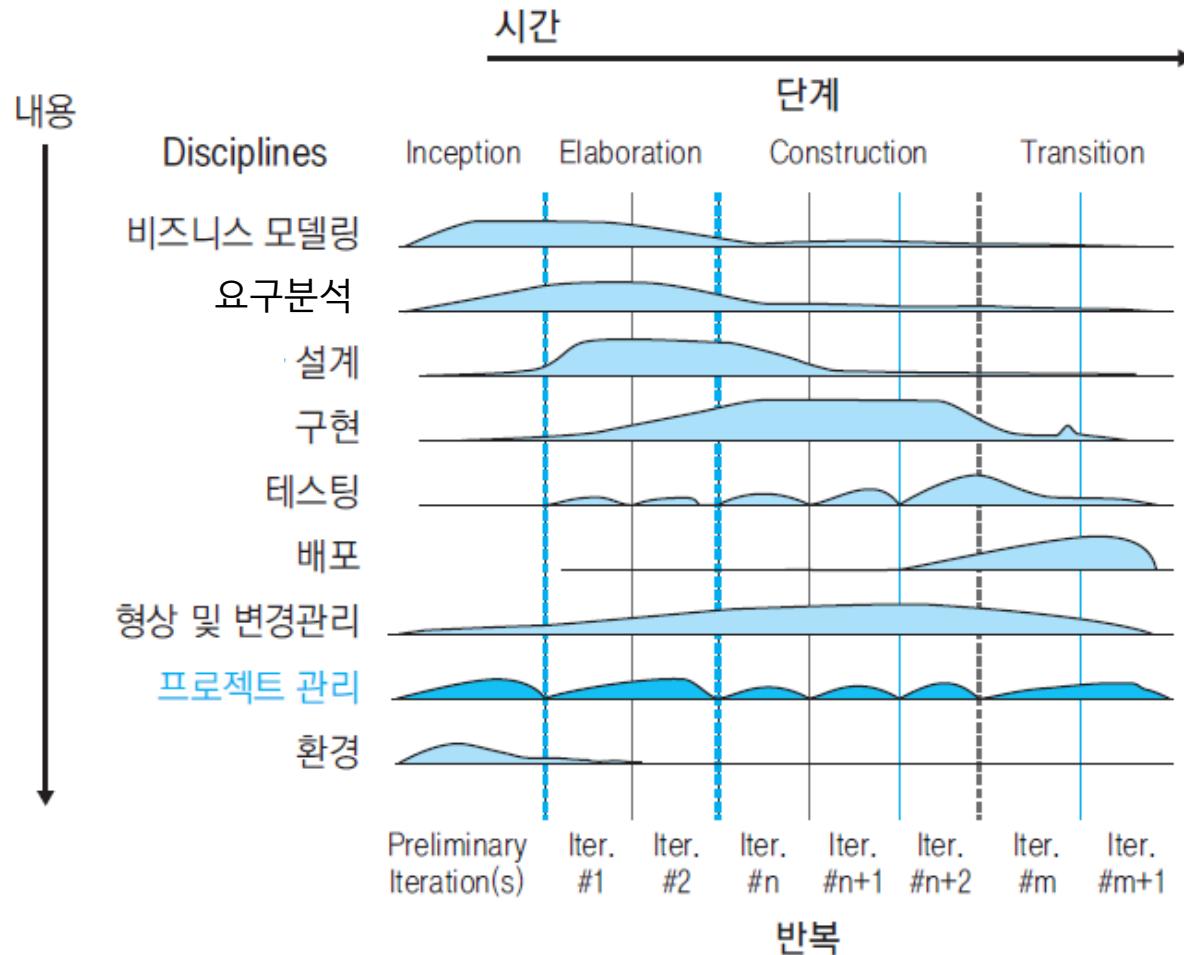
- 소프트웨어의 기능을 나누어 점증적으로 개발
  - 실패의 위험을 줄임
  - 테스트 용이
  - 피드백
- 여러 번의 점증적인 릴리스(incremental releases)
- Boehm이 제안
- 반복 순환 단계
  - ① 목표, 방법, 제약 조건 결정
  - ② 위험 요소 분석 및 해결
  - ③ 개발과 평가
  - ④ 다음 단계의 계획

# 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 복잡
  - 위험 분석을 잘못하여 지나친 경우 피해가 큼
  - 성공 사례가 많이 알려지지 않음
- **적용**
  - 재정적 또는 기술적으로 위험 부담이 큰 경우
  - 요구 사항이나 아키텍처 이해에 어려운 경우

