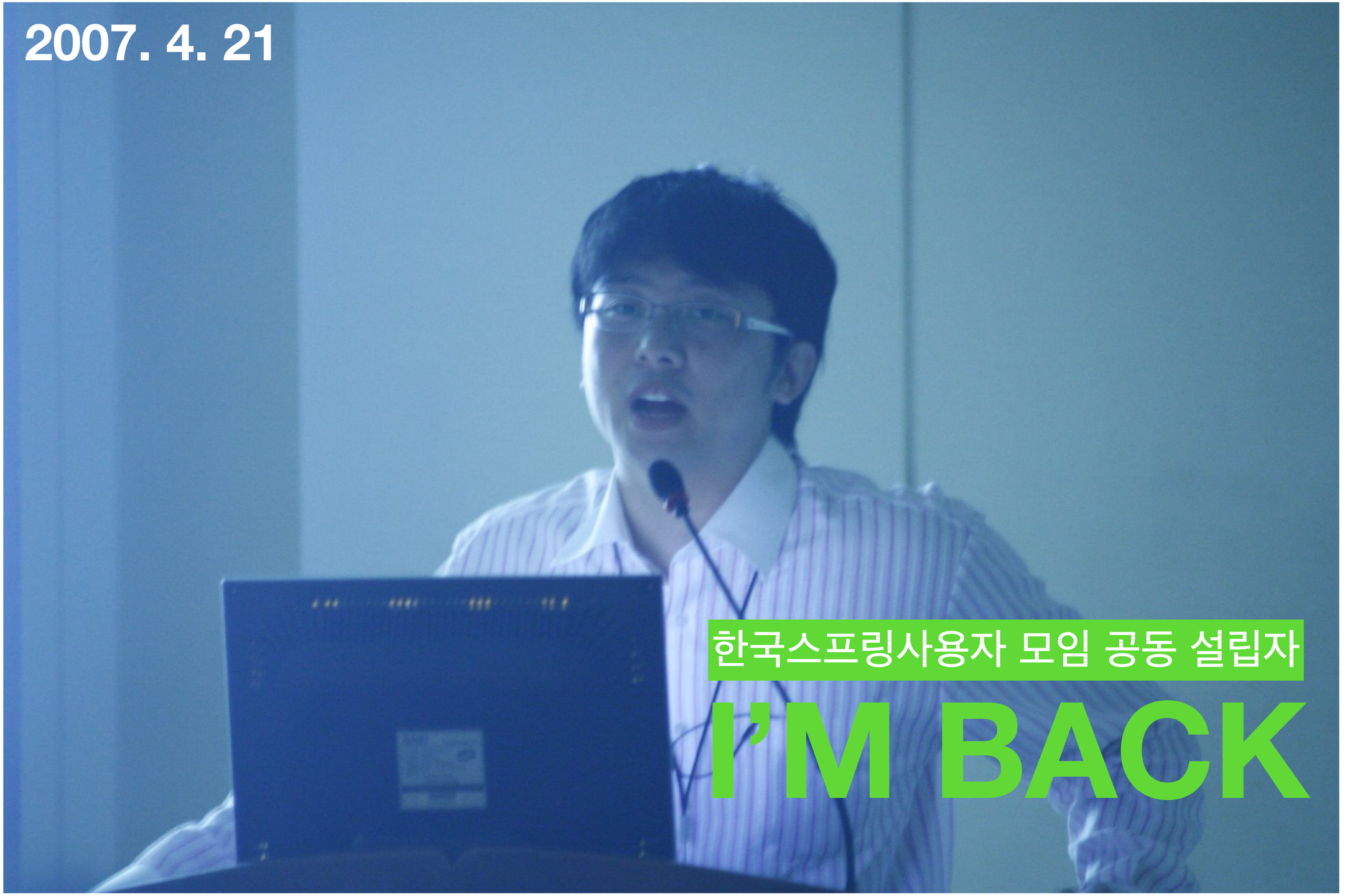
2007. 4. 21



한국스프링사용자 모임 공동 설립자

# I'M BACK

©안영회


# 켄트 벡의 Tidy First?

더 나은 소프트웨어 설계를 위한
32가지 코드 정리법

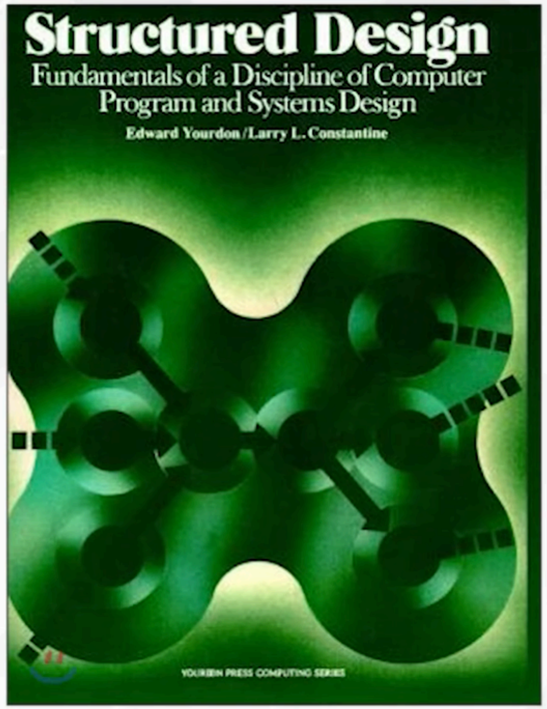
시스템/하드웨어
설계 분야
아마존 베스트셀러

컴퓨팅의 선구자,
레리 콘스탄틴
강력 추천

한빛미디어 Hanbit Media, Inc.

켄트 벡 지음
안영회 옮김

"뉴턴의 운동 법칙을 소프트웨어 설계에 적용하고 있었죠."

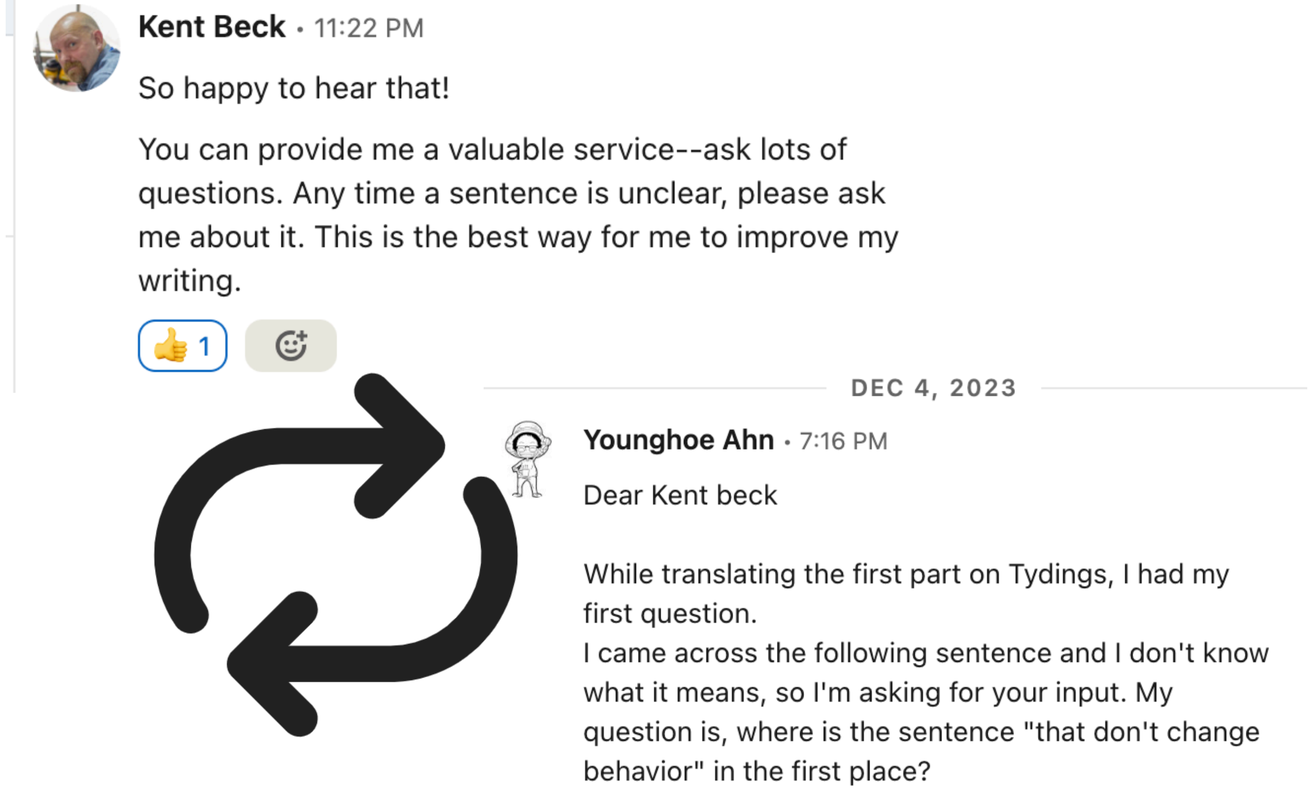



출처: 위키피디아 (Edward Yourdon)


