

- 일반적인 모양과 조화를 위한 스타일을 정하는 작업
 - 소프트웨어 아키텍처에 적용
 - 시스템분할, 전체 제어 흐름, 오류 처리 방침, 서브시스템 간의 통신 프로토콜 포함



계층 구조 스타일

● 시스템을 계층으로 구성하는 방법

하위 tier (또는 layer) 는 상위 tier(또는 layer)에 API 제공

Tier vs Layer ?

N-tier architecture is a client-server architecture.

전형적 3-tier architecture:

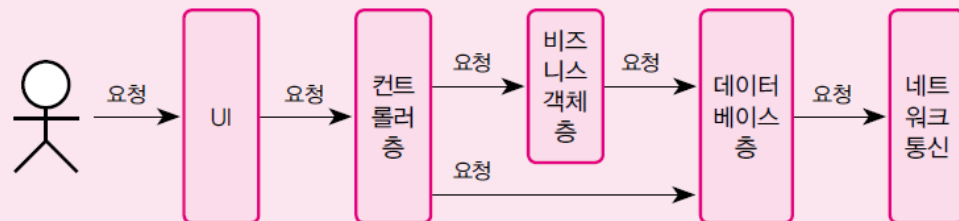
- presentation
- application processing (business logic, service)
- data storage

4-tier ?

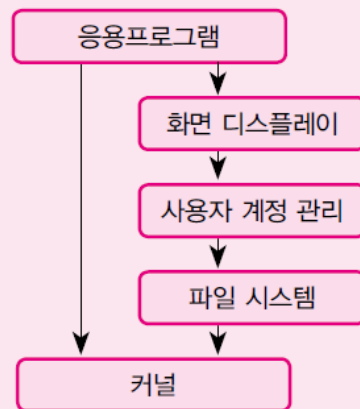
persistent tier

(b), (c) →

Layered architecture



(a) N-tier 아키텍처의 예



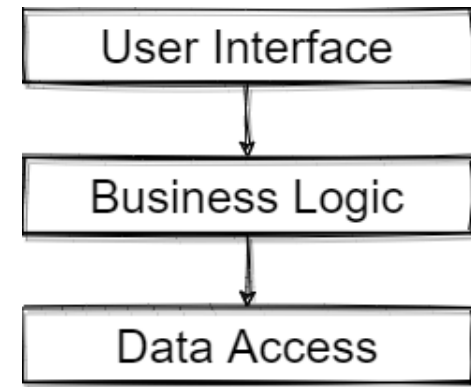
(b) 운영체제에서의 계층구조



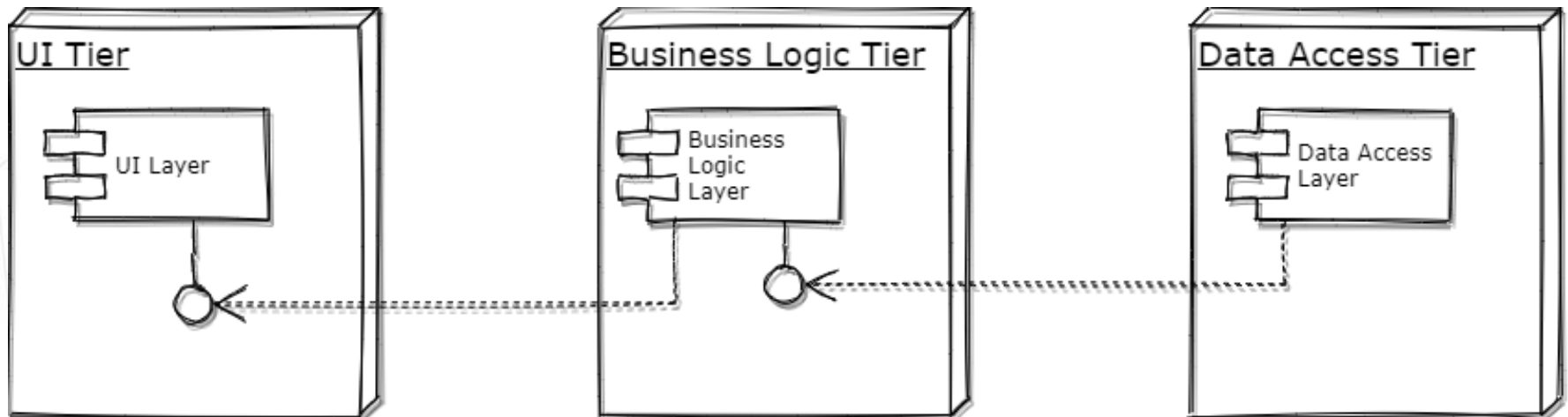
(c) 통신 시스템의 계층

Tiers vs Layers

- Tiers: logical separations of code
- Layers: physical separations of code.