# Unified 프로세스



# Unified 프로세스

---

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

# Unified 프로세스

---

## ① 도입(inception)

- 1, 2회 정도 반복으로 도입 단계를 진행
- 간단한 유스케이스 모델과 소프트웨어 구조, 프로젝트 계획을 작성

## ② 정련(elaboration)

- 여러 번의 반복 과정으로 이루어짐
- 대부분의 유스케이스를 작성
- 아키텍처 설계

## ③ 구축(construction)

- 남아 있는 유스케이스에 대하여 구현하고 통합
- 시스템을 목표 환경에 점증적으로 설치

## ④ 전환(transition)

- 시스템을 배치, 사용자를 교육
- 베타 테스팅, 결함 수정, 기능 개선

# Unified Process의 장단점

---

- 장점
  - 방법론과 프로세스가 잘 문서화되어 있어 교육받기 좋음
  - 고객의 요구 변경과 관련된 리스크를 적극적으로 해결
  - 통합을 위한 노력과 시간을 줄일 수 있음
  - 쉽고 빠르게 코드를 재사용
- 단점
  - 프로세스가 너무 복잡, 이해하기 어렵고 정확히 적용하기가 어려움
  - 소프트웨어 프로젝트 참여자들의 협동, 의사소통에 대한 가이드가 없음
  - 조직화되지 않은 개발로 완전히 알려지지 않은 형태의 소프트웨어 개발로 이어질 수 있음

# 애자일 프로세스

---

- 폭포수 프로세스의 단점을 해결
- 절차와 도구보다 개인과 소통을 중요시 함
- 1~4주간의 짧은 주기로 개발을 반복
- 실행되는 소프트웨어를 개발하여 단계적으로 시스템 전체를 완성 – 테스트중심 개발

# 애자일 프로세스

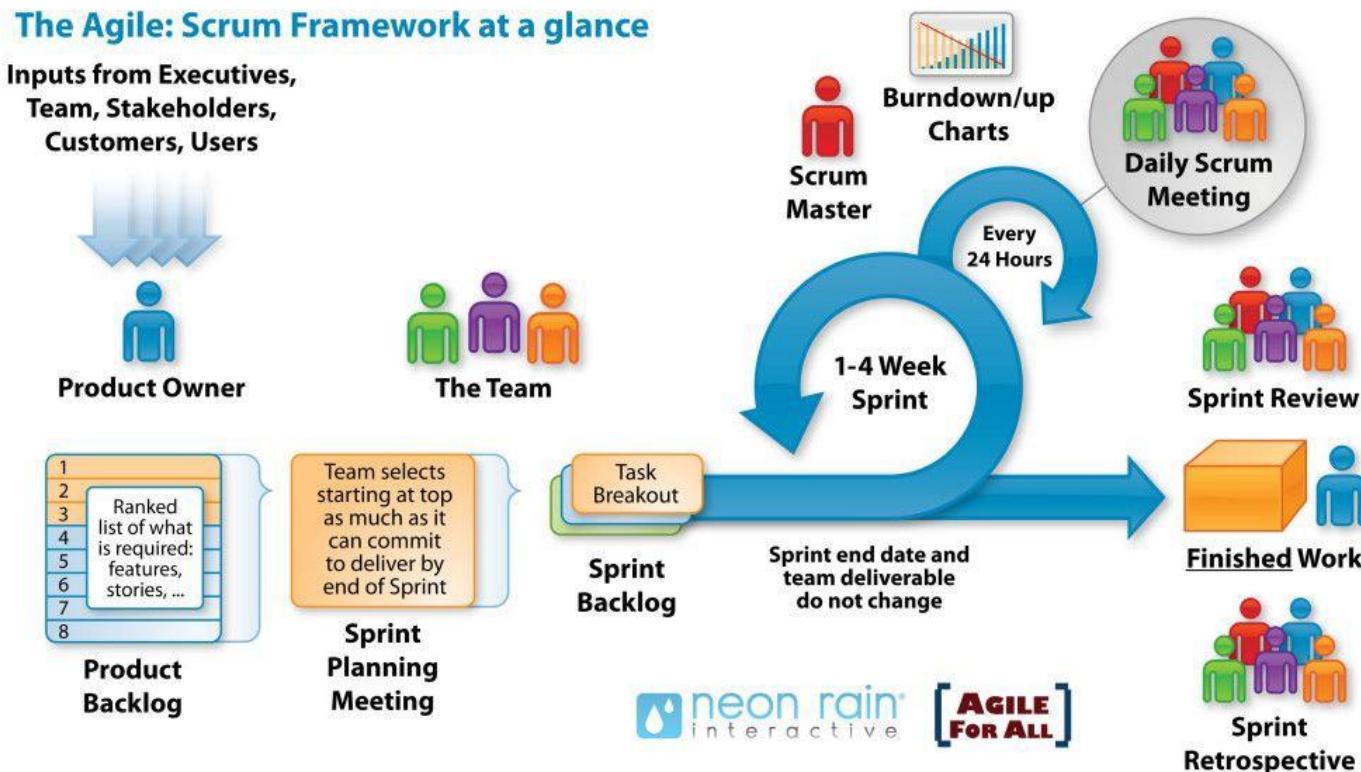
---

- 애자일 선언
  - 1) 프로세스나 도구 보다는 개인과 상호 작용을,
  - 2) 포괄적인 문서보다는 작동하는 소프트웨어를,
  - 3) 계약에 대한 협상보다는 고객과의 협력을,
  - 4) 계획을 고수하기 보다는 변화에 대응을

더욱 가치 있게 여긴다.