출처:https://www.youtube.com/watch?v=X3h9yS9S26M

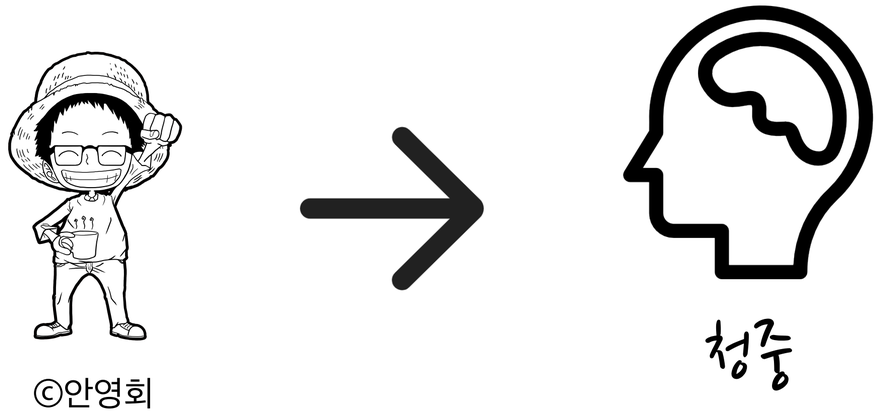
## 맥락(Context) 해설

책에는 없는 내용을 둘러싼 맥락




## 설계에 대한 견해

설계 덕후로서 감상




©안영회

PART 01

# 15가지 코드 정리법

- 카탈로그
- 몸으로 익히기
- 리팩터링과 비교
- 유기체적 공진화


# 15가지 코드 정리법

카탈로그


- 보호 구문
- 안 쓰는 코드
- 대칭으로 맞추기
- 새로운 인터페이스로 기존 루틴 부르기
- 읽는 순서
- 응집도를 높이는 배치
- 선언과 초기화를 함께 옮기기
- 설명하는 변수
- 설명하는 상수
- 명시적인 매개변수
- 비슷한 코드끼리
- 도우미 추출
- 하나의 더미
- 설명하는 주석
- 불필요한 주석 지우기

## Bottom-up

켄트 벡이 선호하는 방식

## 아기 발걸음

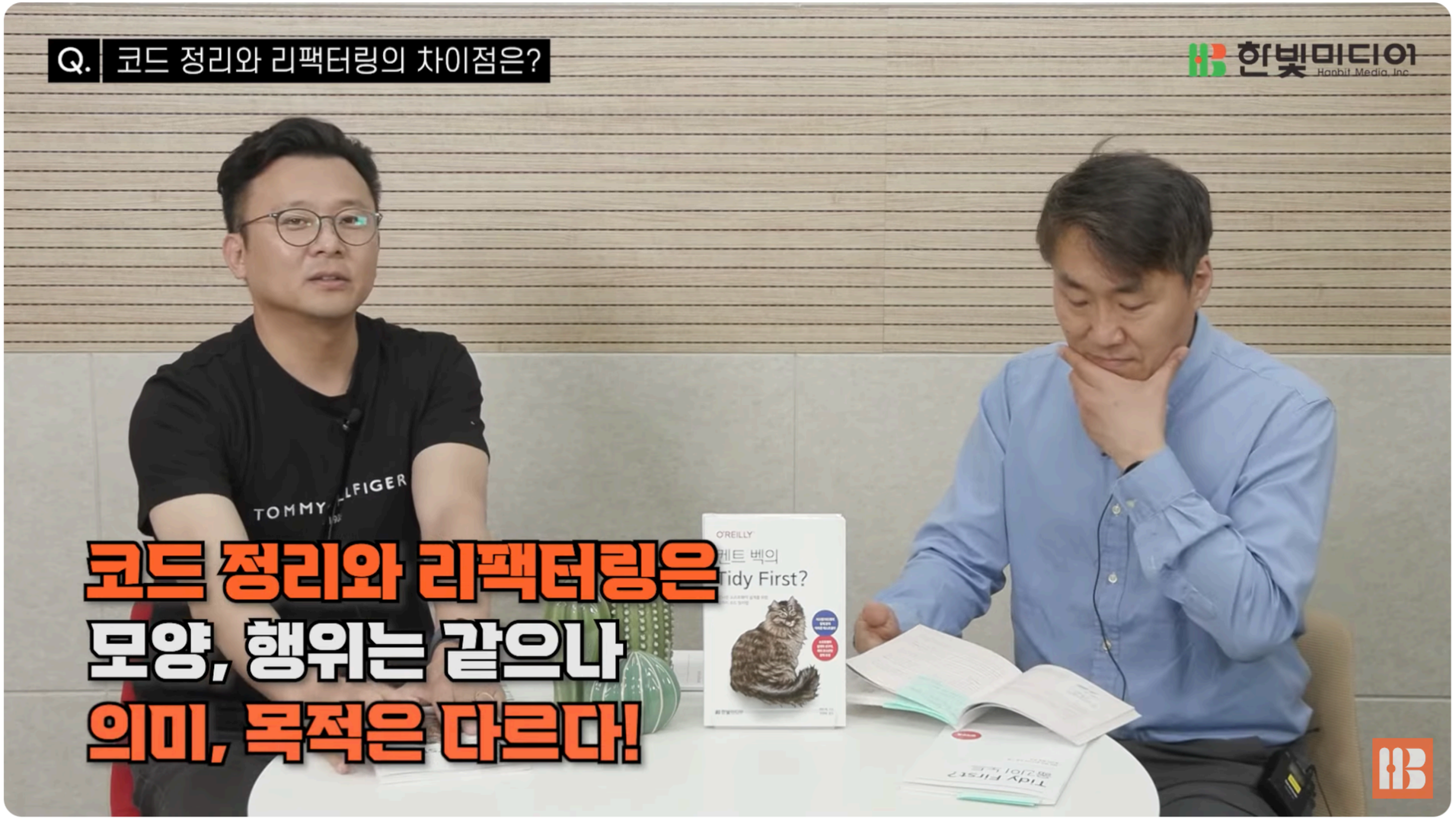
자연스러운 학습의 모양새

## Man in the mirror

먼저 나와 인간 관계 활동 익히기

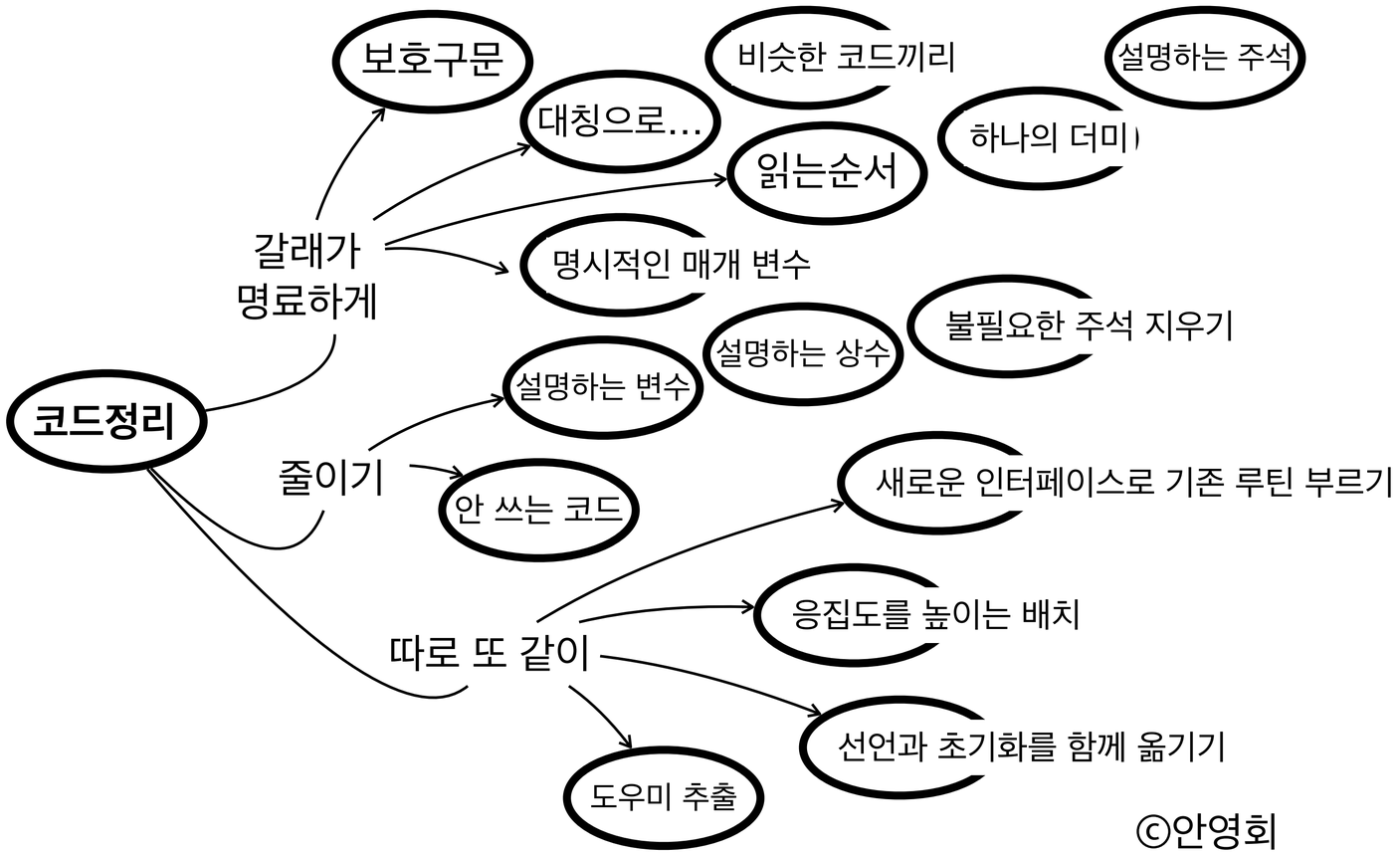
# 15가지 코드 정리법

리팩터링과 비교




출처: https://www.youtube.com/watch?v=dvi4S8LI17g

# "코드 정리를 통해 코드를 유기체로 다루는 법을 익할 수 있다"



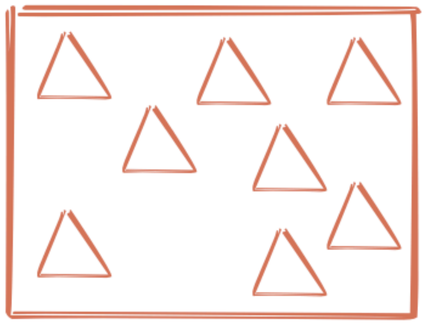



# 관리

- 코드 정리 구분
- 연쇄적인 정리
- 코드 정리의 일괄 처리량
- 리듬
- 걸림 풀기
- 코드 정리 시점

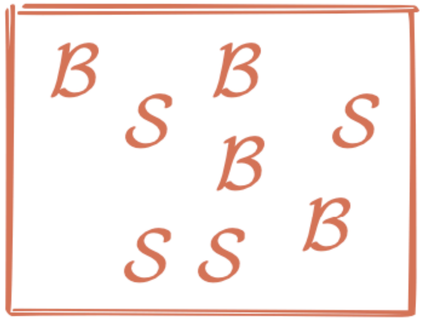

# 코드 정리 구분

일단 닥치는 대로 코딩



다양한 변경 필요성을 구분하지 않은 상태에서 변경 시도

작업의 종류 구분(인지)




