Spring 2023

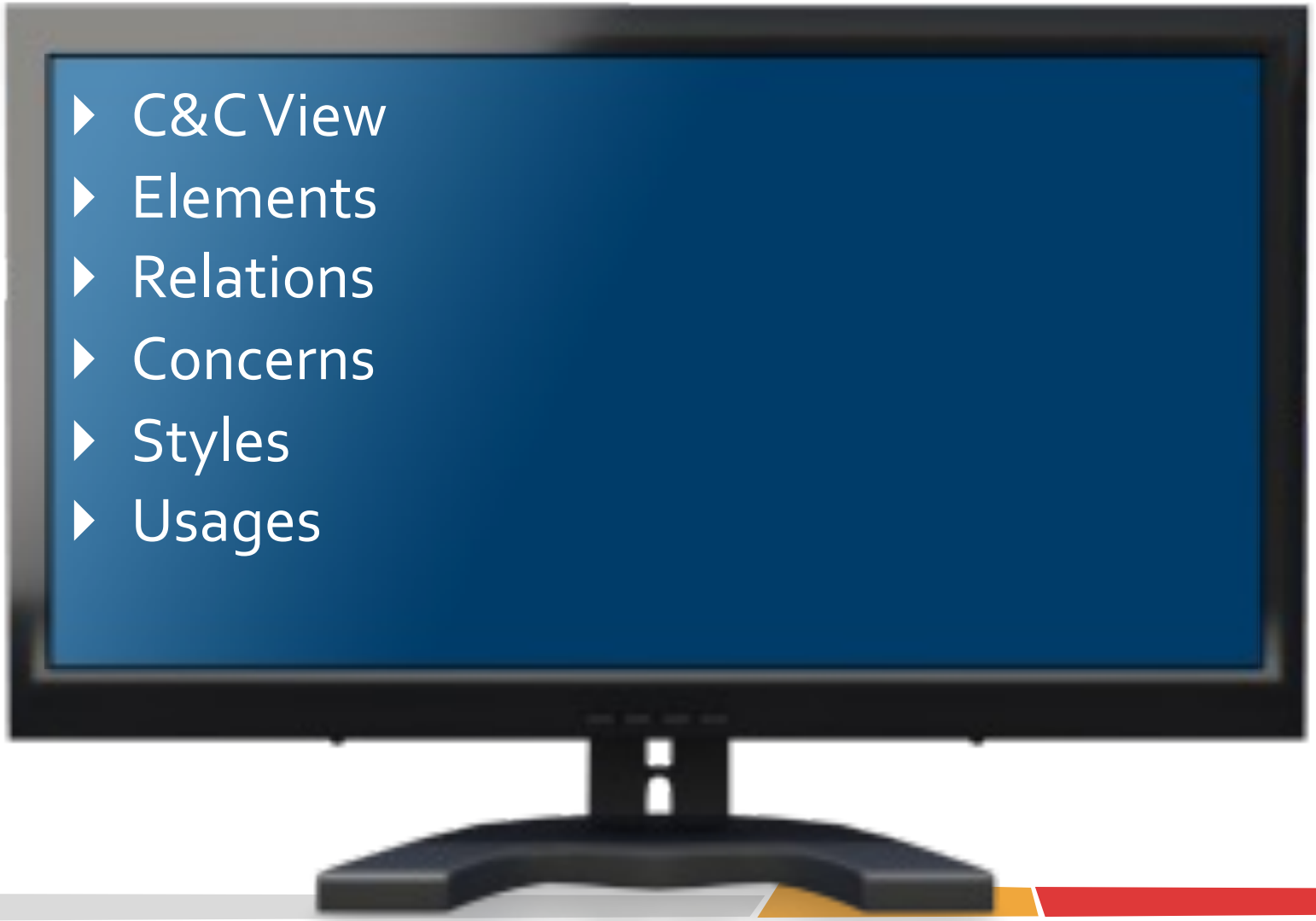
C&C View and Allocation View

Seonah Lee

Gyeongsang National University

목 차

C&C View



C&C View

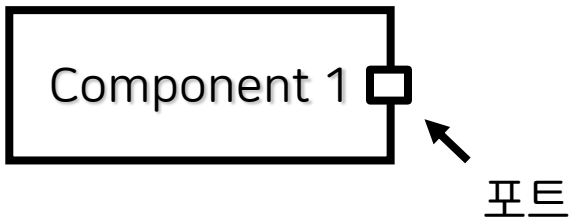
- **Component and Connector 뷰(C&C View)**
  - 실행시간 개체와 잠재적인 상호작용 표현
  - 실행시간에 존재하는 요소
    - 프로세스, 오브젝트, 클라이언트, 서버, 데이터 스토어
  - 상호작용의 경로
    - 통신 링크, 프로토콜, 정보 흐름, 공유 저장소에 대한 접근
    - 상호작용은 복잡한 기반구조**(infrastructure)**를 통해서도 실행 가능
  - 동일한 컴포넌트 타입의 다중 인스턴스 포함
    - 오브젝트 다이어그램 또는 협업 다이어그램과 유사
  - 적절한 상호작용 형식의 결정이 중요

Components

- **Run-time**과 연관되는 핵심 단위
  - 시스템에서 실행되는 주요한 컴퓨팅 요소와 데이터 저장소
- **Components define the locus of computation**
  - **Examples: filters, databases, objects, ADTs**

Ports

- 컴포넌트의 인터페이스
- 외부 환경과의 잠재적인 상호작용 지점
  - 실행시간 특성을 강조하고 설계 요소들의 인터페이스와 구별하기 위해 포트라고 명명
- 포트의 수와 타입은 명확히 문서화
  - 컴포넌트는 동일하거나 상이한 타입을 가진 다수의 포트를 포함

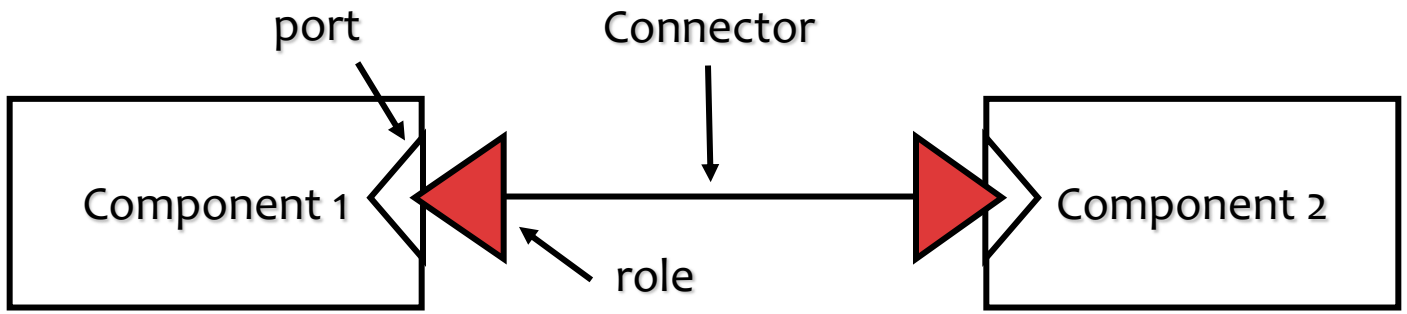


Connectors

- **Connectors: mediate interactions of components**
  - **Examples: procedure call, pipes, event broadcast**
- **Component들 간의 연동 Mechanism을 표시**
  - 커넥터가 지원하는 상호작용의 특징 정의
  - 커넥터의 형태를 명확하게 정의
    - 얼마나 많은 컴포넌트가 상호작용에 참가 가능한가
    - 지원하는 인터페이스의 수와 종류
    - 요구되는 요소
  - 프로토콜(protocol)
    - 커넥터에 의해 표현되는 상호작용
    - 상호작용에서 발생 가능한 이벤트 또는 액션의 패턴

Roles

- 커넥터의 인터페이스
- 상호작용을 수행하기 위해 컴포넌트에 의해 사용되는 방식 정의
- 일반적으로 상호작용 참가자의 기대 정의
  - 클라이언트-서버 커넥터 : invokes-service 역할, provide-service 역할
  - 발행자-구독자 커넥터 : publisher 역할, subscriber 역할



Attachments

- 어떤 커넥터가 어떤 컴포넌트에 부착되는가 표시
- 시스템을 컴포넌트와 커넥터의 그래프로 정의
- 컴포넌트 포트를 커넥터 역할과 연관
  - 컴포넌트 포트 **p**와 커넥터 역할 **r**의 부착 관계 성립
    - p에 의해 서술된 인터페이스를 사용하고
    - r에 의해 서술된 기대에 순응하면서
    - 컴포넌트가 커넥터를 통해 상호작용