# 스크럼

- 개발 팀원 모두가 함께 소통하고 협력하여 짧은 주기를 반복하며 소프트웨어를 개발하는 작업, 역할, 결과물
- 백로그를 정하고 여기에 우선순위를 부여
- 짧은 주기(스프린트)



# 스크럼 역할

---

- **스크럼 마스터**
  - 프로세스 순조롭게 진행되도록 만들고
  - 생산성에 영향주는 장애물 제거
  - 스크럼 이벤트(meeting) 주재
- **Product owner**
  - 백로그 정의
  - 백로그 아이템 순서 정하기
  - 팀의 백로그 아이템을 잘 이해하고 수행할 수 있도록 투명하게 관리하고 최적화
- **팀원**
  - Self-organizing, cross-functional
  - 5~9 명
  - 매일 가까운 거리에서 함께 일하여야

# 스크럼 프로세스

---

- 스프린트 (Sprint)
  - 달력기준 1~4주 단위의 반복개발기간
- 3가지 미팅
  - 스프린트 계획(Sprint Meeting)
  - 일일 스크럼(Daily Scrum)
  - 스프린트 리뷰(Sprint Review)
- 3가지 산출물
  - 제품 백로그(Product backlog)
  - 스프린트 백로그(Sprint Backlog)
  - 소멸 차트(Burndown chart)

# 스프린트 계획

---

- 스프린트 동안 수행할 스프린트 목표와 이를 이루기 위한 작업 상세 내역을 정함
- 작업 상세 내역은 스프린트 동안 수행할 작업을 목록화한 스프린트 백로그로 정리
- 스프린트 내 완료기준을 정하고 팀원들과 공유



# 백로그

- 스프린트 목표를 달성하기 위한 작업 목록



# 백로그

프로젝트: Agile 팀 프로젝트 서버: http://demo2010a8080/tfs/connectorcollectionone 쿼리: 반복 백로그...					
ID	작업 항목	제목 1	제목2	상태	담당자
22	사용자 스토리	고객으로서 ...에 로그인할 수 있습니다.		활성	Jeff Hay
30	작업	UI 대화 상자		활성	Alan Brewer
31	작업	사용자 흐름		활성	Alan Brewer
32	작업	단위 테스트		활성	Alan Brewer
33	작업	백엔드 변경 내용		활성	Alicia Thomber
34	작업	스키마 변경 내용		활성	Alicia Thomber
23	사용자 스토리	고객으로서 ...에서 로그아웃할 수 있습니다.		활성	Jeff Hay
35	작업	설정 저장		활성	Alan Brewer
36	작업	사용 권한 확인		활성	Alan Brewer
37	작업	단위 테스트		활성	Chen Yang
38	작업	스키마 변경 내용		활성	Chen Yang
39	작업	API 설정		활성	Chen Yang
24	사용자 스토리	고객으로서 내 ...에 추가할 수 있습니다.		활성	Jeff Hay
40	작업	쇼핑 카트 아이콘...		활성	Cassie Hicks
41	작업	카트 만들기		활성	Cassie Hicks
42	작업	항목 추가 및 제거		활성	Cassie Hicks
43	작업	단위 테스트		활성	Cassie Hicks

# 일일미팅(Daily Scrum)

---

- Scrum Master가 주재하여 매일 15분 정도 논의
- 지난 스크럼 미팅 이후 수행한 작업, 오늘부터 다음 스크럼 미팅까지 수행할 작업
- 문제 있는 부분에 대한 토론