3 Layers

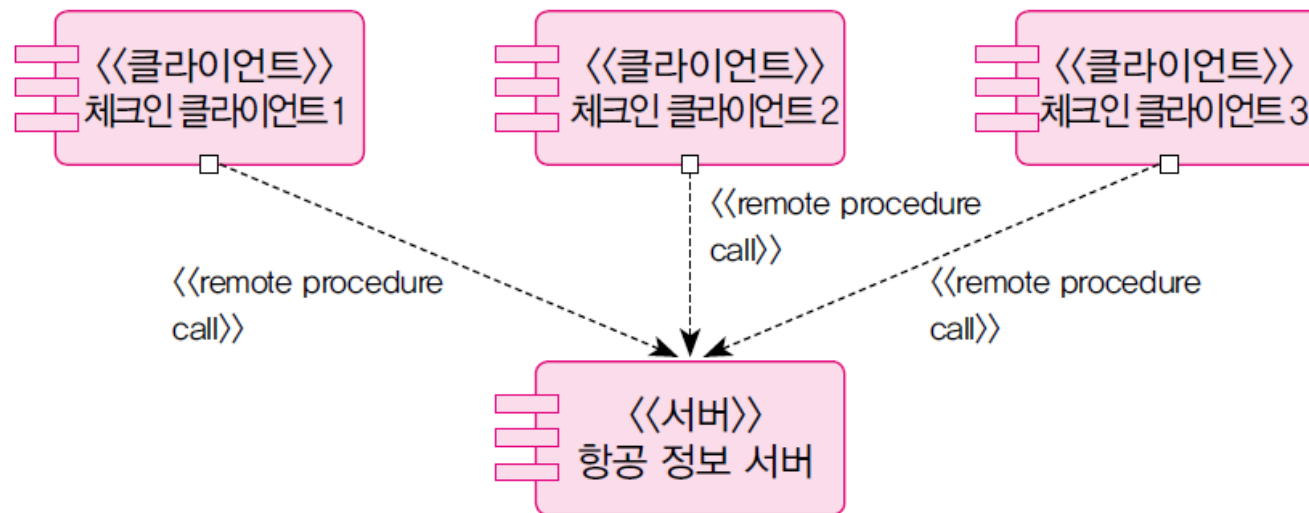


3 Tiers

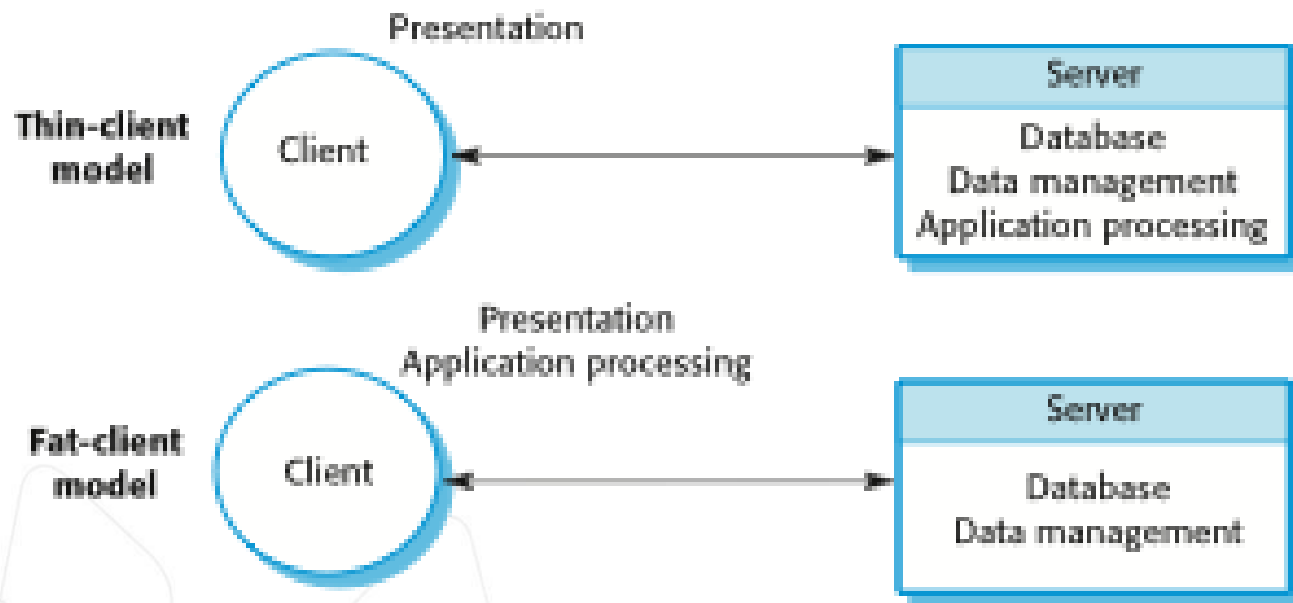
2 tier 클라이언트 서버 스타일 (분산시스템 architecture 중 하나)

● 클라이언트 서버(Client Server)

- 서버와 여러 개의 클라이언트로 구성
- 요청과 결과를 받기 위하여 동기화 되는 일을 제외하고는 모두 독립적이다.
- 특정 서브시스템이 다른 서브시스템에 서비스를 제공하도록 지정할 때 유용하다.
- 대부분의 웹 기반 애플리케이션과 파일 서버, 전송 프로토콜 포함



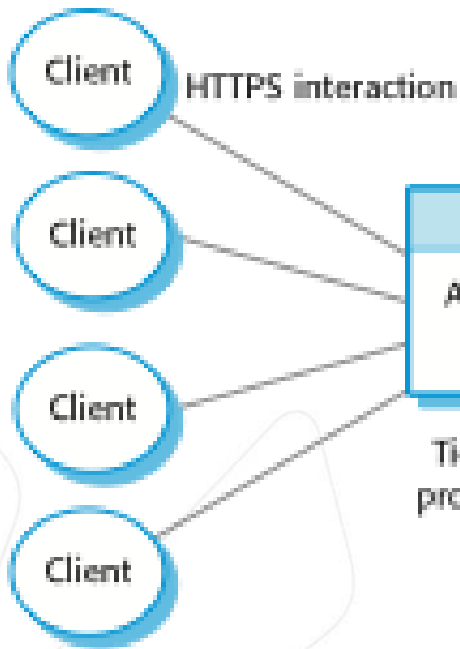
클라이언트 서버 스타일의 적용



N tier 클라이언트 서버 스타일 (분산시스템 architecture 중 하나)

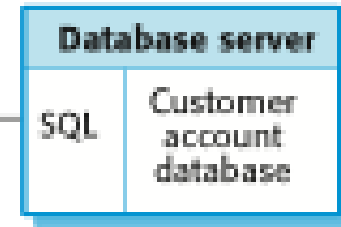
ex) Three-tier architecture for an Internet banking system