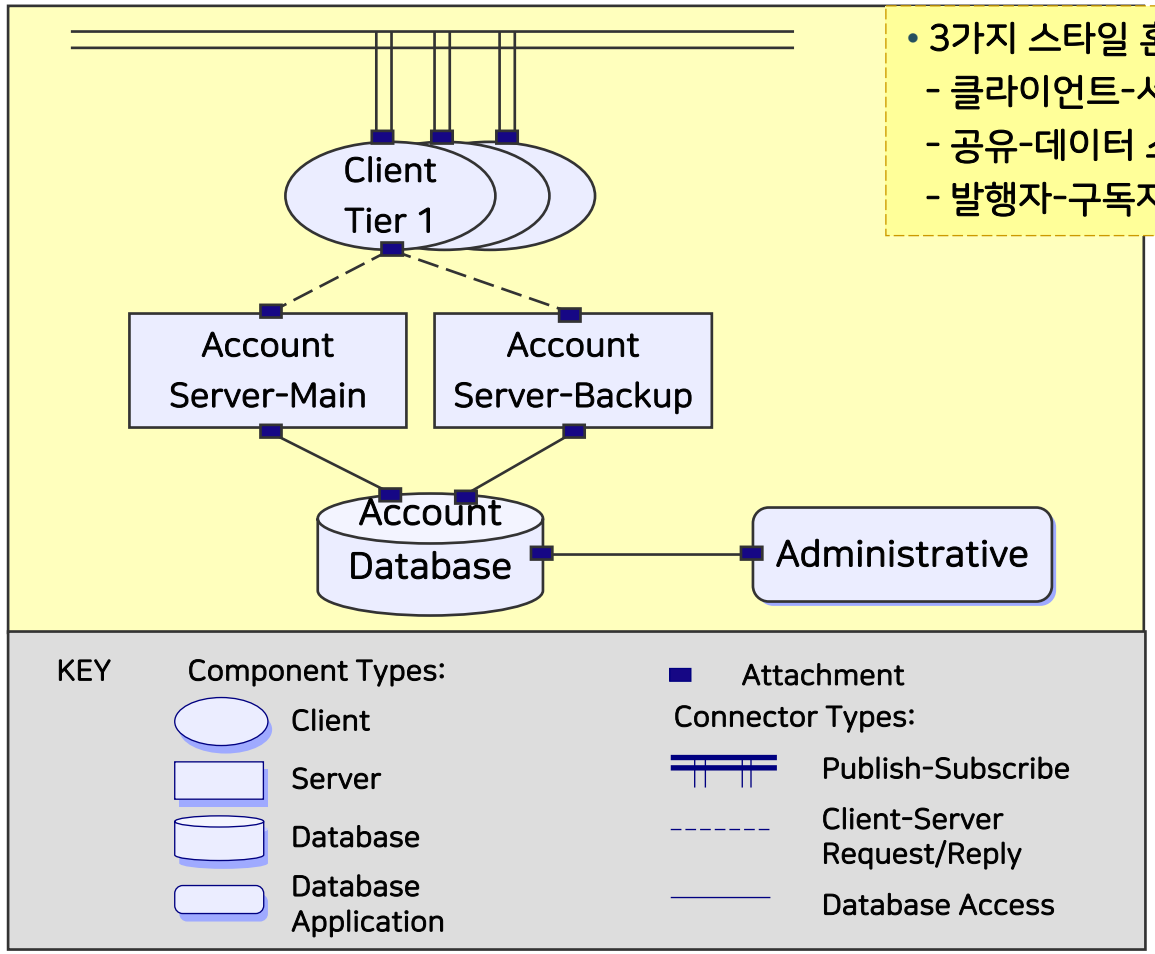
Concerns

- 시스템의 주요 실행 컴포넌트는 무엇이고 어떻게 상호작용하는가?
- 중요 공유 데이터 저장소는 무엇인가?
- 시스템의 어떤 부분이 얼마나 중복되어 있는가?
- 실행 시에 데이터가 어떻게 진행되는가?
- 통신 개체들에 의해 어떤 상호작용 프로토콜이 사용되는가?
- 시스템의 어떤 부분이 병렬적으로 실행되는가?
- 실행 시에 시스템의 구조가 어떻게 변경되는가?

Styles

- 파이프-필터 스타일(**Pipe-and-Filter Style**)
- 공유-데이터 스타일(**Shared-Data Style**)
- 게시-구독 스타일(**Publish-Subscribe Style**)
- 클라이언트-서버 스타일(**Client-Server Style**)
- 피어 투 피어 스타일(**Peer-to-Peer Style**)

Examples



- 3가지 스타일 혼합
  - 클라이언트-서버 스타일
  - 공유-데이터 스타일
  - 발행자-구독자 스타일

- 관련된 지원 문서화에 있어 핵심
- 한번에 이해될 수 있도록 단순화
- 어휘집을 구성하는 컴포넌트와 커넥터 타입을 명시적으로 선언
- 컴포넌트와 커넥터에 대한 인터페이스의 수와 종류에 관한 논의에 있어 핵심 역할
- 구현 메커니즘 보다는 어플리케이션 기능에 집중하는 컴포넌트와 커넥터 타입 사용

Usages

- 실행시간 시스템 품질 속성 판단
  - 성능(**performance**), 신뢰성(**reliability**), 가용성(**availability**) 등
  - 신뢰성(**Reliability**)
    - 해당 컴포넌트와 커넥터의 주요 실패 원인은 무엇인가**?**
    - 용도 **-** 전체 시스템 신뢰성 결정
  - 성능(**Performance**)
    - 어떤 로드 하에서 컴포넌트가 얼마의 응답시간(**response time**)을 보일 것인가**?**
    - 해당 커넥터에 대해 어떤 지연(**latency**)과 처리량(**throughput**)이 기대되는가**?**
    - 용도 **-** 지연, 처리량, 버퍼링 요구와 같은 시스템 요소 결정

Usages

- 아키텍트로 하여금 전체 시스템 요소를 예측 가능하도록 함
  - 개별적인 요소와 상호작용 요소에 대한 평가 및 측정치를 기반
- 설계 요소들을 표현하는 데는 부적합

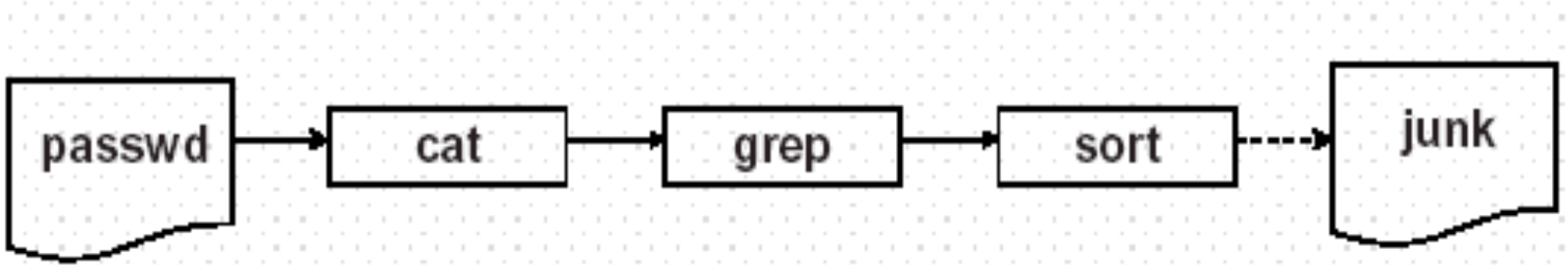
Pipe-and-Filter Style

Pipe-and-Filter Style

- 데이터 스트림의 연속적인 변환
  - 필터에 데이터 도착; 변환; 파이프를 통해 다음 필터로 전송
  - 전체 파이프와 필터 네트워크가 점진적으로 데이터를 처리
- **batch sequential** 스타일과의 차이점
  - **Batch-sequential** 스타일은 각 단계에서 다음 단계로 데이터를 전달하기 전에 데이터를 완전히 처리함
- **Not really suitable for interactive systems.**

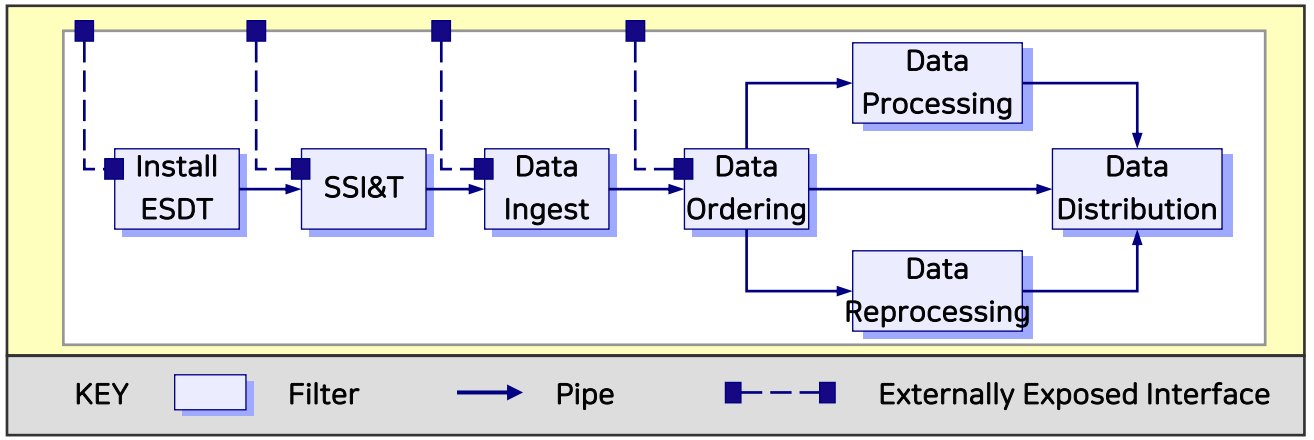
Pipe-and-Filter Style

- 제약 사항
  - 데이터는 한 방향으로 흐름
  - **Filter**간의 **state** 정보 공유가 필요 없음
  - 더 이상 계산이 없을 때까지 **Pipe**와 **Filter**를 실행한다 **(non-deterministically)**
- 예제: **Unix command system**
  - **cat etc/passwd | grep “joe” | sort > junk**