한 번에 담기(배포&통합)

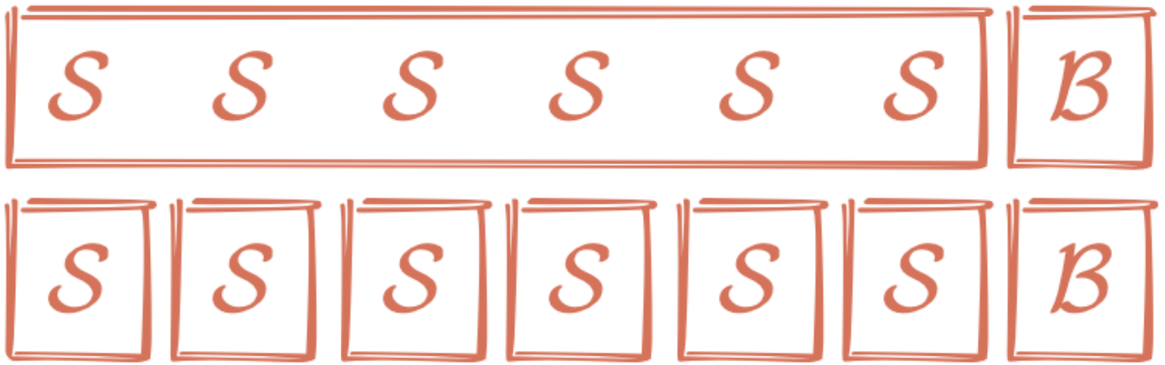

나누어 담기


"자꾸 자꾸 손이 가"

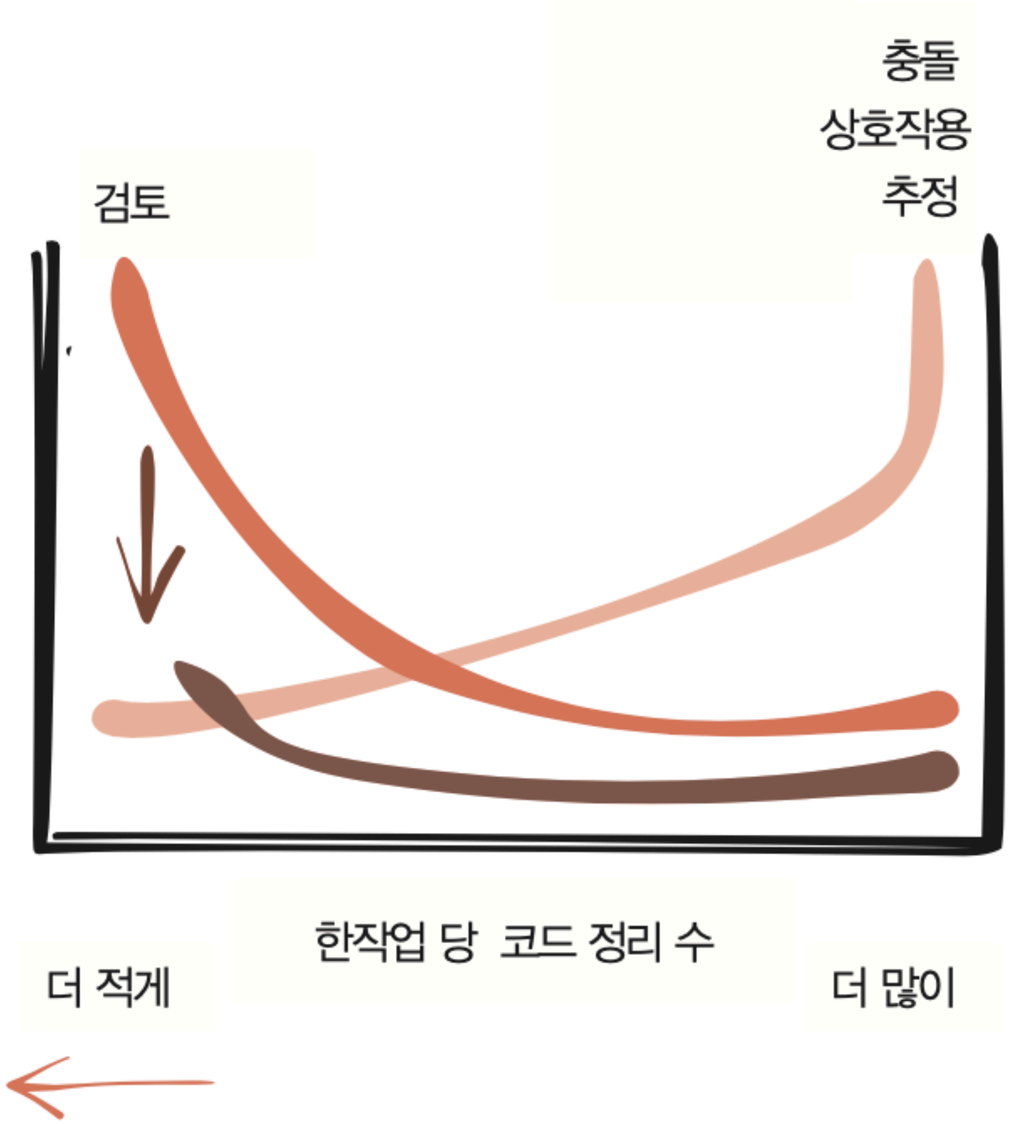

## 자꾸 손이 가게 하려면

- 이왕이면 더 작게해서, 더 자주 즐기기
- 작은 단계로 두어서, 다음 수를 내다보기

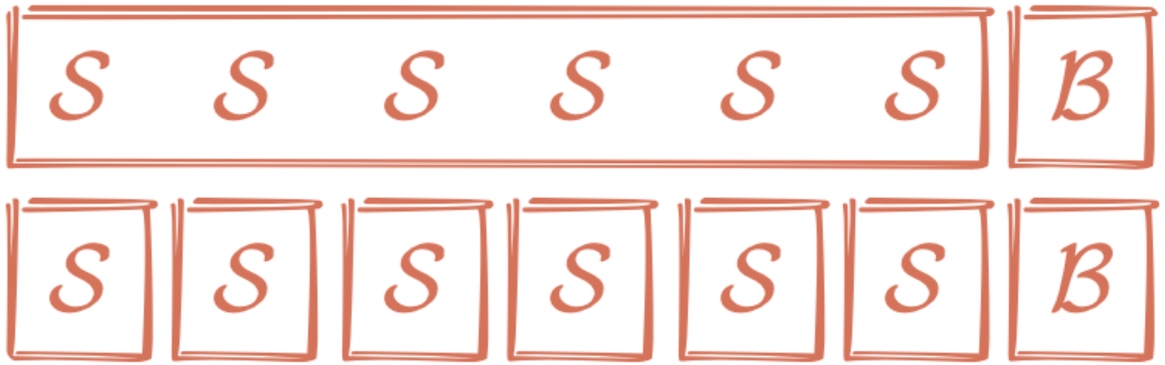
"둘은 어떤 차이가 있을까?"