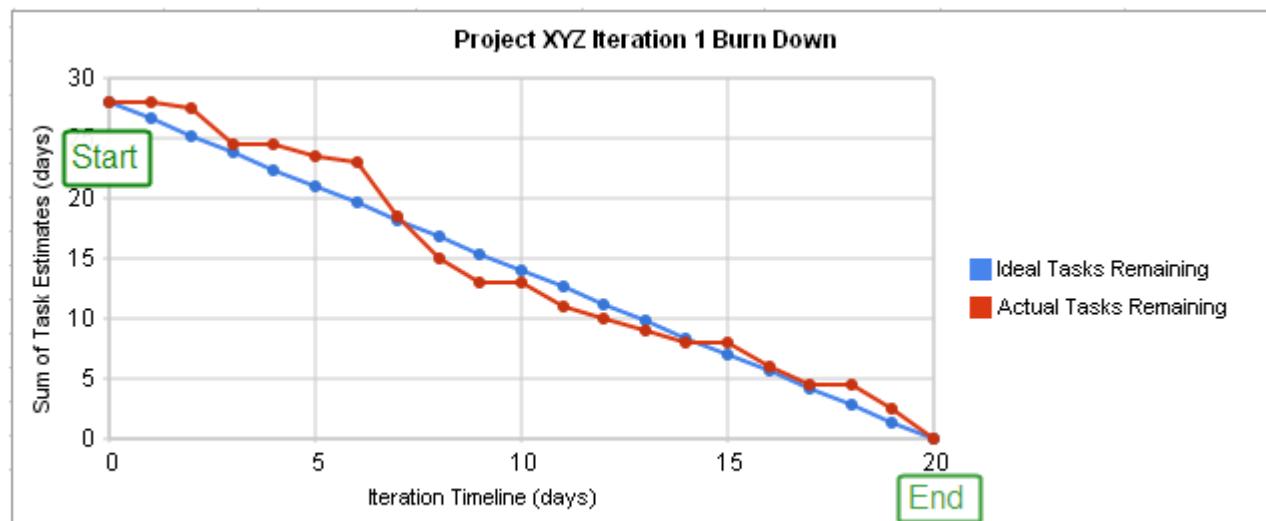


# The War Room



# Burndown Chart

- 매일 팀이 완료할 작업이 얼마나 남았는지에 대한 새 추정치를 보여줌
- 이상적으로 이 차트는 스프린트의 마지막 날에 “남은 작업이 0”이 되는 방향의 감소 그래프



소멸 차트 (출처 : [https://en.wikipedia.org/wiki/Burn\\_down\\_chart](https://en.wikipedia.org/wiki/Burn_down_chart))

# 스프린트 리뷰

---

- 릴리스될 Increment에 대한 발표
- 스프린트의 결과를 이해하고 백로그에 대한 변경 필요성 파악
- 회의 방법
  - Project owner가 완성된 백로그와 완성되지 않은 백로그 설명
  - 팀은 완성된 작업을 설명하고 문제점 토의, 질문과 답변
  - 다음 스프린트에서 무엇을 할 것인지 토의



# 스프린트 회고(Retrospective)

---

- 목적
  - 지난 스프린트를 통하여 학습하기 위함(사람, 관계, 프로세스, 도구)
  - 잘된 것 잠재적인 개선 아이템을 찾아내기 위함
  - 품질 향상을 위한 계획 수립
- 스크럼 프로세스를 개선시킬 기회
- <http://scrumtrainingseries.com/>

# 사용자 스토리

---

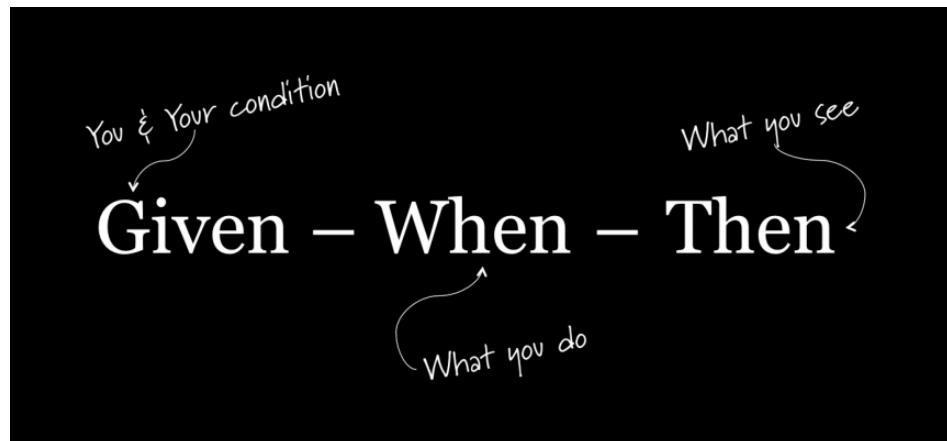
- 템플릿
  - [사용자/역할] 는
  - [목표/혜택/이익]를 얻기 위하여
  - [행위/작업] 을 원한다.
- 예
  - [고객] 은
  - [현찰을 사용하기] 위하여
  - [ATM에서 현금을 인출하기] 를 원한다.

# 사용자 스토리 인수조건

---

- **인수 조건(Acceptance Criterion) 1:**

- Given 계좌에 잔고가 충분하고
- And 카드가 유효하며
- And 인출기에 현금이 있다면,
- When 고객이 현금 인출을 요청할 때
- Then 계좌에서 인출이 가능한다면
- And 현금이 인출되고
- And 카드가 반납됨을 확인한다.