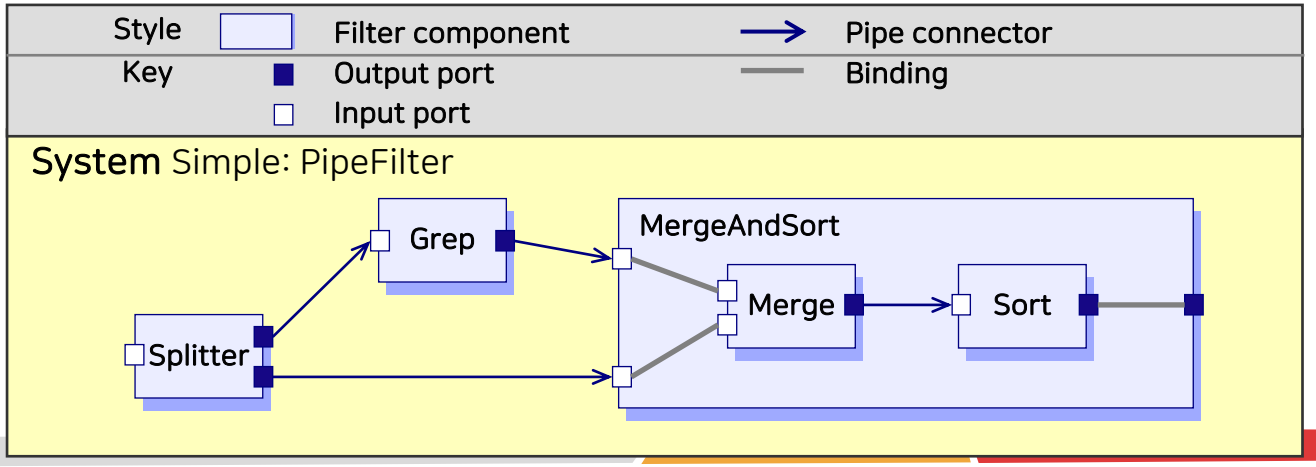


Graphical Notations

- ECS 시스템



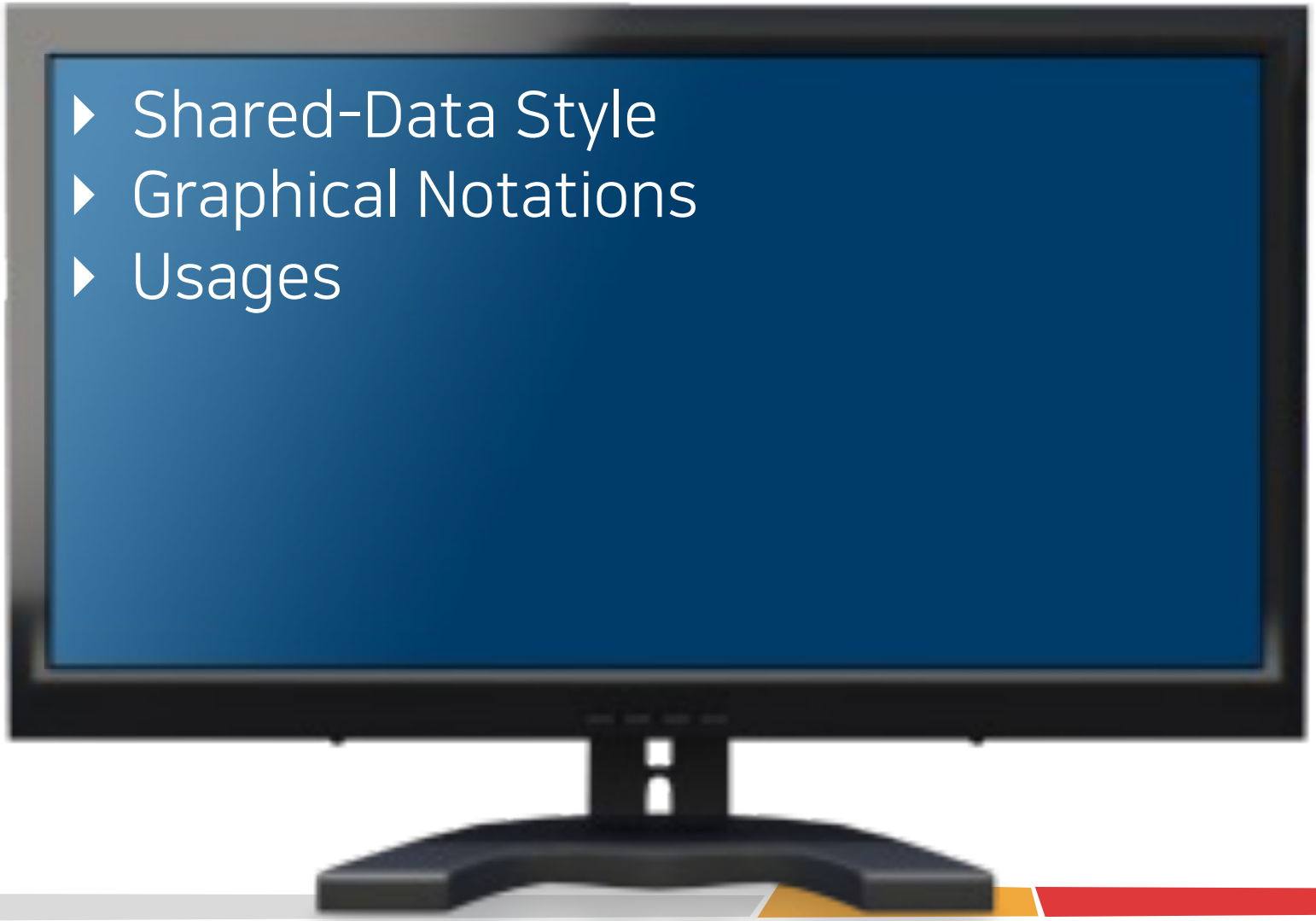
- String-Processing Application



Usages

- 시그널 프로세싱(**signal-processing**) 어플리케이션의 프론트-엔드 구성하여 데이터 변환
  - 첫 번째 필터는 센서로부터 데이터를 수신
  - 데이터 압축 후 두 번째 필터로 전송
  - 두 번째 필터는 다른 센서 간의 데이터를 종합
  - 마지막 필터는 다른 어플리케이션에 대한 입력으로 데이터 제공
- 필터 그래프에 의해 제공되는 종합적인 변환 주도
- 시스템 성능 추론
  - 입력/출력 스트림 지연, 파이프 버퍼 요구사항, 스케줄 가능성(**Schedulability**)