Tier 1. Presentation



Tier 2. Application processing and data management

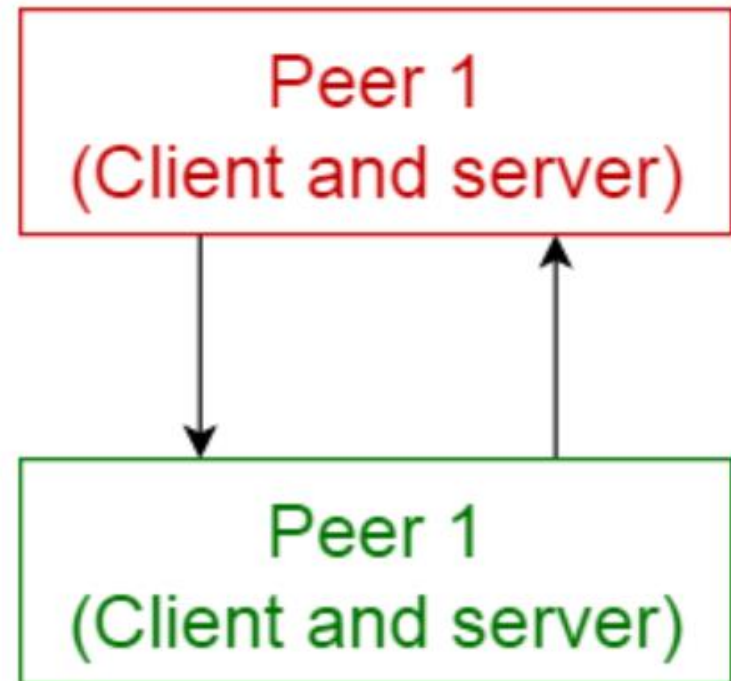
SQL query



Tier 3. Database processing

● 활용:

- Gnutella나 G2와 같은 파일 공유 네트워크
- P2PTV 같은 멀티미디어 프로토콜

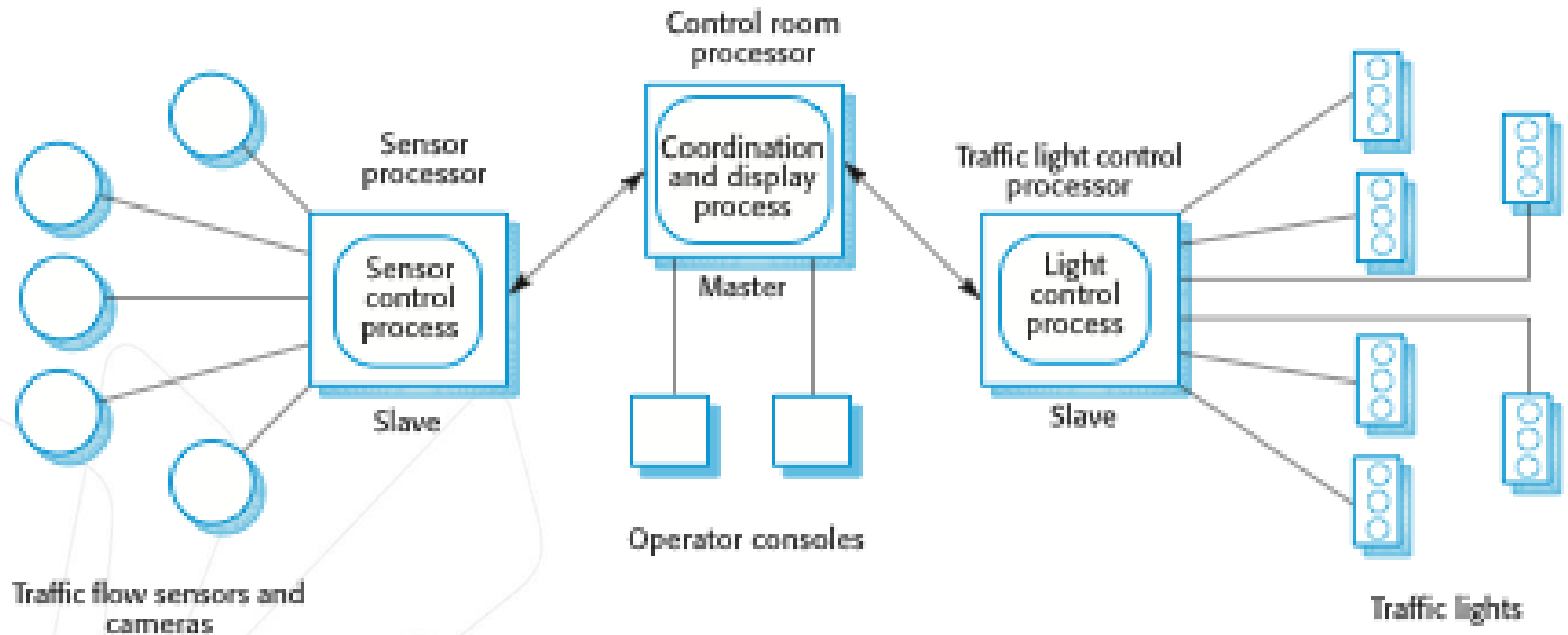


Master-slave 스타일 (분산 시스템 architecture 중 하나)

- Master-slave architectures are commonly used in real-time systems where there may be **separate processors** associated with **data acquisition** from the system's environment, **data processing** and **computation** and **actuator management**.
- The 'master' process is usually responsible for computation, **coordination and communications** and it controls the 'slave' processes.
- 'Slave' processes are dedicated to **specific actions**, such as the acquisition of data from an array of sensors.