# 코드 정리의 일괄 처리량




"일괄 처리 규모가 줄수록
검토 비용과 코드 정리
비용이 함께 줄어든다"



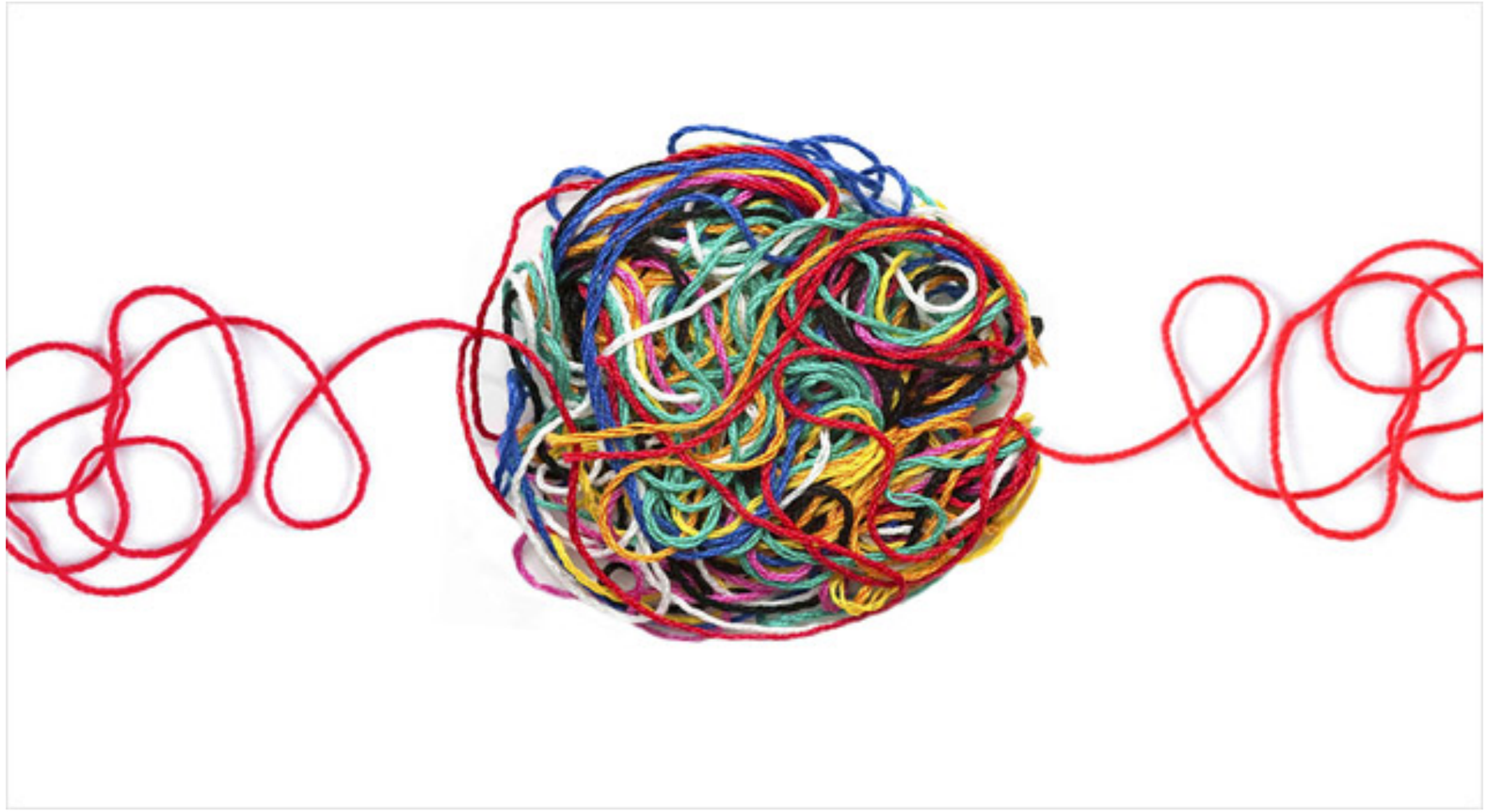

소프트웨어 설계는 Fractal

작고 빈번하게 할수록 (선택) 유리

소프트웨어 설계는 길을 닦는 일

쓰임새를 지켜보며 지속해야 하는 일

"실타래를 풀려면 실이 엉켜 있다는 사실을 알아차려야 시작할 수 있다"



# 코드 정리 시점

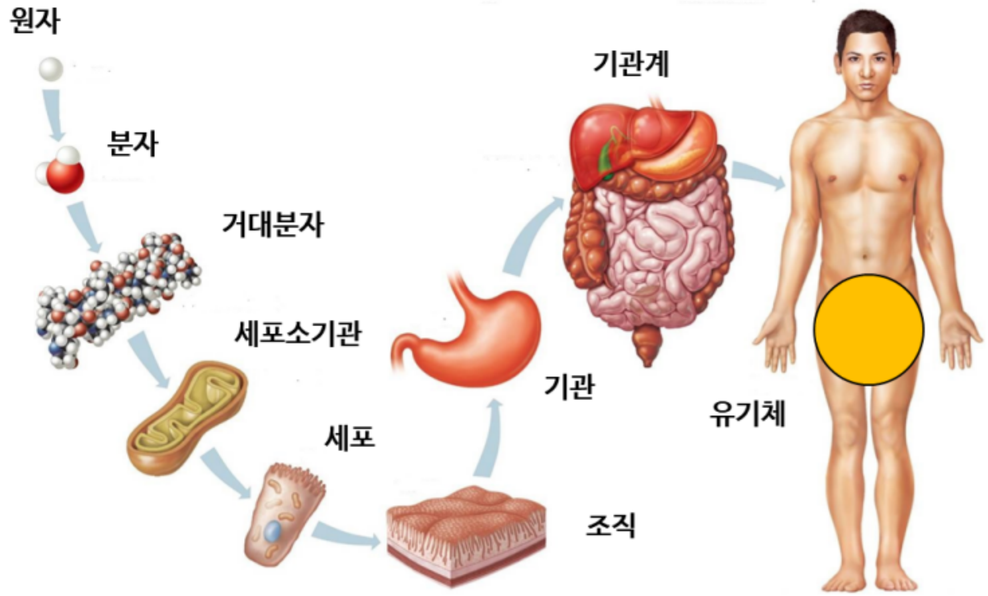

- "아예 안 한다면?"
- 나중에 정리하기(재미목록으로)
- 동작 변경 후에 코드 정리
- 코드 정리 후에 동작 변경

# 이론

- 요소들을 유익하게 관계 맺는 일
- 구조와 동작
- 경제 이론: 시간 가치와 선택 가능성
- 되돌릴 수 있는 구조 변경
- 결합도와 응집도

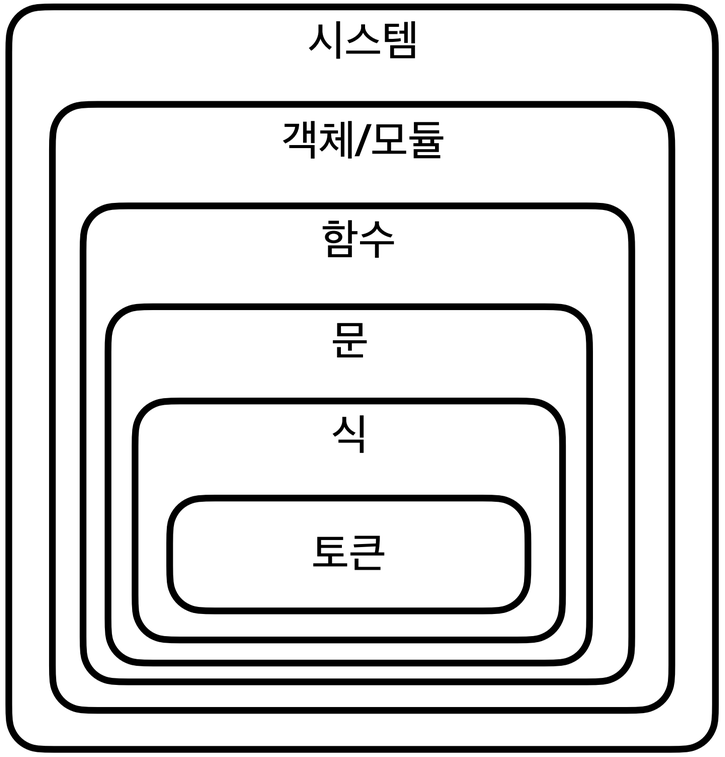

# 요소들을 유익하게 관계 맺는 일

## "요소들 / 관계 맺는 일 / 유익하게"




유기체성!

→ 외방

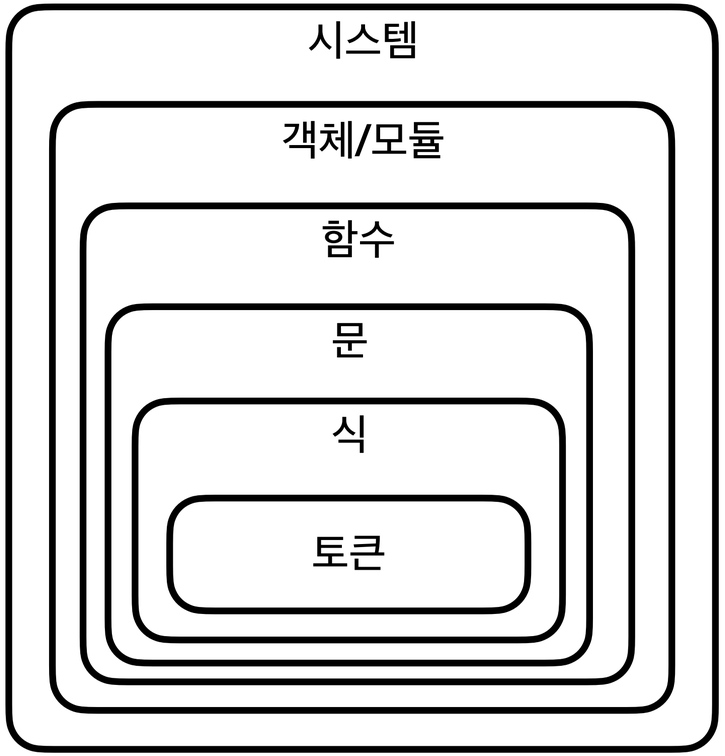



# 요소들을 유익하게 관계 맺는 일

## 설계자의 할 일

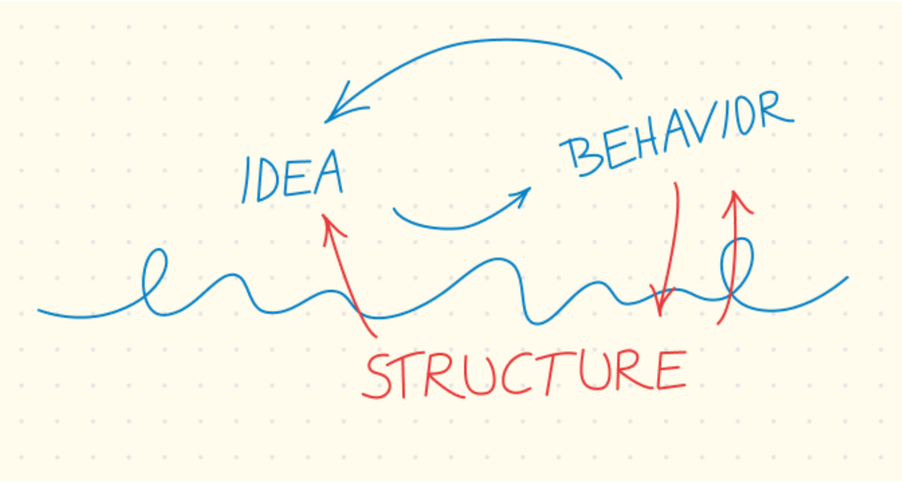
- 요소를 만들고 삭제하기
- 관계를 만들고 삭제하기
- 관계의 이점을 높이기

반영




소프트웨어는 두 가지 방식으로 가치를 만듭니다.