Shared-Data Style

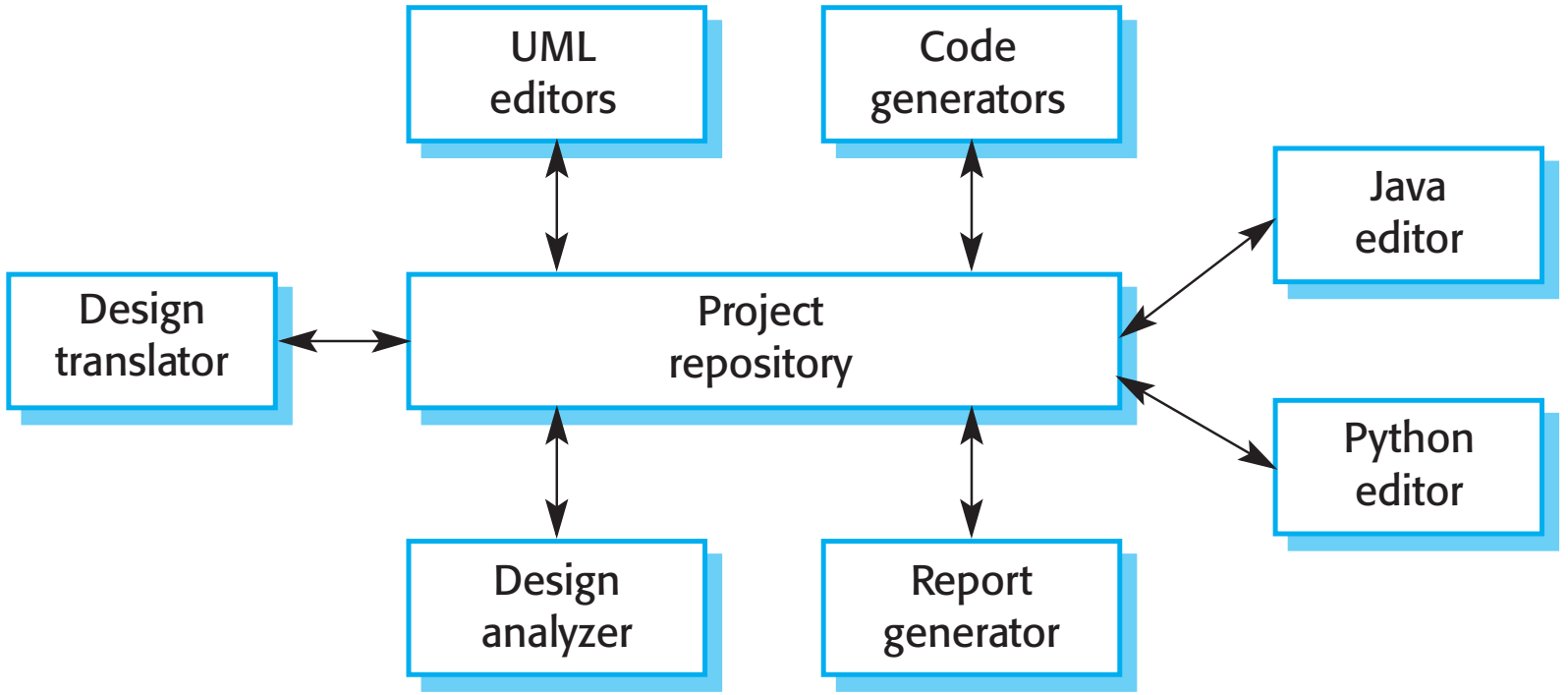


Shared-Data Style

- 상호작용 패턴은 지속성 데이터 교환에 의해 통제
- 다중 접근자와 하나 이상의 공유 데이터 저장소로 구성
- 데이터베이스 시스템, 지식 기반 시스템
- 데이터 소비자가 데이터를 찾는 방식에 따른 분류
  - **Blackboard**
    - 공유 데이터 저장소가 데이터 소비자에게 데이터가 도착했음을 통지
  - **Repository**
    - 데이터 소비자가 데이터 검색에 대한 책임을 짐

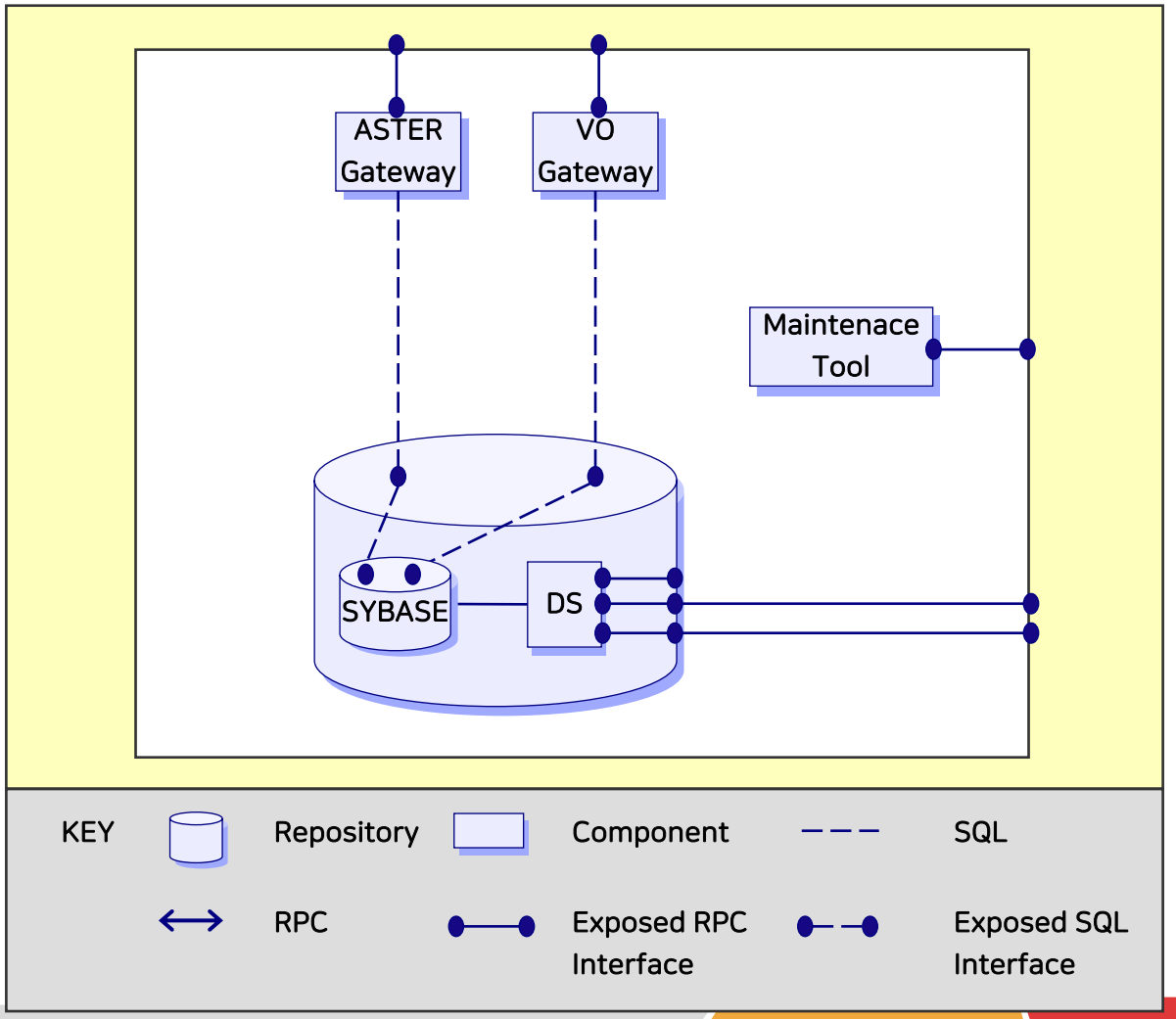
Graphical Notations

- A repository architecture for an IDE



Graphical Notations

- ECS 시스템

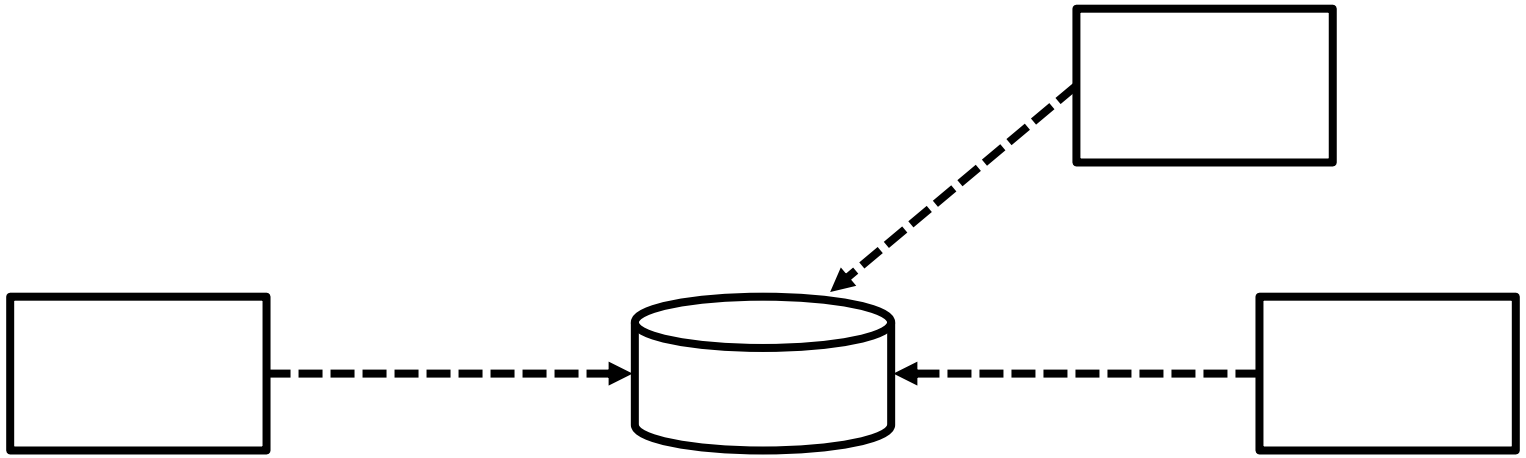


Gateway와 Maintenance Tool은 데이터 접근자

SYBASE와 DS는 데이터 저장소의 컴포넌트

Usages

- 지속성을 가진 다양한 데이터 항목들이 다중 접근자를 가지는 경우
  - 데이터의 생산자와 소비자간의 결합을 분리
  - 수정 용이성**(modifiability)** 향상



Usages

- 아키텍처 분석 시 고려사항
  - 다음을 집중적으로 분석
    - 성능**(performance)**
    - 보안**(security)**
    - 프라이버시**(privacy)**
    - 신뢰성**(reliability)**
    - 호환성**(compatibility)**
  - 하나 이상의 데이터 저장소를 가진 경우
    - 데이터와 계산을 데이터 저장소와 데이터 접근자에 매핑시키는 방법이 중요
  - 데이터 분산이나 중복
    - 성능은 향상시키지만 복잡성을 증가시키고 신뢰성과 보안을 감소

Publish-Subscribe Style

Publish-Subscribe Style

- 컴포넌트는 통보된 이벤트를 통해 상호작용
- 발행(**Publish**)
  - 이벤트를 통보하는 컴포넌트
  - 발행-구독 **run-time infrastructure**
    - 발행된 이벤트를 모든 구독자에게 전달
- 커넥터
  - 이벤트 버스
  - 컴포넌트는 이벤트를 버스에 위치
  - 커넥터는 적합한 컴포넌트에 이벤트를 전달