# 스크럼 진행순서

---

1. PO는 제품의 요구기능(User Story)과 우선순위를 제품 백로그로 정한다.
2. PO가 정한 제품의 우선순위에서 어디까지 작업을 할지 팀과 조율 한다.
3. 스프린트 목표를 구현 가능하도록 팀에서 스프린트 백로그를 작성한 뒤 작업을 할당한다.
4. 스프린트를 진행하는 동안, 매일 정해진 장소와 시간에 모든 개발 팀원이 참여하는 일일 스크럼 회의를 가진다.
5. 매회의 스프린트가 종료할 때마다, 스프린트 리뷰(Review)를 통해 만들어진 제품을 검토하고 개선사항을 이해 한다.
6. 제품의 리뷰를 통해 제품의 지속적 개선사항 도출이 끝나면, 스프린트 회고(Retrospective)를 통해 팀의 개발 문화(프로세스)에 대한 개선의 시간을 갖는다.
7. 다음 스프린트에서 수행할 백로그를 PO와 필요 인원이 모여 선정하고 계획하는 시간을 갖는다.

# 익스트림 프로그래밍

- 사용자 스토리
- 매일 빌드와 통합
- 테스트 주도 개발(Test-Driven Development)
- 짹 프로그래밍

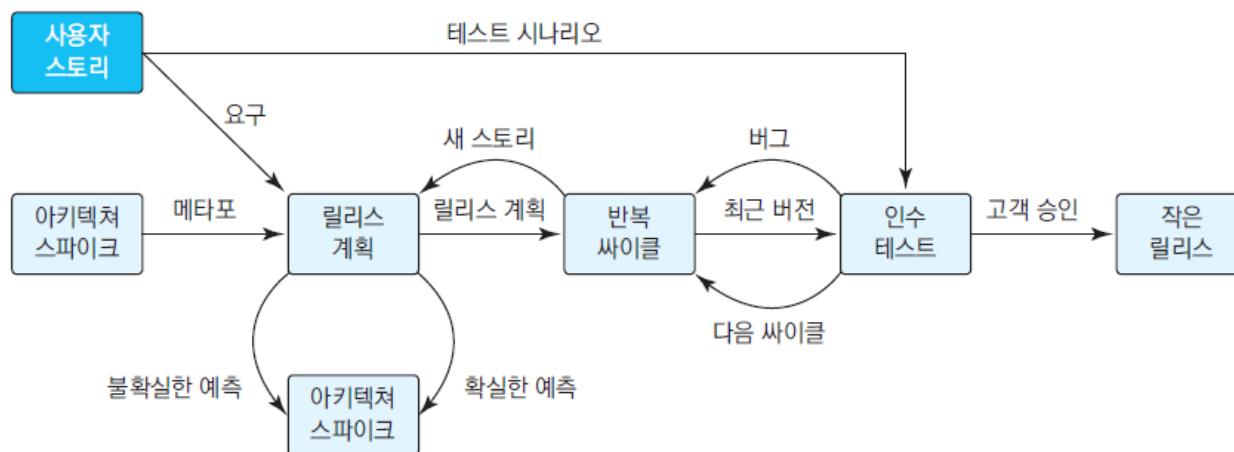


그림 2.17 익스트림 프로그래밍

## 2.4 지원 프로세스

- ISO/IEC 12207에서의 프로세스 그룹