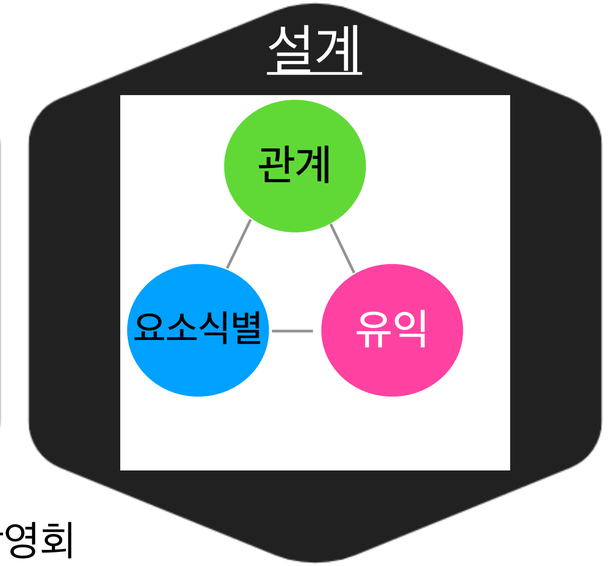
- 현재 소프트웨어가 하는 일
- 미래에 새로운 일을 시킬 수 있는 가능성




참고: https://brunch.co.kr/@graypool/1568

외부 관찰자 관점


유기체처럼 구성하기






# 이론

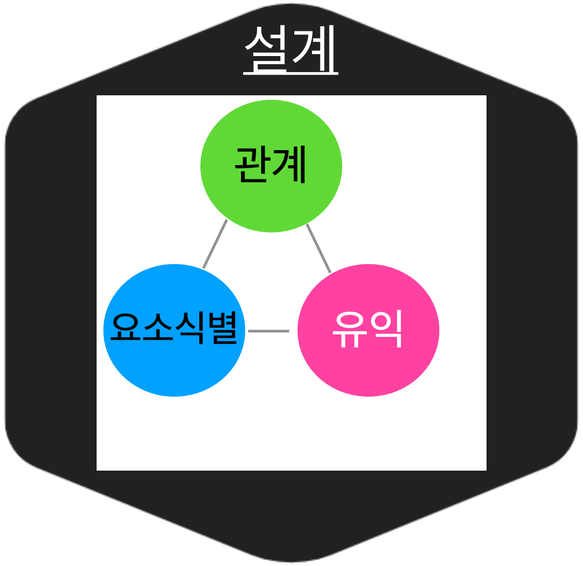
- 경제 이론: 시간 가치와 선택 가능성
- 오늘의 1달러가 내일의 1달러보다 크다
- 옵션
- 옵션과 현금흐름 비교


가치(value) > 가격(price) > 비용 (cost) - 생존 부등식 by 윤석철

코드의 가치 > 고객 만족도 > 비용 (시간)

# 되돌릴 수 있는 구조 변경

유기체처럼 구성하기

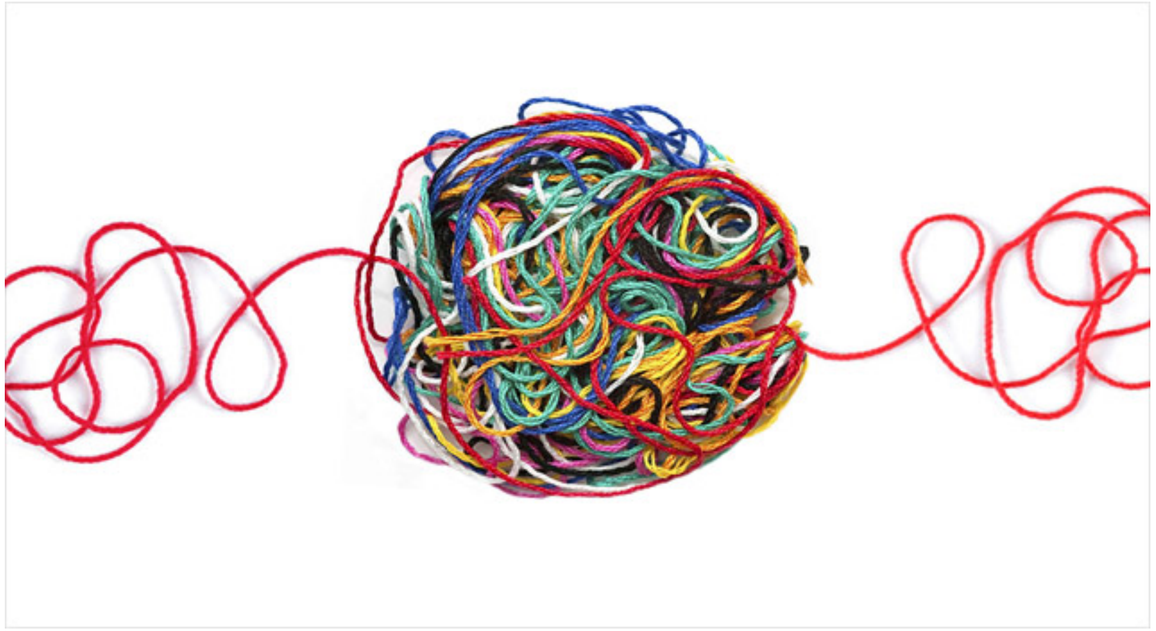



가역성을 비용으로 표현

코드 정리 → extract as a service → 재개발

결합도가 가진 두 가지 성질

- 일대다
- 연쇄작용



"복잡하다"의 의미

"변화가 예상치 못한 결과를 초래한다"