A traffic management system with a master-slave architecture



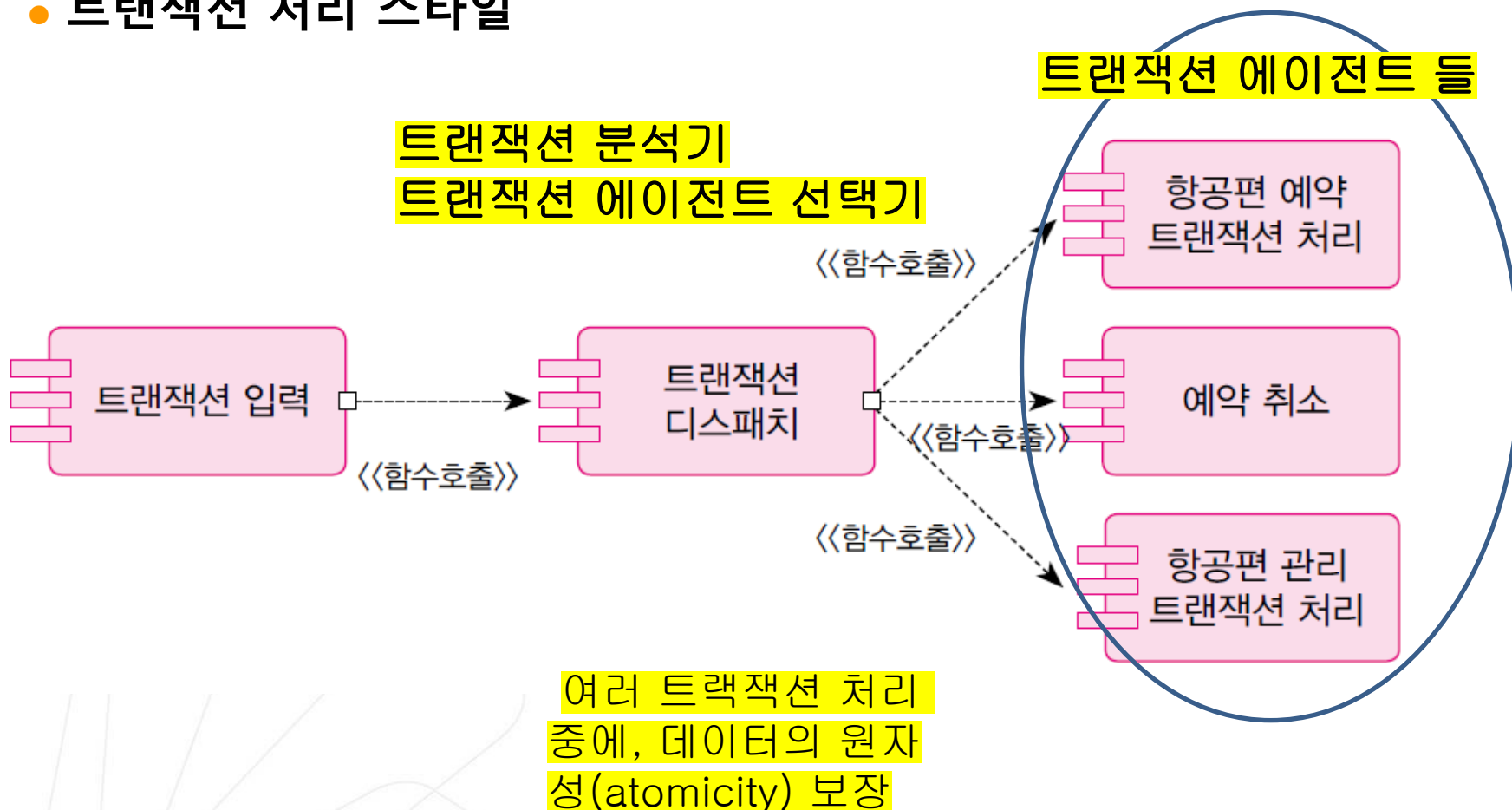
트랜잭션 처리 스타일

- 트랜잭션 처리 아키텍처는 입력을 하나씩 읽어 처리한다. 입력은 시스템에 저장되어 있는 데이터를 조작하는 명령들, 즉 트랜잭션이다. 일괄 처리(batch processing)에 적합
- 트랜잭션을 어디서 처리하는지 결정하는 디스패처(Dispatcher)라고 하는 교통정리 컴포넌트가 필요하다.
- 디스패처는 프로시저 호출이나 메시지를 통하여 요청된 트랜잭션을 처리할 컴포넌트에 배치한다.