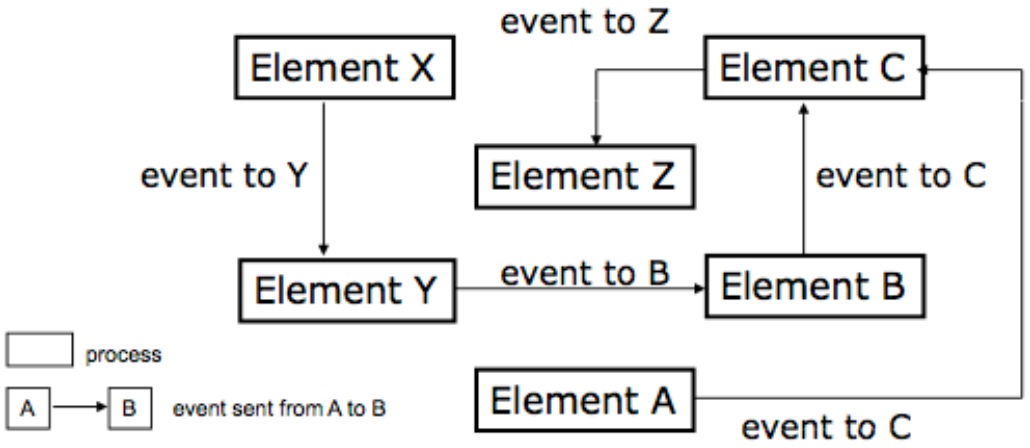
Publish-Subscribe Style:

- 발행-구독 스타일의 종류
  - 암시적 호출(**Implicit Invocation**)
    - 컴포넌트는 각 이벤트에 대해 연관된 프로시저를 등록
    - 이벤트 발행 시 등록된 프로시저 호출
  - 단순히 구독자에게 이벤트를 전달
    - 이벤트 처리 방법은 구독자가 해결
- 메시지 생산자와 소비자 간의 결합을 분리
  - 생산자와 소비자의 수정 용이성(**modifiability**) 향상

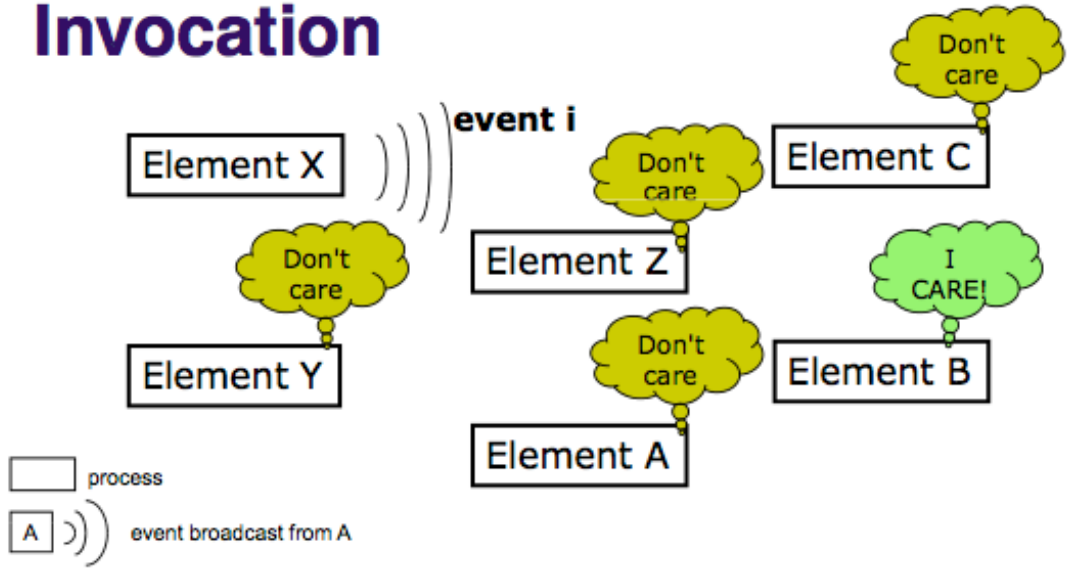
Publish-Subscribe Style

- **Explicit Invocation**
  - 변경 정보를 전달한 대상을 명확히 알고 변경 정보를 전달함
- **Implicit Invocation**
  - 변경 정보를 전달할 대상을 명확히 알지 않고 정보를 전달함