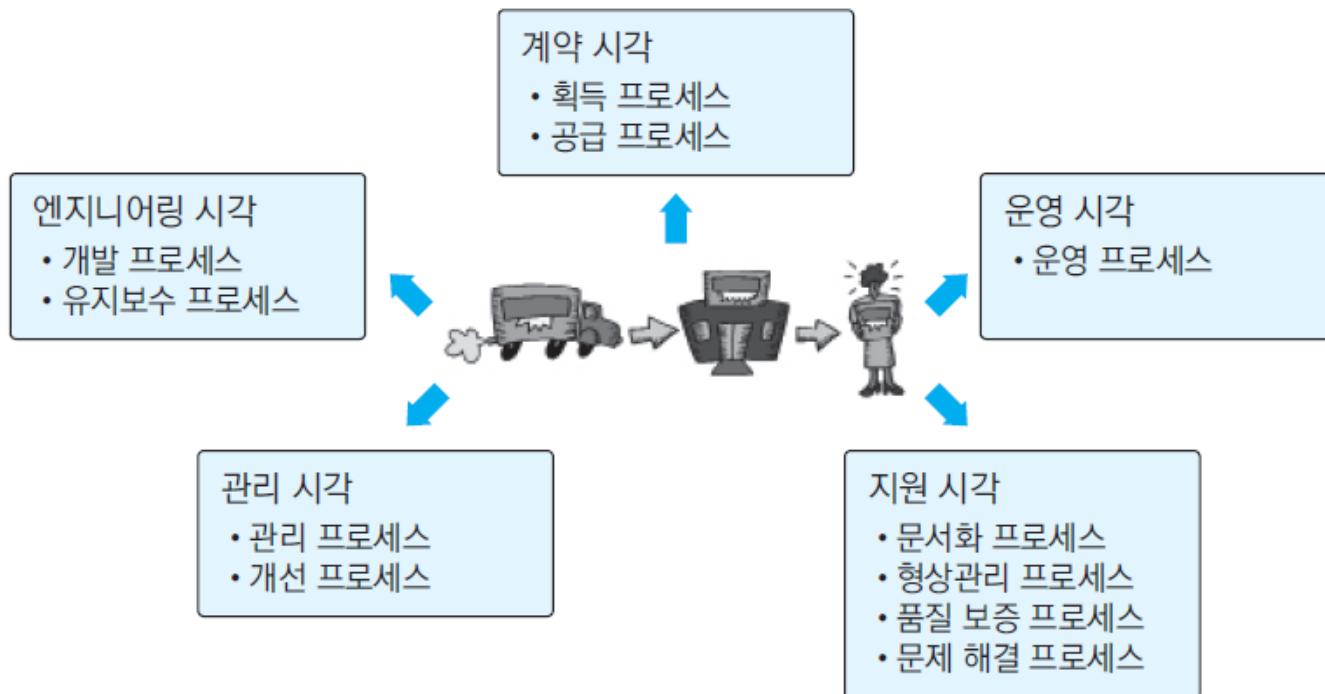


그림 2.19 ISO/IEC 12207에서의 프로세스 그룹

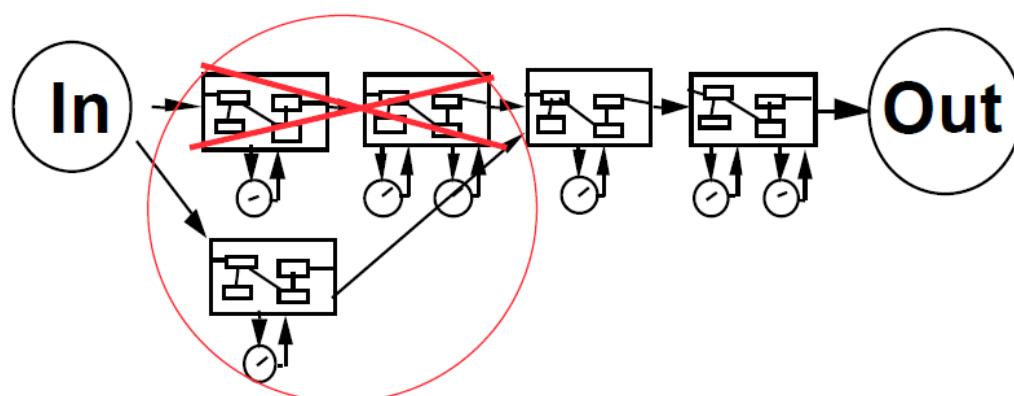
# 관리 프로세스

---

- 비용과 품질 목표를 달성하기 위하여 프로젝트 관리하는 데 필요한 모든 작업
  - 계획, 모니터링과 제어, 분석
- 프로젝트 모니터링과 제어는 개발 프로세스의 모든 단계를 포함하므로 가장 긴 기간 동안 이루어짐

# 품질 보증 프로세스

- 프로세스와 프로덕트에 대한 품질을 관리하고 향상시키는 것
- 인스펙션 프로세스
  - 개발 결과에서 결함을 찾거나 방지하기 위한 노력
  - 정의된 프로세스에 따라 동료 그룹이 작업 결과를 검사하는 것
- 프로세스 관리 프로세스



# 형상 관리 프로세스

- 개발 중에 발생하는 변경을 체계적으로 컨트롤 하는 것
- 개발작업과 독립적인 작업

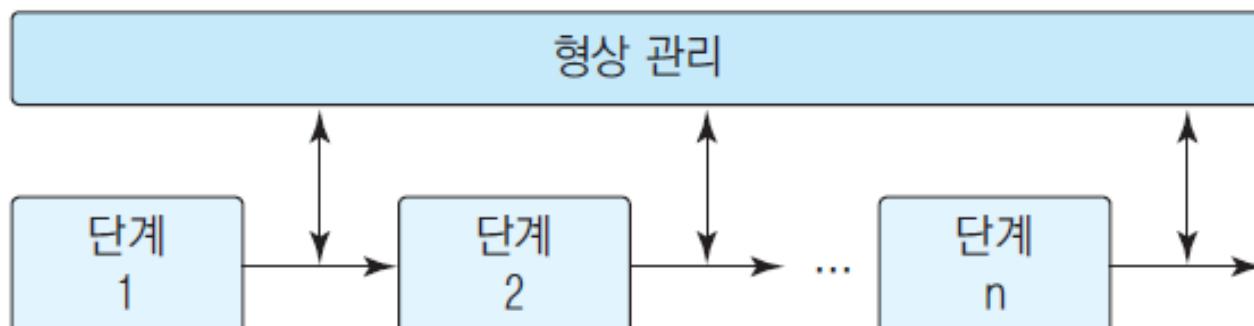


그림 2.22 형상 관리와 개발 프로세스

## 2.5 방법론

---

- **방법론**
  - 소프트웨어 프로세스의 각 작업을 어떻게 수행하느냐를 정의
- **프로세스**
  - 일반적으로 개발할 때 하여야 할 작업만을 명시
  - 어떤 관계가 있는지 나타내지 않음