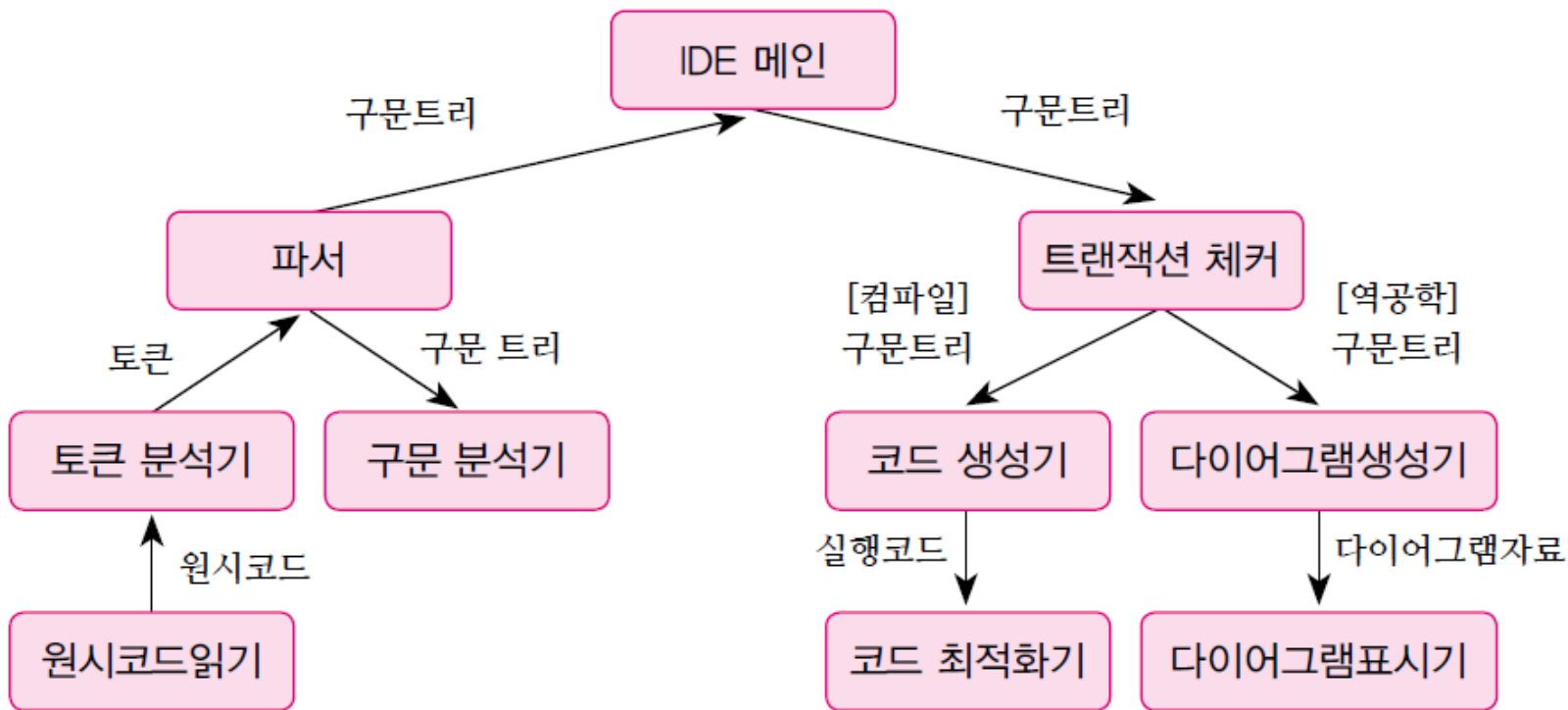
트랜잭션 스타일

● 트랜잭션 처리 스타일



트랜잭션 스타일

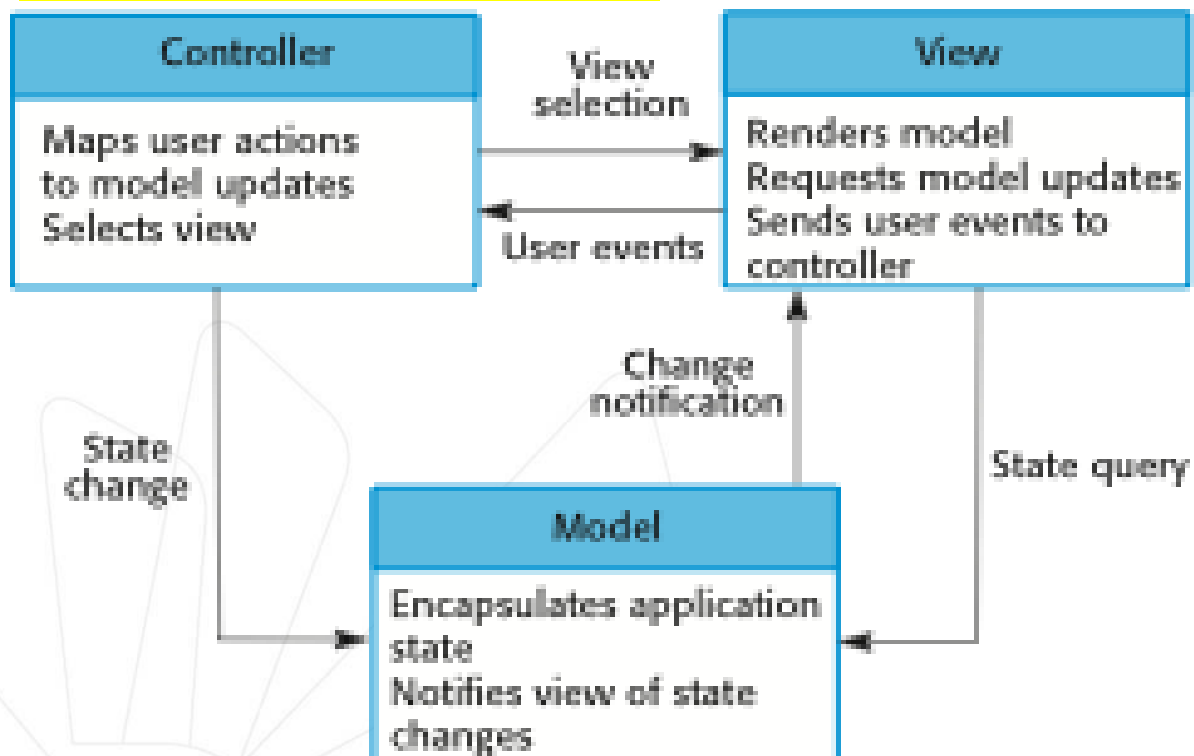
- 트랜잭션 처리 아키텍처 사례(IDE)



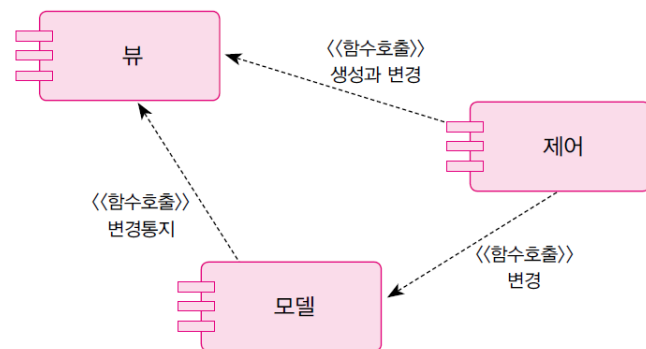
MVC 스타일

- MVC(Model-View-Controller) ◀ 특별한 계층 구조 스타일
사용자 인터페이스를 시스템의 다른 부분과 분리하여, 다른 UI를 만들고
그들 사이의 결합도를 낮추기 위한 아키텍처 스타일이다.