# 결합도와 응집도

## 비용(소프트웨어) ~= 결합도

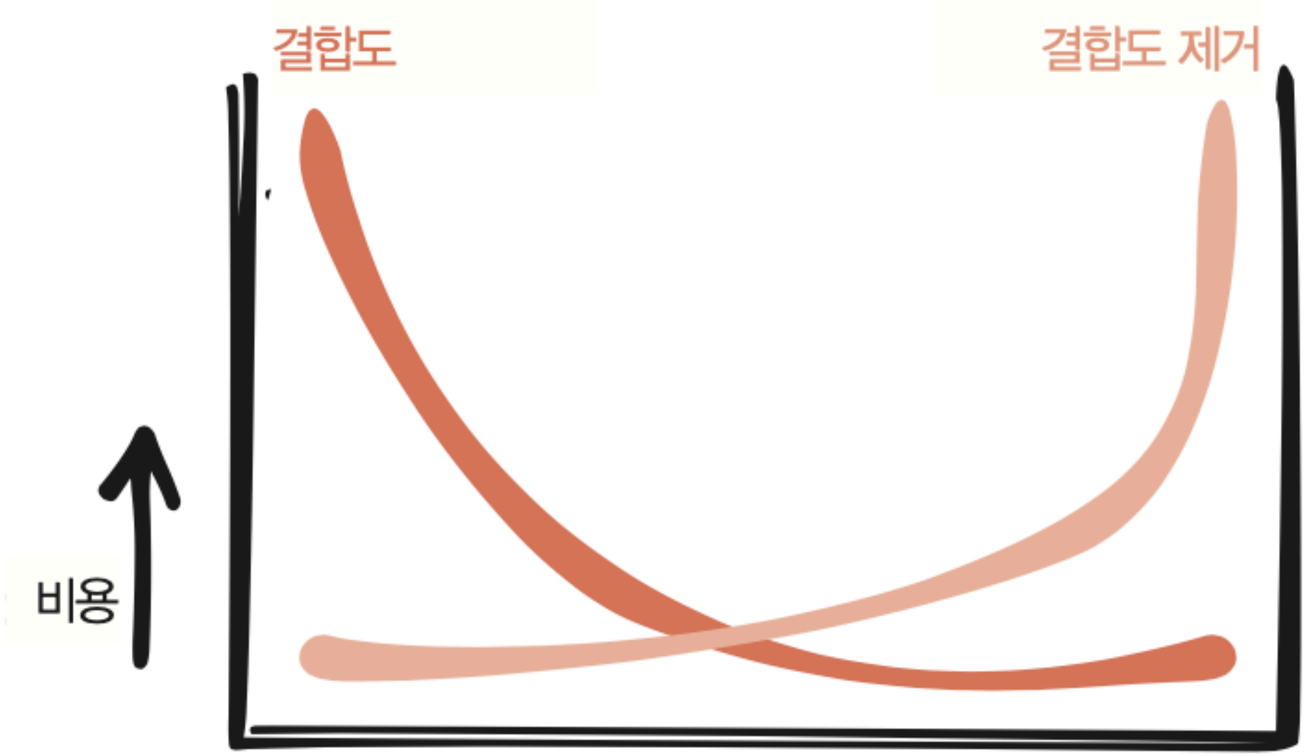
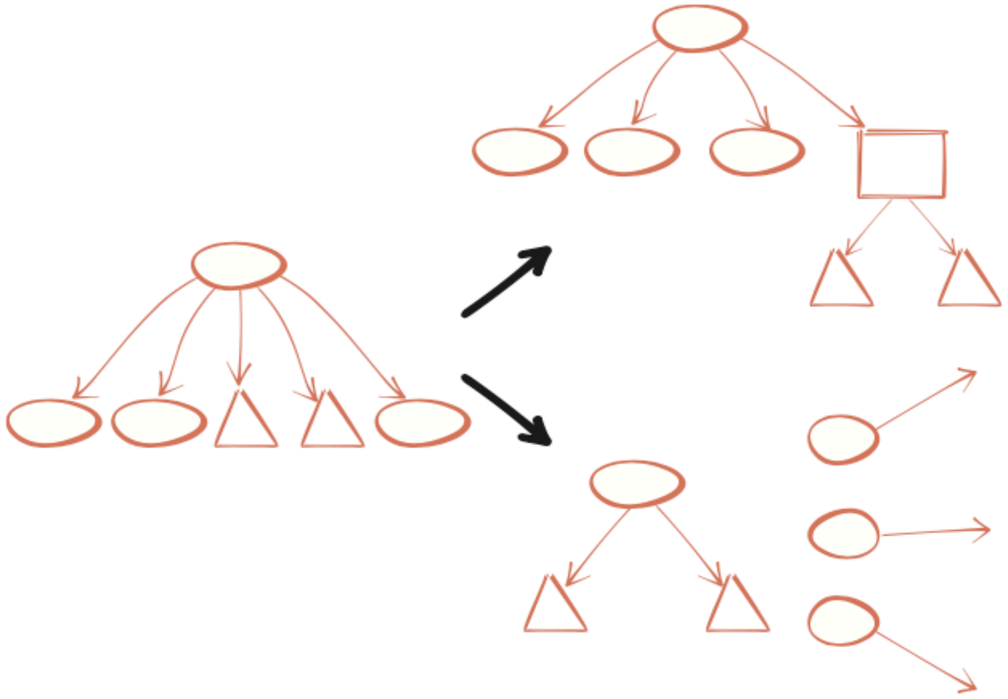
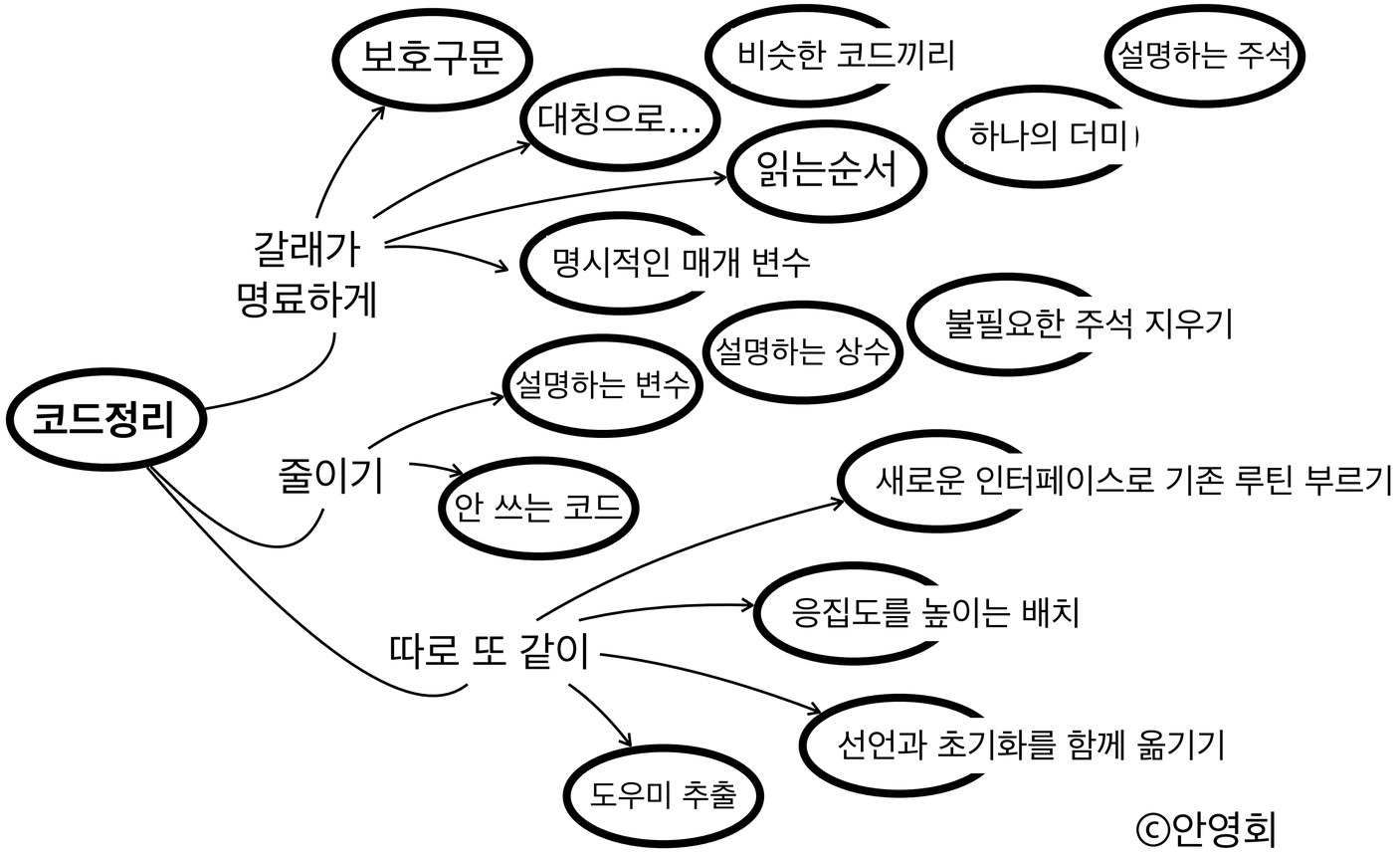



그림 31-1 결합도에 따르는 비용과 결합도 제거 비용의 상충 관계

# 결합도와 응집도

"관계의 양상을 부르는 서로 다른 이름"





# 결론

- 코드 정리법
- 관리
- 이론

—> Tidy Together?

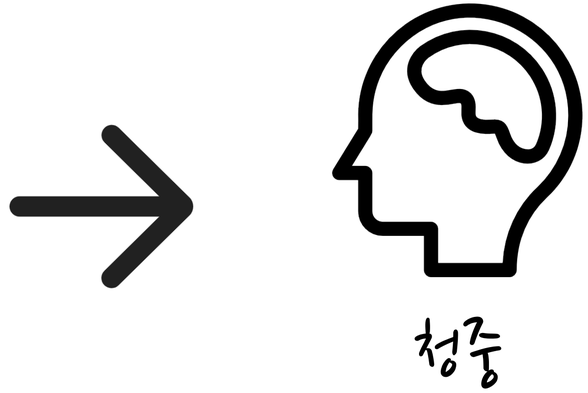

©안영회

"요소들 / 관계 맺는 일 / 유익하게"

= "유기체처럼 구성하기"

= "유기체성을 고려하기"