# 구조적 방법론

- 분리와 정복(divide and conquer)원리 적용
- 자료 흐름도를 구조도로 변경하는 과정
  - 구조도 : 모듈 사이의 관계를 나타내는 그래프

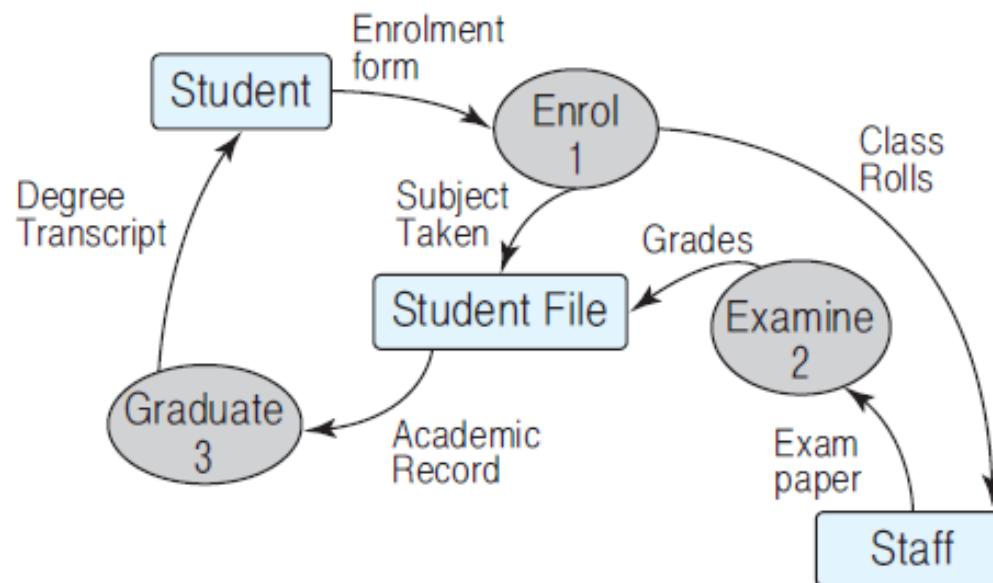


그림 2.23 자료 흐름도

# 정보공학 방법론

---

- 특징
  - 기업 중심
  - 전략적 시스템 계획 중심
  - 데이터 중심
  - 분할과 정복
  - 공학적 접근
  - 사용자의 적극적 참여