Example 2 – Explicit Invocation

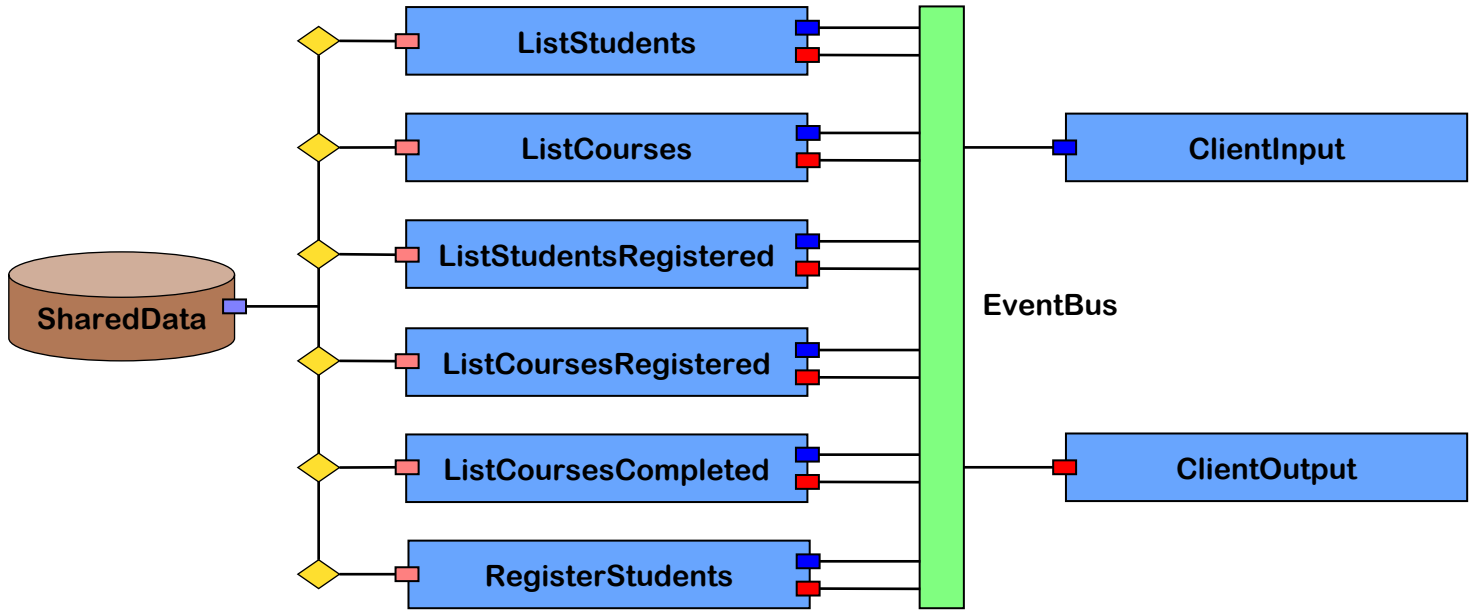


Example 1 – Implicit Invocation



Graphical Notations

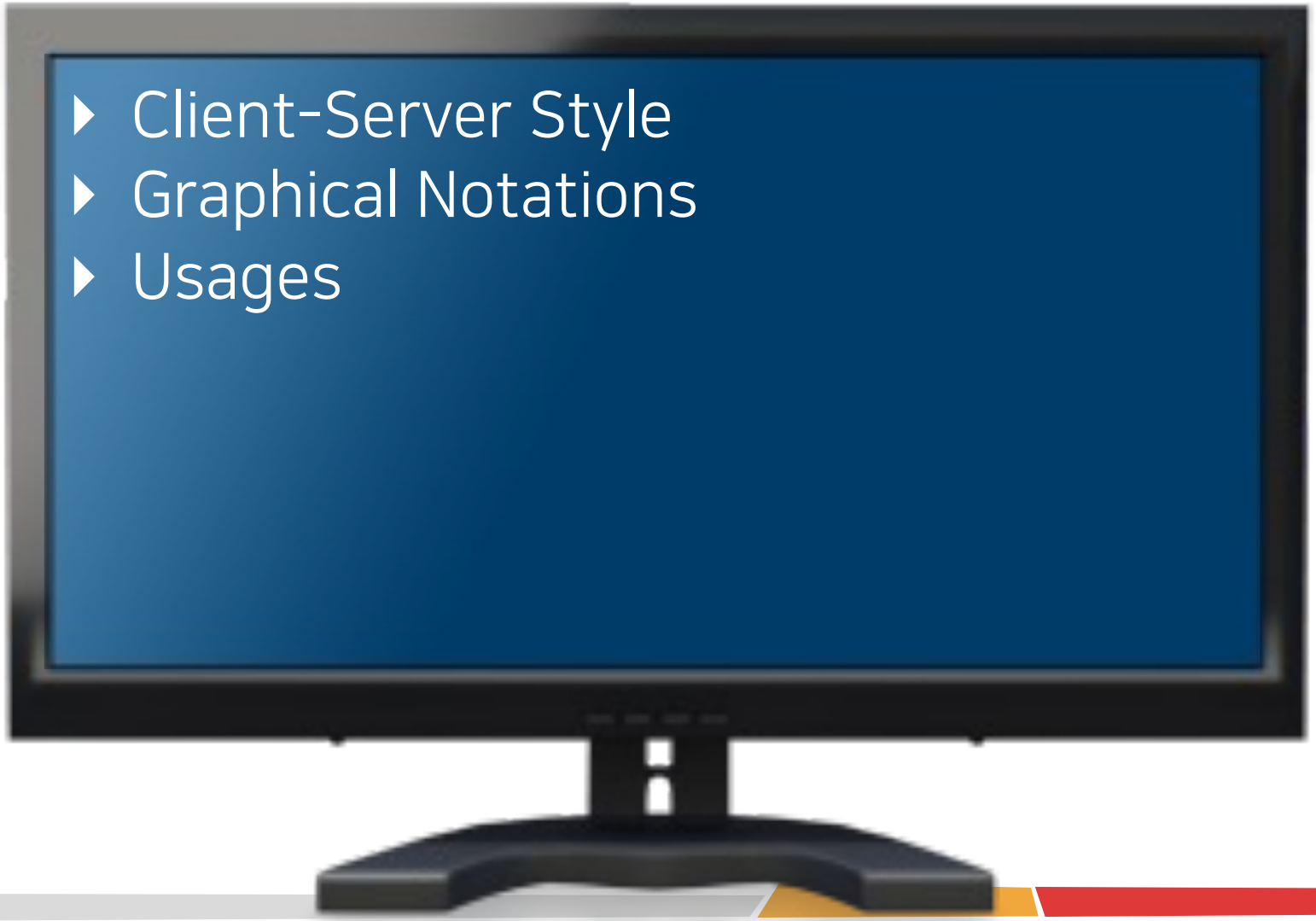
- **Components: Event** 발생 시 기능을 하는 모듈들
- **Connectors: Event** 전달을 목적으로 하는 **Event Bus**



Publish-Subscribe Style: Usage

- 알려지지 않은 수신자에게 이벤트와 메시지 전송
  - 프로시저 수정 없이 새로운 수신자 추가 가능

Client-Server Style

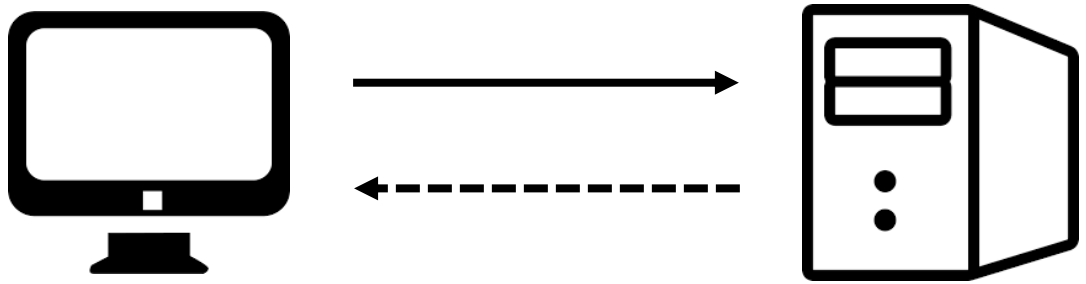


Client-Server Style

- 서비스(혹은 데이터)의 생산자, 소비자 간 결합 분리
- 한 컴포넌트가 다른 컴포넌트의 서비스를 요청
  - 통신은 한 쌍으로 수행
    - 클라이언트로부터의 서비스 요청
    - 서버로부터의 서비스 제공
  - 통신은 비대칭적
    - 클라이언트에 의해서 시작

Client-Server Style

- 서버
  - 하나 이상의 인터페이스를 통해 서비스 제공
  - 하나의 중앙 서버나 여러 대의 분산 서버
- 클라이언트
  - 다른 서버에 의해 제공되는 **0** 또는 그 이상의 서비스 사용

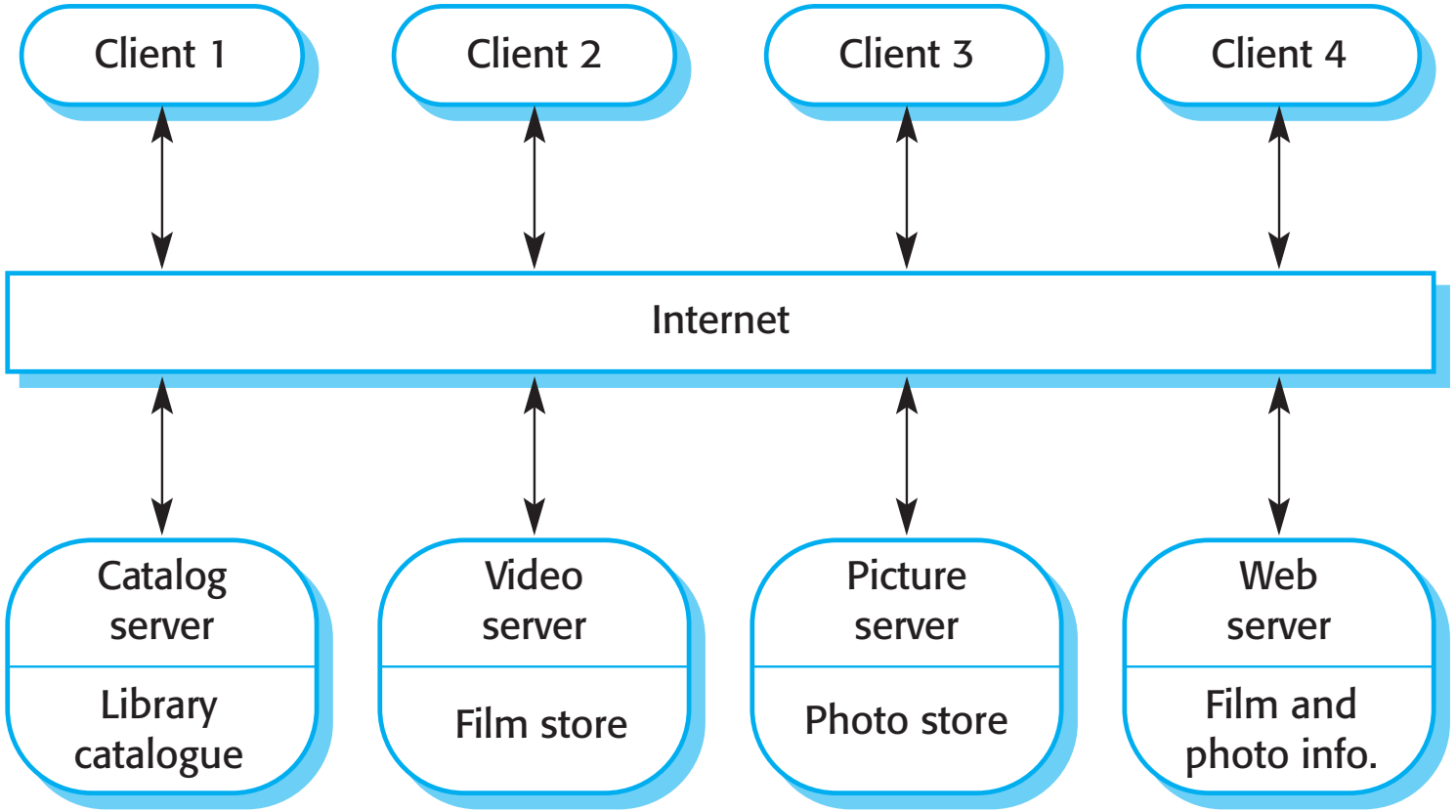


Client-Server Style

- 서비스와 데이터의 생산자, 소비자 간 결합 분리
- 클라이언트-서버 스타일의 예
  - 클라이언트 어플리케이션과 스크린 서버로 분할하는 윈도우 시스템
  - 클라이언트와 데이터로 분할하는 2-티어 데이터베이스 시스템
  - 클라이언트, 비즈니스 로직, 데이터 관리 서비스로 분할하는 웹-기반 정보 시스템