©안영회



청중

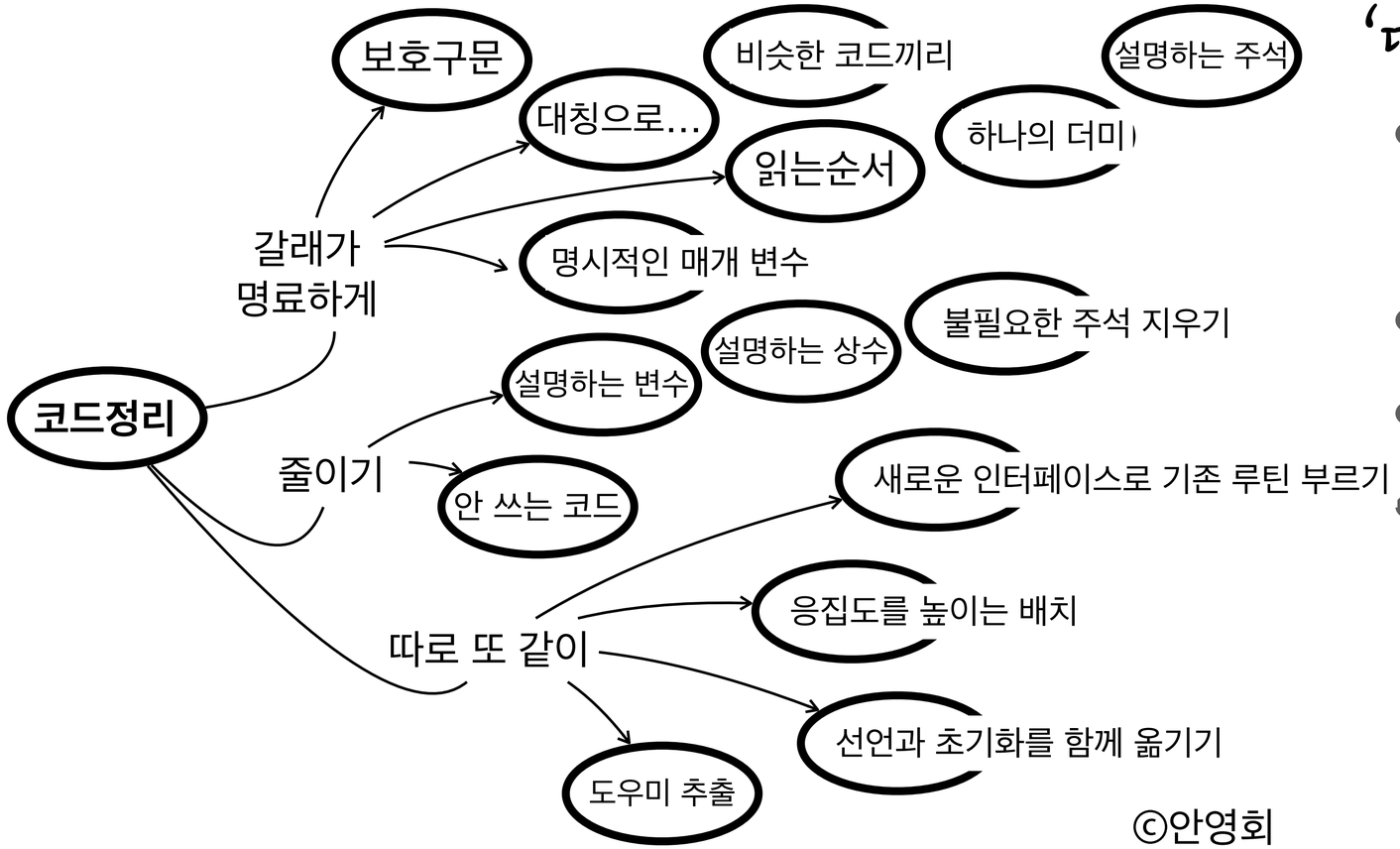

## 표준국어대사전 풀이

「1」 많은 부분이 일정한 목적 아래 통일 · 조직되어 그 각 부분과 전체가 필연적 관계를 가지는 조직체.

「2」『생명』 생물처럼 물질이 유기적으로 구성되어 생활 기능을 가지게 된 조직체.

이미지 출처: https://wordrow.kr/%EC%9E%90%EC%84%B8%ED%95%9C-%EC%9D%98%EB%AF%B8/%EC%9C%A0%EA%B8%B0%EC%B2%B4/2/

# 따로 또 같이 (함께성)






'따로 또 같이'를 구현하는 다양한 프로그래밍 기법

- Dependency Injection (Interface와 implementation 분리)
- RESTful API 활용
- MSA 아키텍처
- BFF 혹은 포트와 어댑터 적용

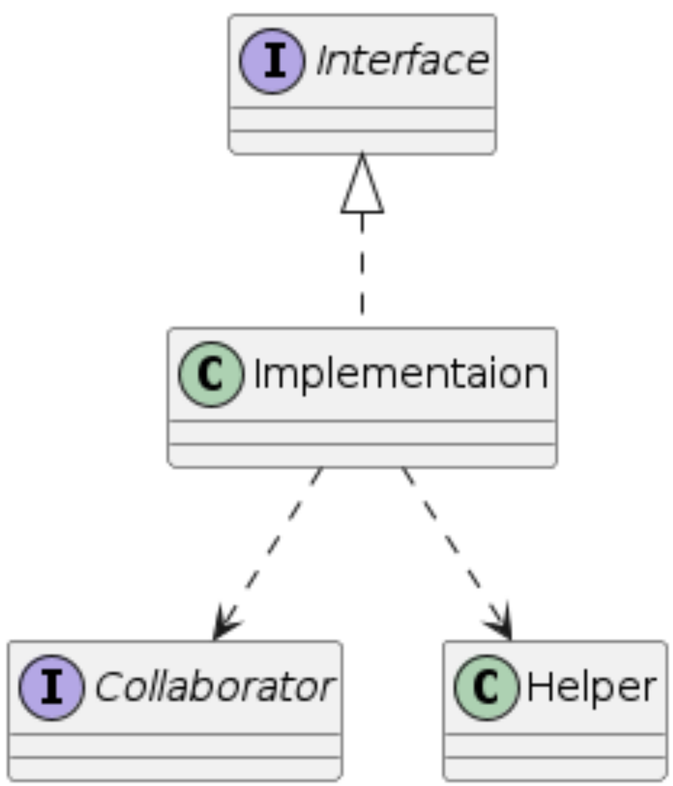
참고: https://brunch.co.kr/@graypool/1666

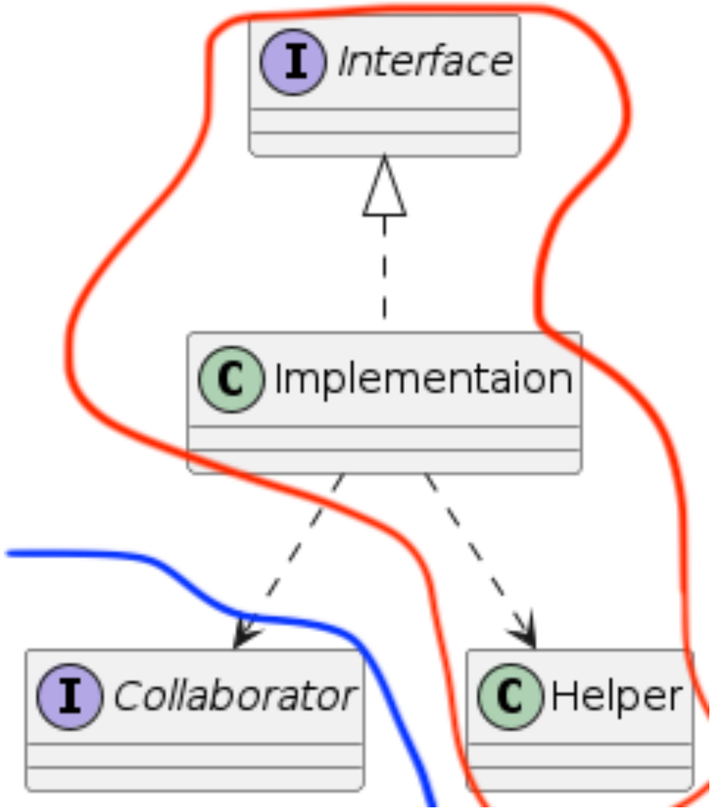
참고: https://brunch.co.kr/@graypool/259

관계

"요소들을 유익하게 관계 맺는 일"