for user input. decide what to do

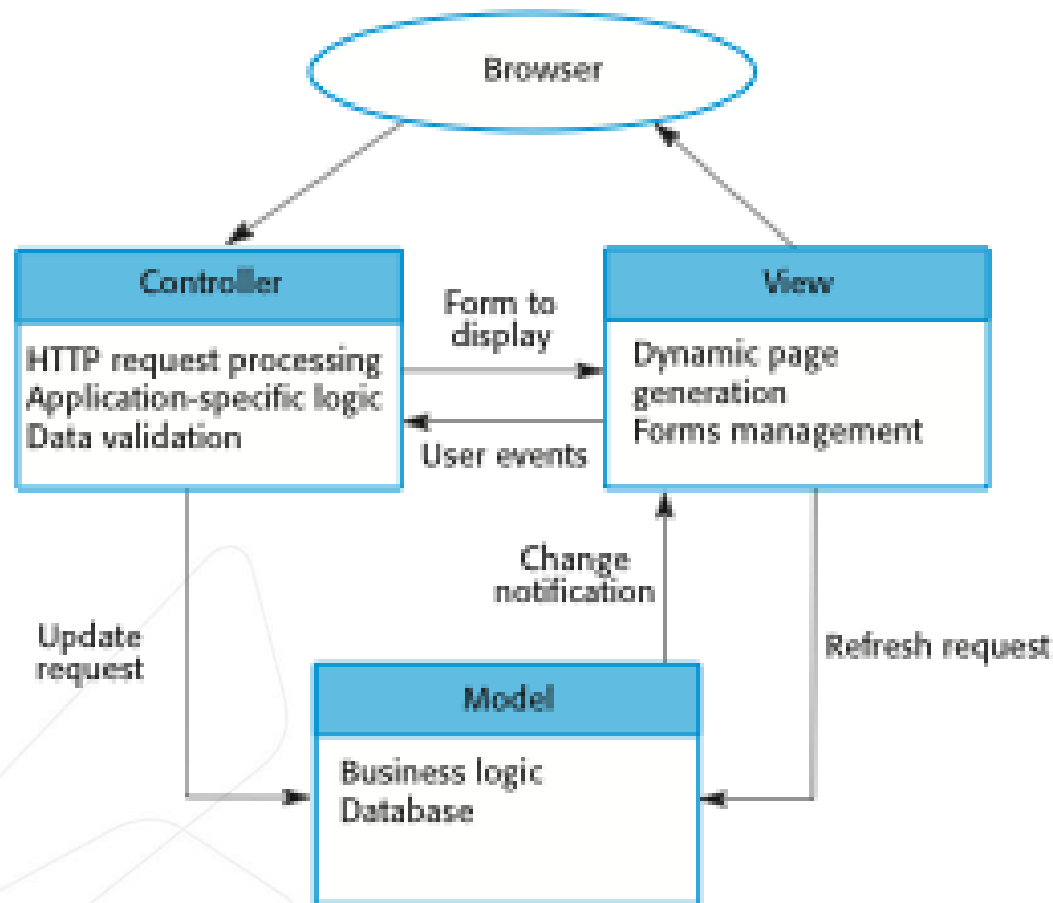


contain data and business logic



MVC 아키텍처 스타일

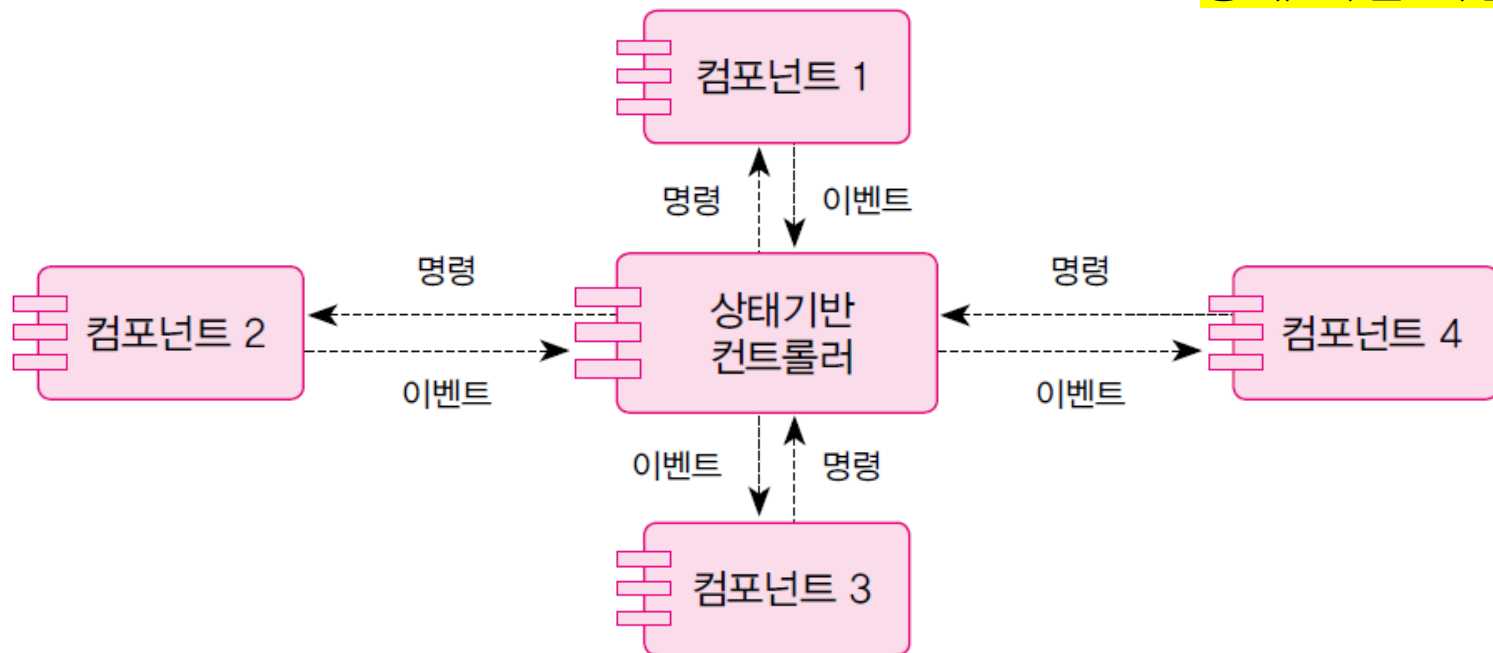
ex) a web application using MVC



이벤트 중심 스타일

- 이벤트 중심 시스템 아키텍처는 상태 기반 컨트롤러와 제어 대상이 되는 여러 컴포넌트로 구성된다.
- 이벤트 중심 아키텍처

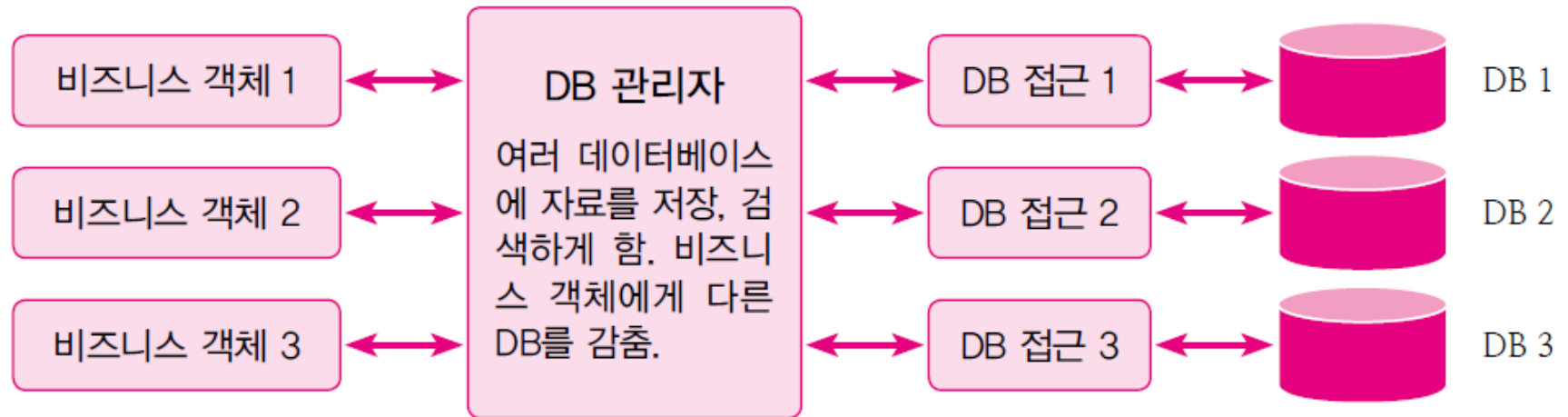
상태 머신 기반



객체 영속 스타일