Graphical Notations

- A client–server architecture for a film library



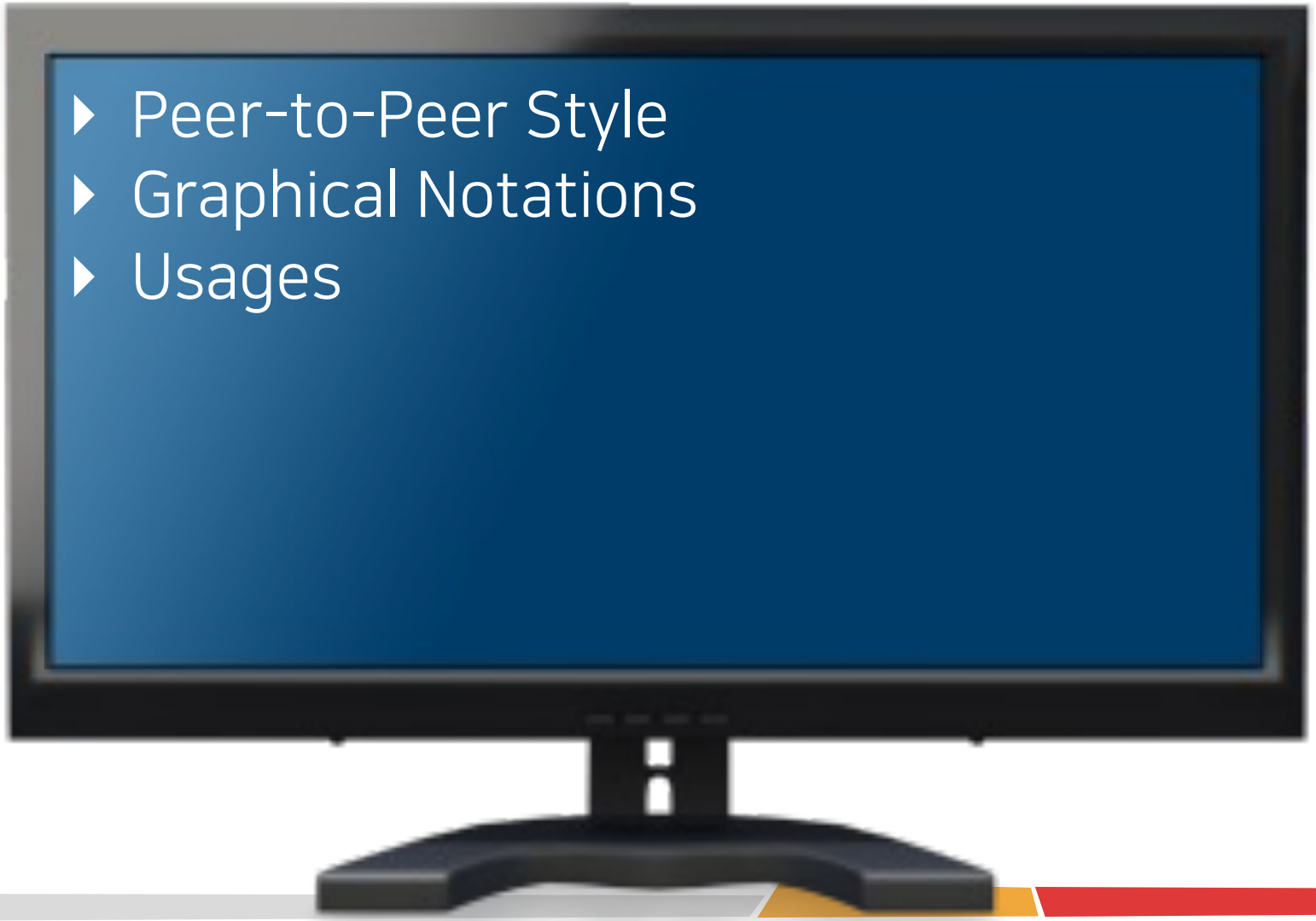
Usages

- 시스템 이해 촉진
  - 클라이언트 어플리케이션을 사용되는 서비스와 분리
- 기능 그룹핑
  - 하드웨어 상의 배치를 위한 기초 자료
  - 기존 시스템 서비스와의 상호연동을 위한 기초 자료
- 클라이언트와 서버로의 기능 분할
  - 독립적인 티어로 할당
  - 성능 개선과 신뢰성 지원

Usages

- 아키텍처 분석 시 고려사항
  - 서버가 클라이언트가 요구하는 서비스를 충분히 제공하는 지를 결정
  - 확실성(**dependability**)
    - 서비스 실패로부터 복구 가능한가
  - 보안(**security**)
    - 제공되는 서비스가 적절한 권한을 가진 클라이언트에게만 제공되는가**?**
  - 성능(**performance**)
    - 서버가 예상되는 서비스 요청의 양과 속도를 감당할 수 있는가**?**

Peer-to-Peer Style

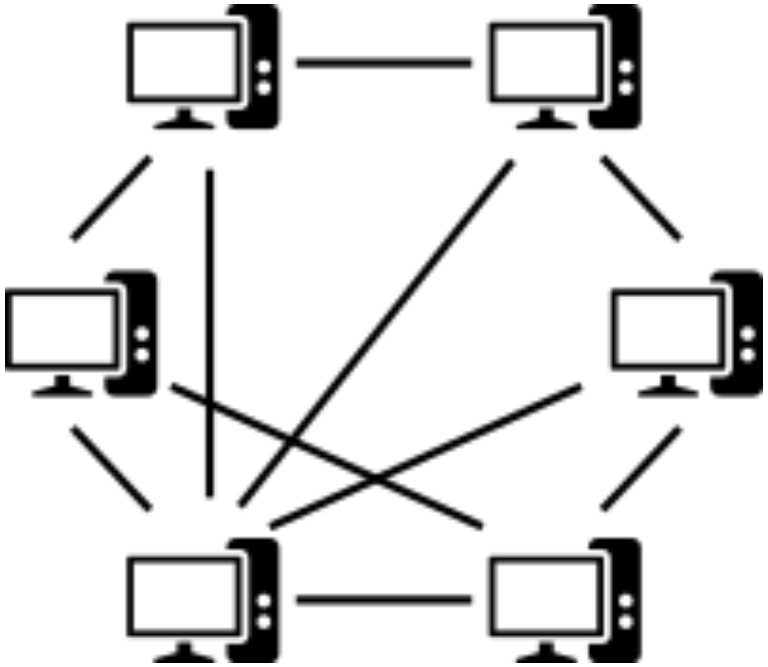


Peer-to-Peer Style

- 컴포넌트
  - 서비스를 교환하는 피어**(peer)**로서 직접 상호작용
  - 대칭적인**(symmetric)** 요청/응답**(request/reply)** 상호작용
    - 한 컴포넌트는 어떤 컴포넌트와도 상호작용 가능
- 커넥터
  - 복잡한 양방향 상호작용 프로토콜 포함
  - 둘 이상의 피어 투 피어**(peer-to-peer)** 컴포넌트 간에 존재하는 **2**방향 통신 반영
- 피어 투 피어 스타일의 예
  - 분산 객체 기반구조**(distributed object infrastructure)**에 기초한 아키텍처
    - **CORBA, COM+, Java RMI** (단, 분산 객체 기반구조가 **Peer-to-Peer** 스타일인 것은 아님)