경계

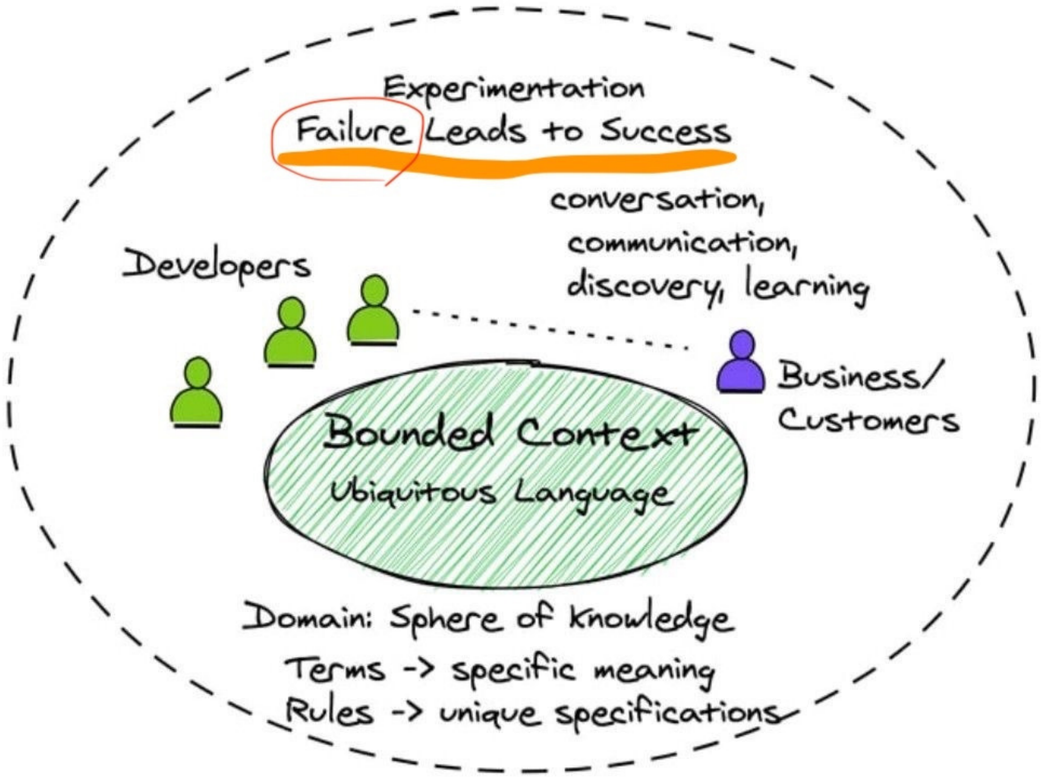



출처: Vaughn Vernon의 링크드인 페이지

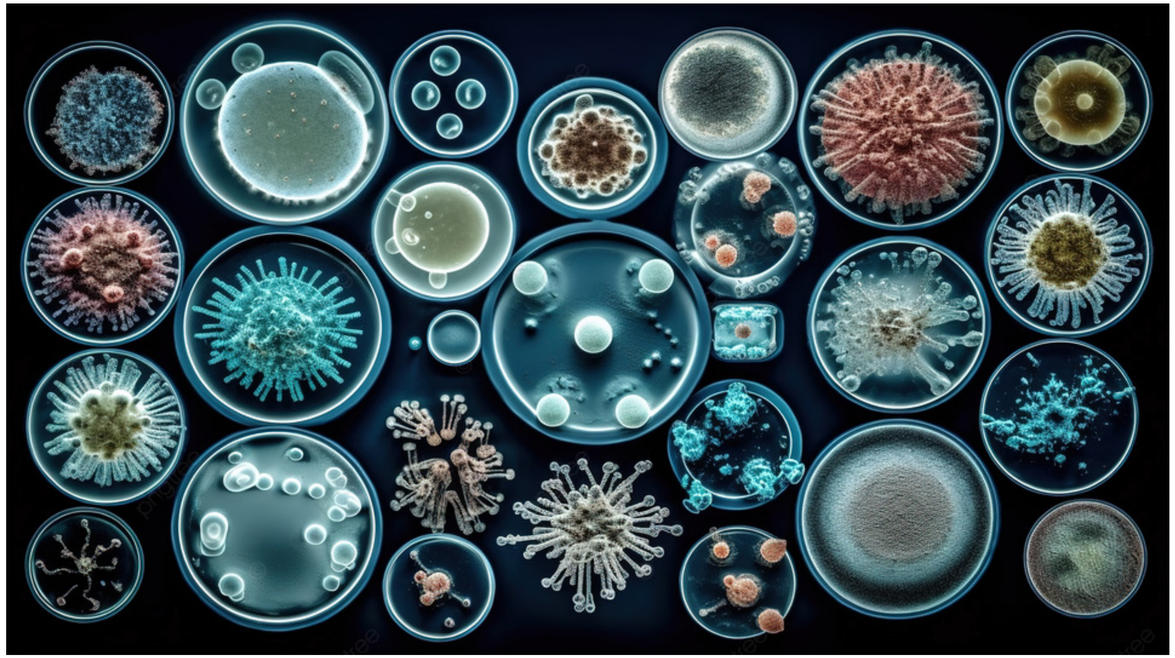
# "소프트웨어 설계는 프랙탈이므로 설계는 크게도 작게도 할 수 있다"

## 경계



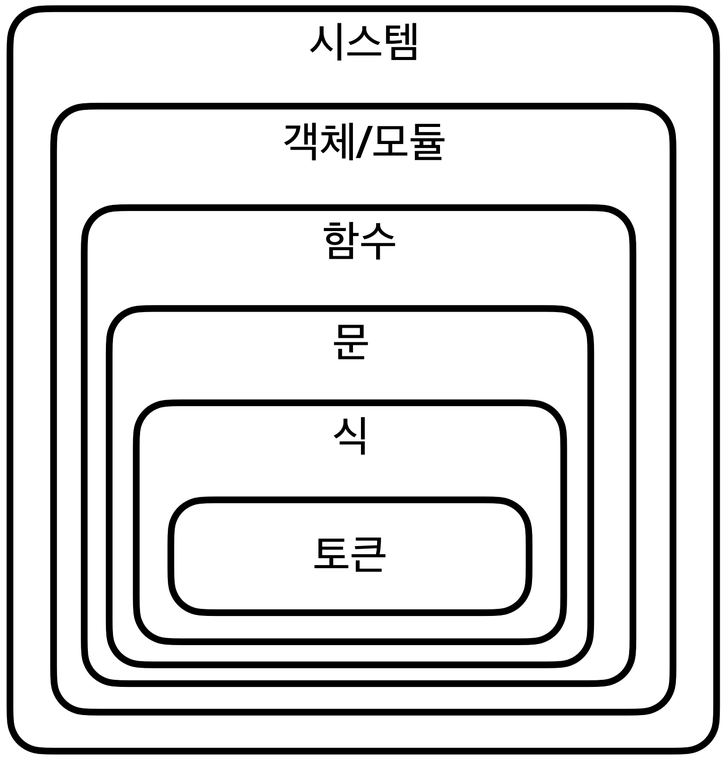

참고: https://brunch.co.kr/@graypool/509

## loosely-coupled!

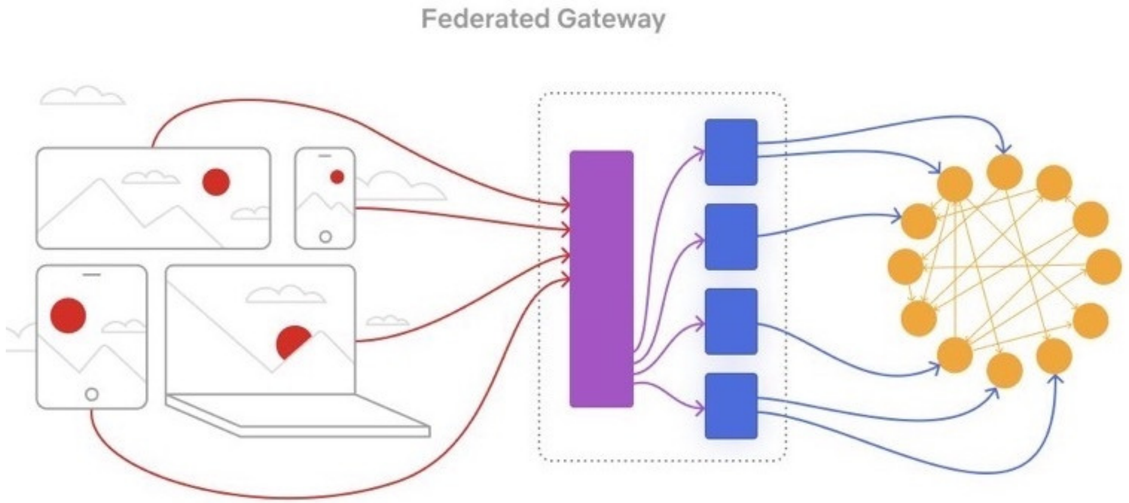


참고:https://yozm.wishket.com/magazine/detail/1926

# "소프트웨어 설계는 프랙탈이므로 설계는 크게도 작게도 할 수 있다"




## 층위

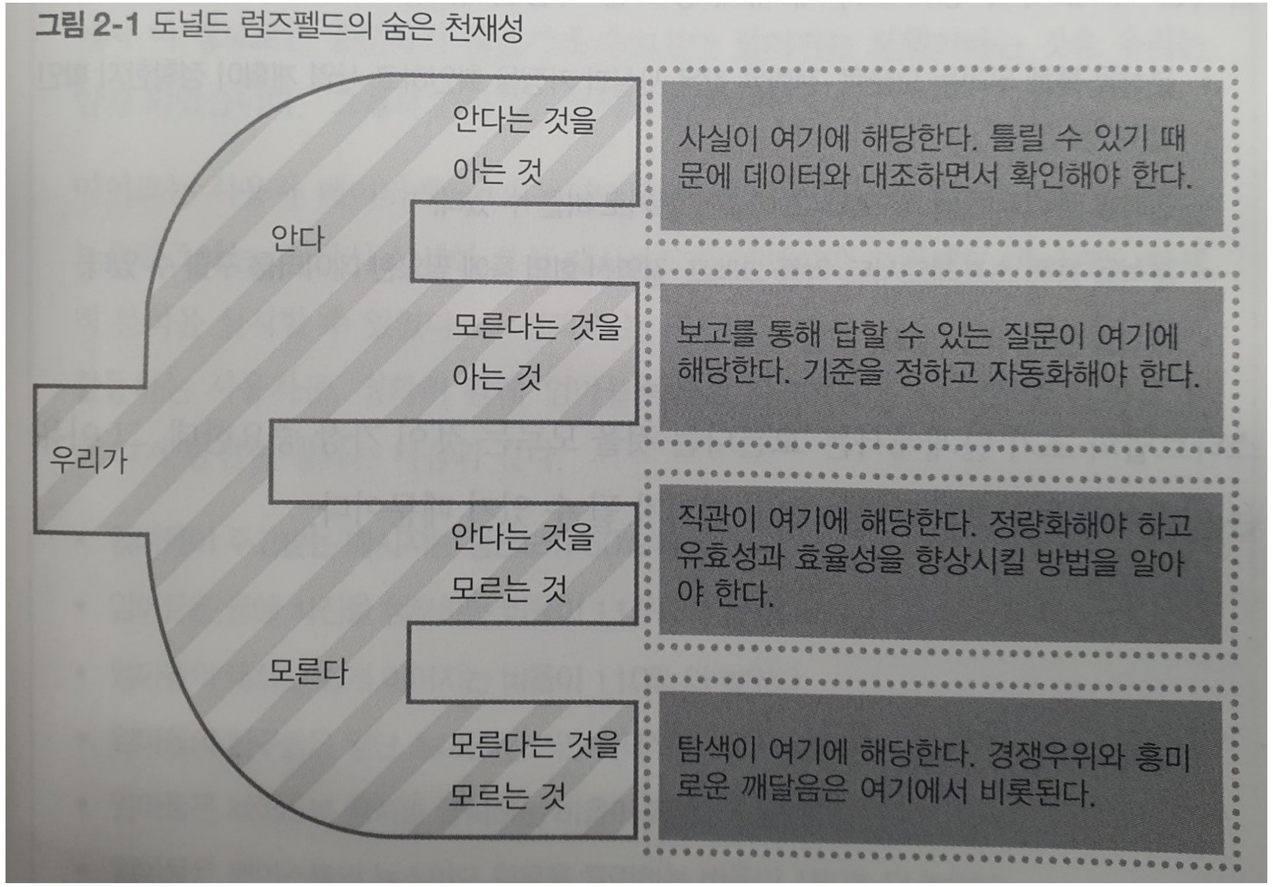



참고: https://brunch.co.kr/@graypool/487

# 설계 덕후로서 감상

## "내 인지의 한계를 아는가?"

**그림 2-1** 도널드 럼즈펠드의 숨은 천재성




## "사람마다 다르게 인식한다."