- 비즈니스 시스템은 데이터베이스에 객체를 저장하고 나중에 이를 검색할 필요가 있음
- 이런 시스템을 객체 영속 시스템 (Object Persistence System)이라고 함

영속 시스템 프레임워크



- 비즈니스 객체에게
균일한 DB 접근 인터페이스 제공.
- 비즈니스 객체 와 DB 저장 컴포넌트
를 분리

서로 다른 DB
제품들

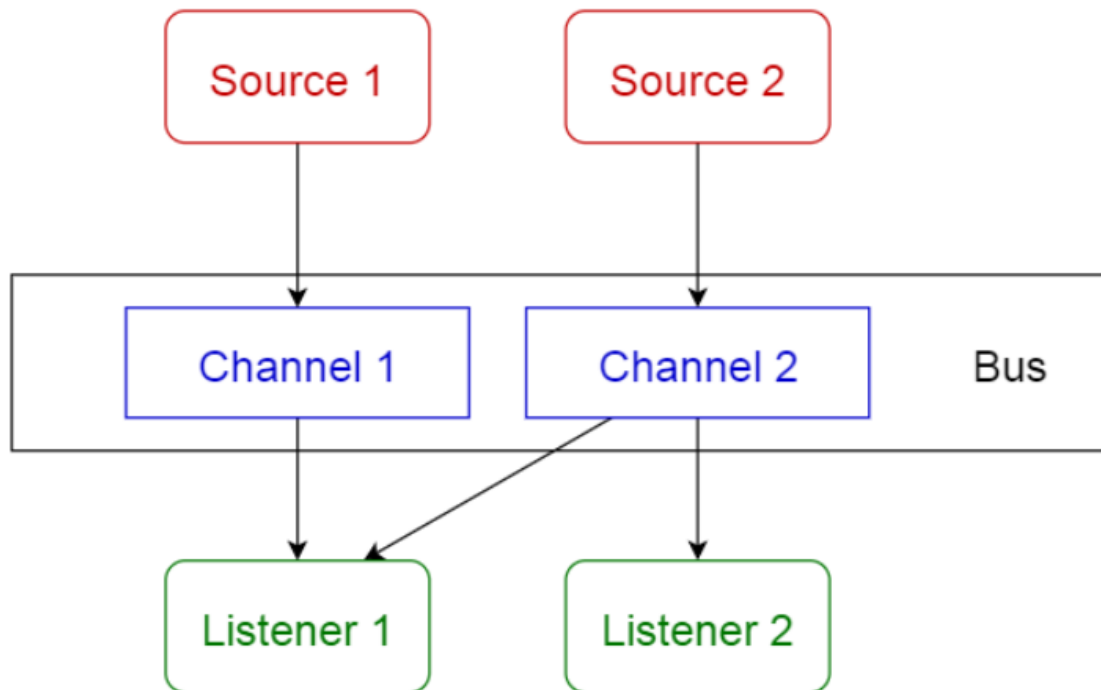
이벤트-버스 (Event-bus) 기반 스타일

소스(Source)는 이벤트 버스를 통해 특정 채널로 메시지를 발행하며 (publish), 리스너(Listener) 는 특정 채널에서 메시지를 구독한 (subscribe).