# 절차와 방법

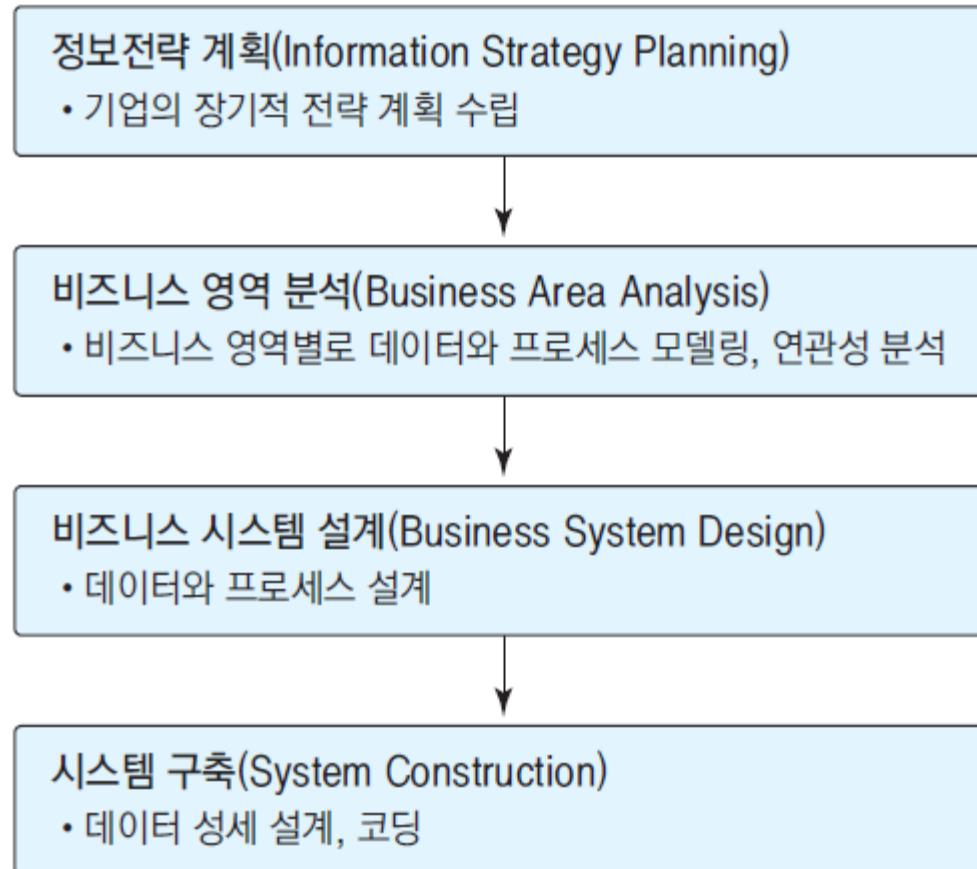
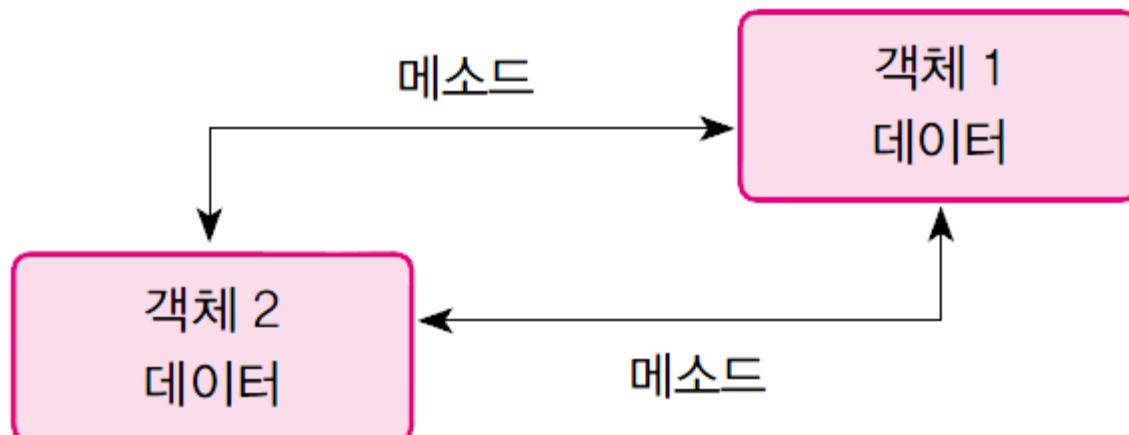


그림 2.24 정보공학 방법론의 절차

# 객체지향 방법론

- 자료와 함수를 가까운 곳에 정의하여 객체로 묶어두고 객체 사이에 메시지를 호출하여 원하는 기능을 담당하게 하는 것
- 객체지향 패러다임



# 세 가지 방법론의 비교

표 2.2 방법론의 비교

	구조적 방법론	정보공학 방법론	객체지향 방법론
특징	프로그램 로직 중심. 그림 (DFD, 구조적 다이어그램)을 그려 요구사항을 분석, 문서화 하는 체계적 방법	기업정보 중심이며 전략 계획을 수립한 후 데이터 중심으로 CASE 도구를 사용하여 공학적으로 접근한다.	비즈니스를 유스케이스로 분석하고 자료와 함수를 묶은 클래스를 파악하여 상호작용하는 객체들로 시스템을 구성하는 방법
설계 관심사	함수위주	자료 위주	클래스 위주
설계의 핵심	모듈	엔티티	객체
중심 방법	프로그래밍 기법	기업의 전략 및 산출물 중심	설계의 표현

공부를 해야 비로소 사람이 될 수 있다.

나무는 먹줄을 따르면 곧아지고  
쇠는 숫돌에 갈면 날카로워진다.  
이렇듯 군자도 매일 성찰해야  
앎이 밝아지고 행동에 허물이 없게 된다.  
학문이란 죽은 뒤에야 끝나는 것이다.  
학문의 방법에는 끝이 없지만,  
그 뜻은 잠시라도 내려놓을 수가 없다.  
학문을 하면 사람이고, 학문을 하지 않으면 짐승이다.

-순자, 권학편

성악설로 유명한 순자는 사람은 배우지 않으면  
악한 습성이 그대로 나오고,  
배워야 비로소 인간이 된다고 주장합니다.  
더 나아가 배우고 또 배우다 보면  
어느새 가능성이 극대화되어 진정한 인간, 성인이 될 수 있기에  
배움은 그만두어서는 안된다고 말합니다.