- 리스너는 구독하는 채널에 발행된 메시지에 대해 알림을 받음. 비동기적 정보 전달

Event-driven vs Message-oriented

- Event: event을 기다리는 listener가 있음
- Message: addressable recipient가 있음

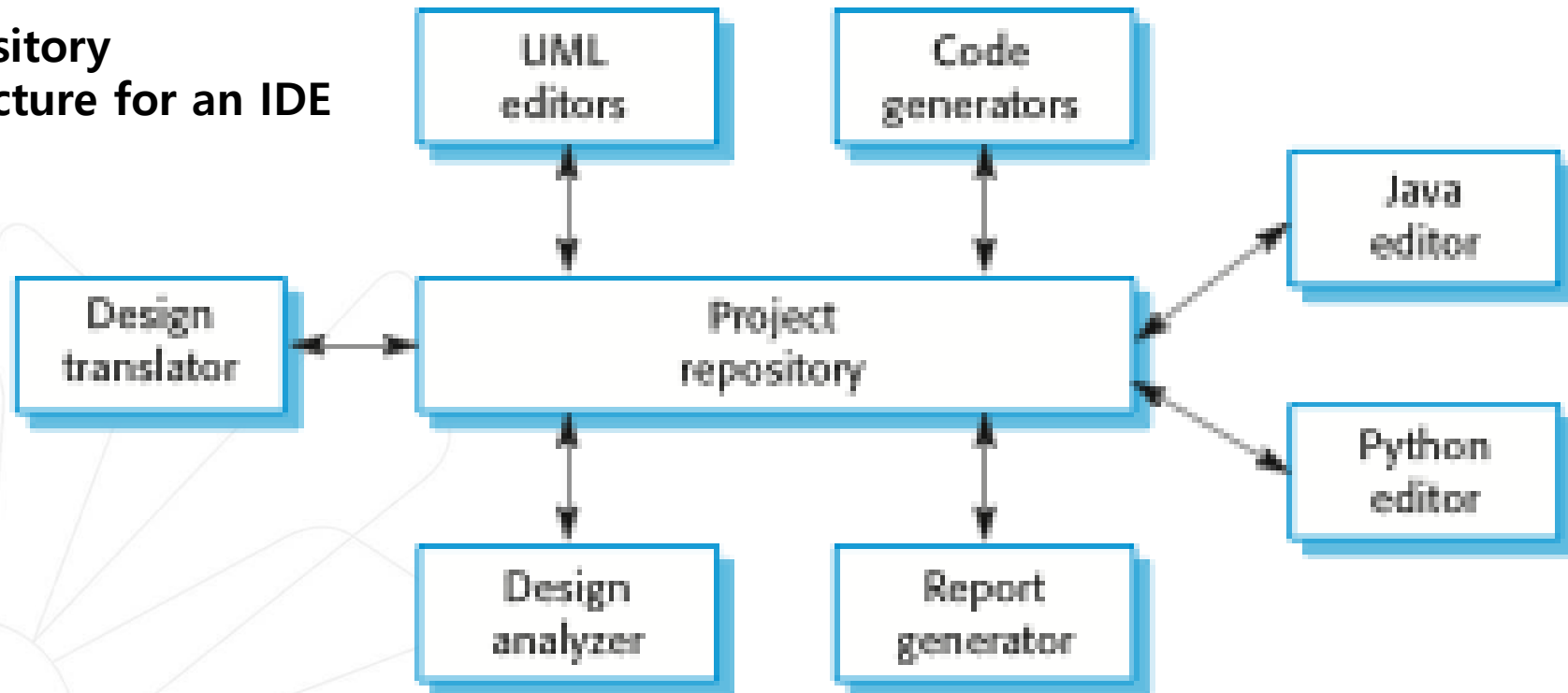


활용: 알람 서비스

블랙보드 (repository) 아키텍처

- The blackboard model was originally designed as a way to handle complex, **ill-defined problems** (eg. AI problems), where the solution is the sum of its parts.
- 활용: 음성 인식, 차량 식별 및 추적, 등

A repository architecture for an IDE



파이프-필터 (Pipe-Filter) 스타일

- 여러 컴포넌트들이 **데이터 스트림을** 생성하고 처리하는 시스템에서 사용
- 각 처리 과정은 필터 (filter) 컴포넌트에서 이루어지며, 처리되는 데이터는 파이프 (pipes)를 통해 흐름. 파이프는 버퍼링 또는 동기화 목적으로도 사용될 수 있음.
- 예제-1: 컴파일러: 연속한 필터들은 어휘 분석, 파싱(문법분석), 의미 분석, 코드 생성을 수행.


예제-2: UNIX 명령어 파이프 서브시스템: **cat** simple.txt | **grep** a | **sort**



- 확장성, 필터 재사용, 가장 느린 파이프에 속도 좌우됨



아키텍처 스타일들간 비교

스타일	장점/특징	단점 
계층	변경용이, 계층표준화	응용이 제한적
클라이언트-서버	클라이언트가 경량(slim)	서버 오버헤드
트랜잭션 처리	트랜잭션의 원자성 보장	동기화 (멀티 쓰레드 실행 시) 오버헤드
MVC	멀티뷰 지원, 개발자 협력 용이	초보자 이해 어려움. 불필요 갱신
이벤트 버스	발행자-구독자 연결 단순	이벤트를 필터링해 구독자에게 전달하는 오버헤드 발생

스타일	장점/특징	단점
객체 영속성	개체 영속성	오버헤드
블랙보드	기능 추가 변경이 용이	제어 어려움
피어 투 피어	분산 처리	품질 보장이 어려움
파이프 필터	확장성, 필터 재사용	가장 느린 필터에 속도 좌우
브로커	객체의 추가, 삭제, 변경 용이	표준화 필요