Graphical Notations

- Peers share resources without the use of a centralized administrative system

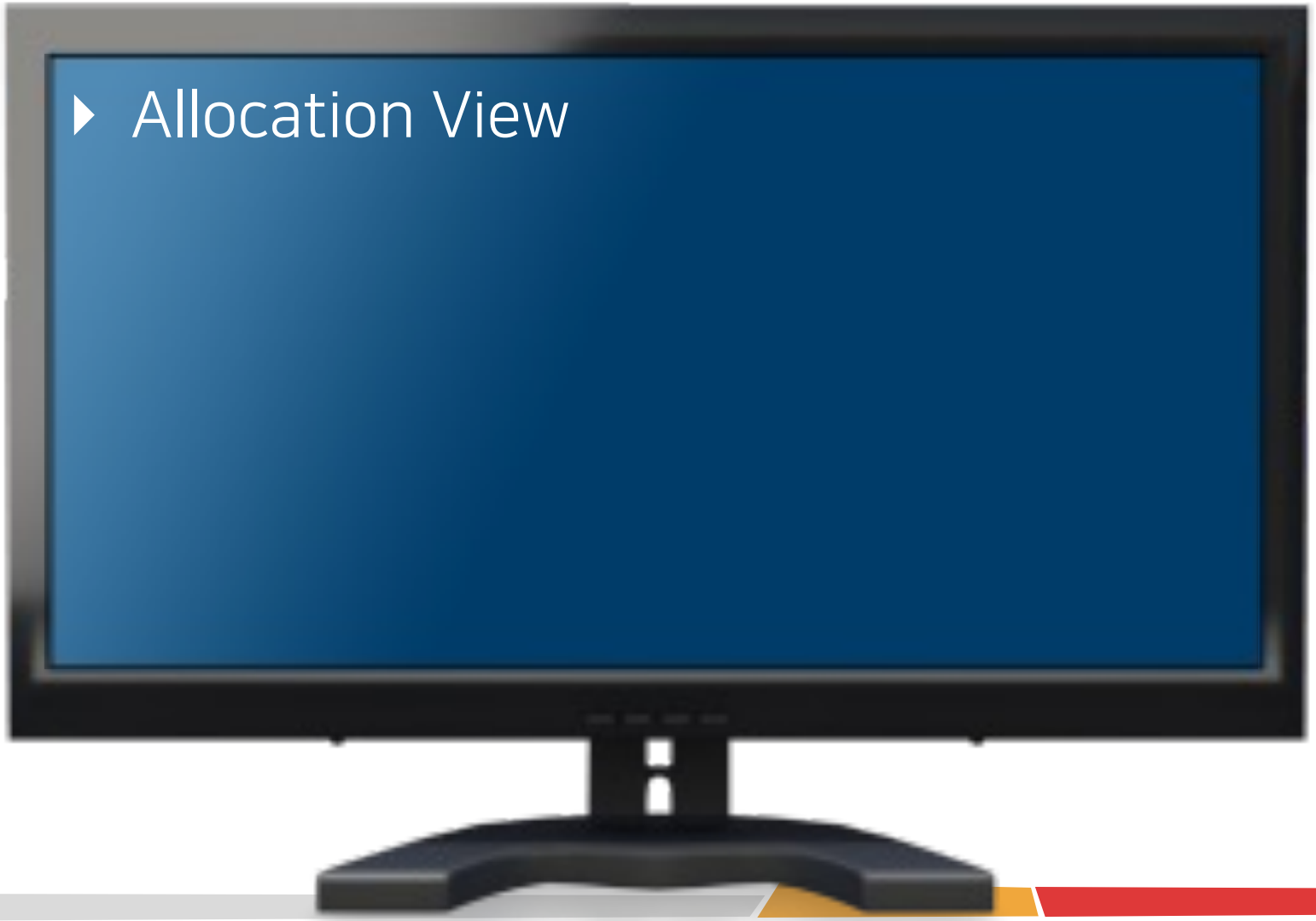


https://en.wikipedia.org/wiki/Peer-to-peer

Usages

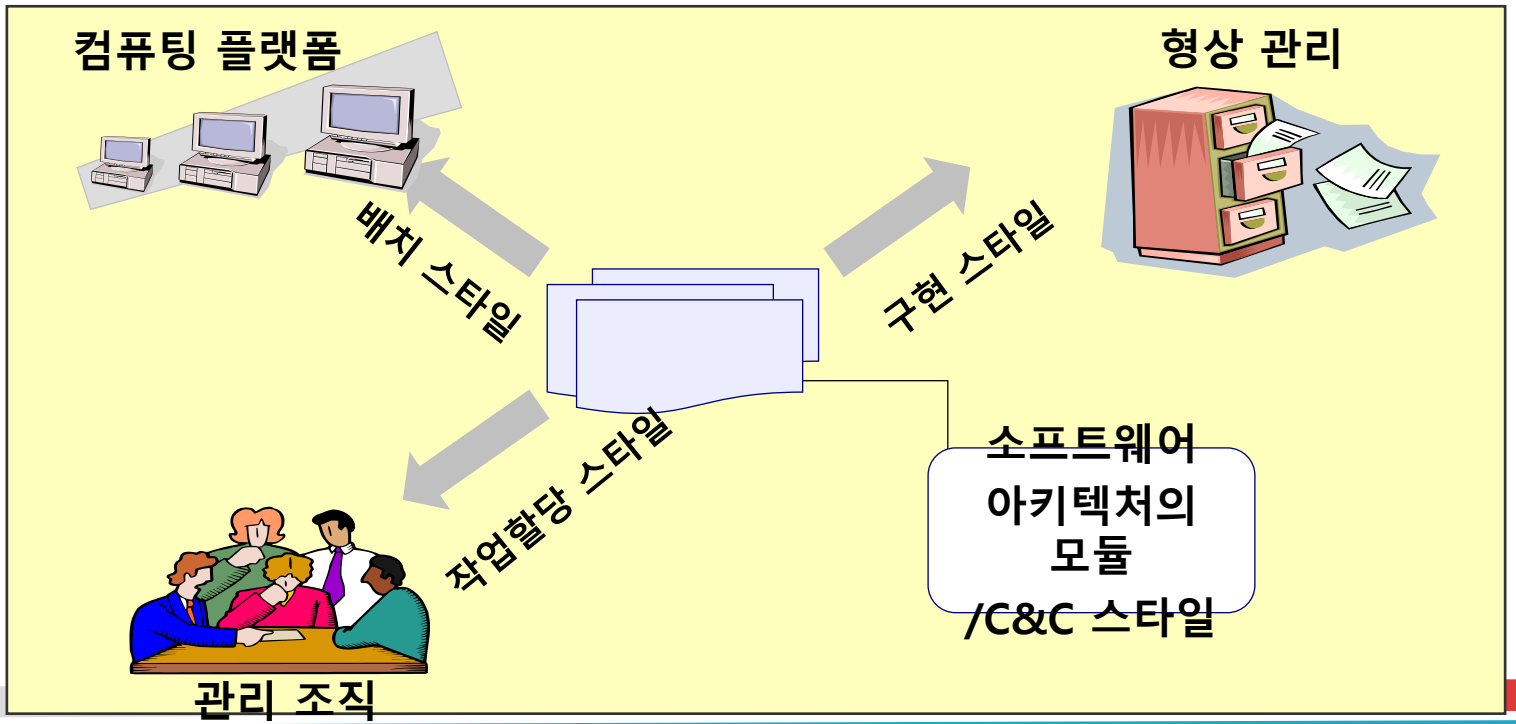
- 협력 영역에 의한 어플리케이션 분할 뷰 제공
  - 피어는 직접적으로 상호작용
  - 한 피어는 필요에 따라 클라이언트와 서버 역할 모두를 수행
  - 분할은 분산 시스템 플랫폼 상의 배치에 있어 유연성**(flexibility)**을 제공
- 분산 처리 시스템에 사용
  - **CPU**와 디스크 자원의 효율적인 사용
    - **CPU** 집중 작업을 네트워크 상의 컴퓨터에 분산

Allocation View

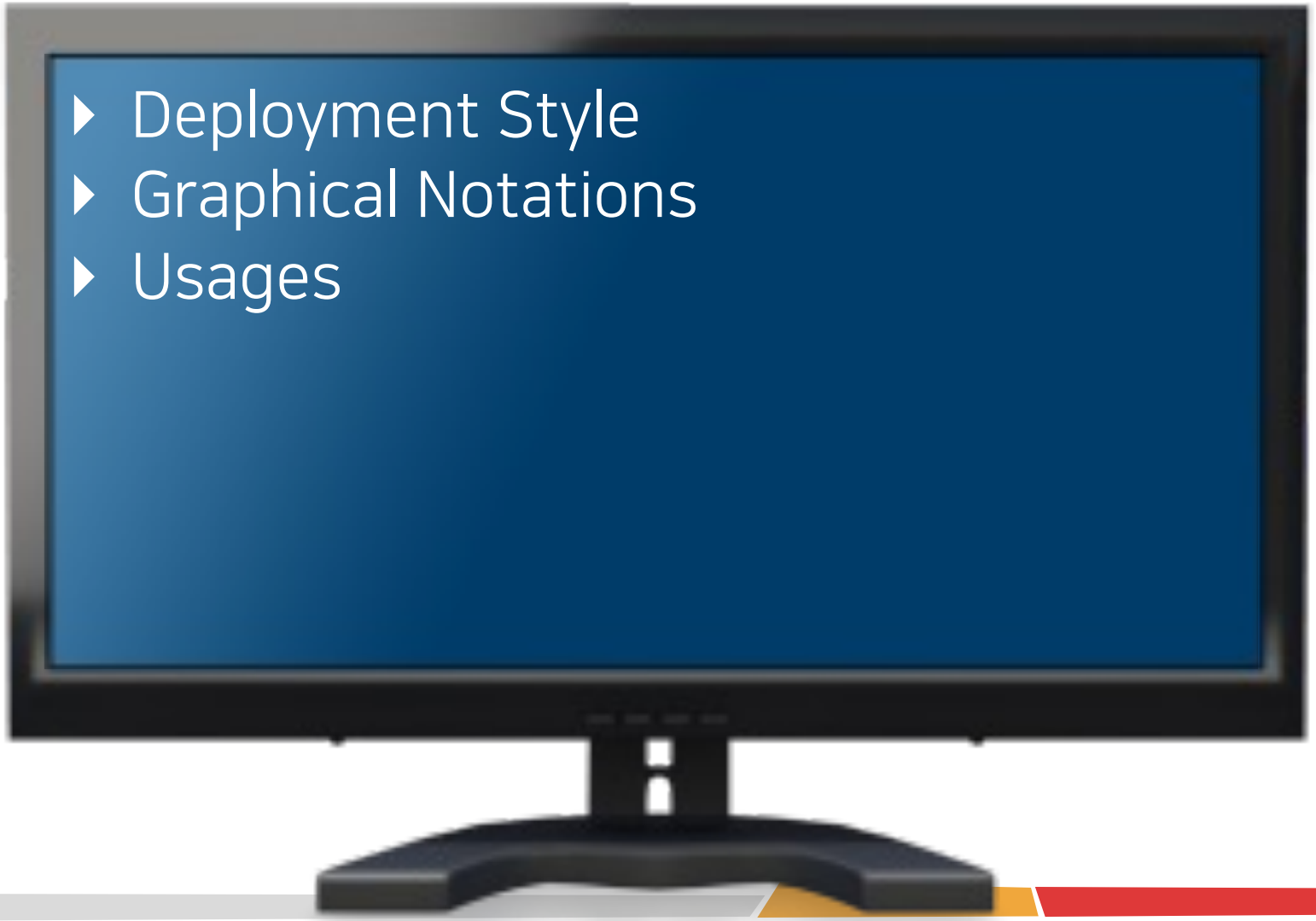


Allocation View

- 할당 뷰(Allocation View)
  - 소프트웨어 아키텍처와 외부 환경 간의 매핑
  - 소프트웨어 아키텍처는 하드웨어, 파일 시스템, 팀 구조와 상호작용



Deployment Style

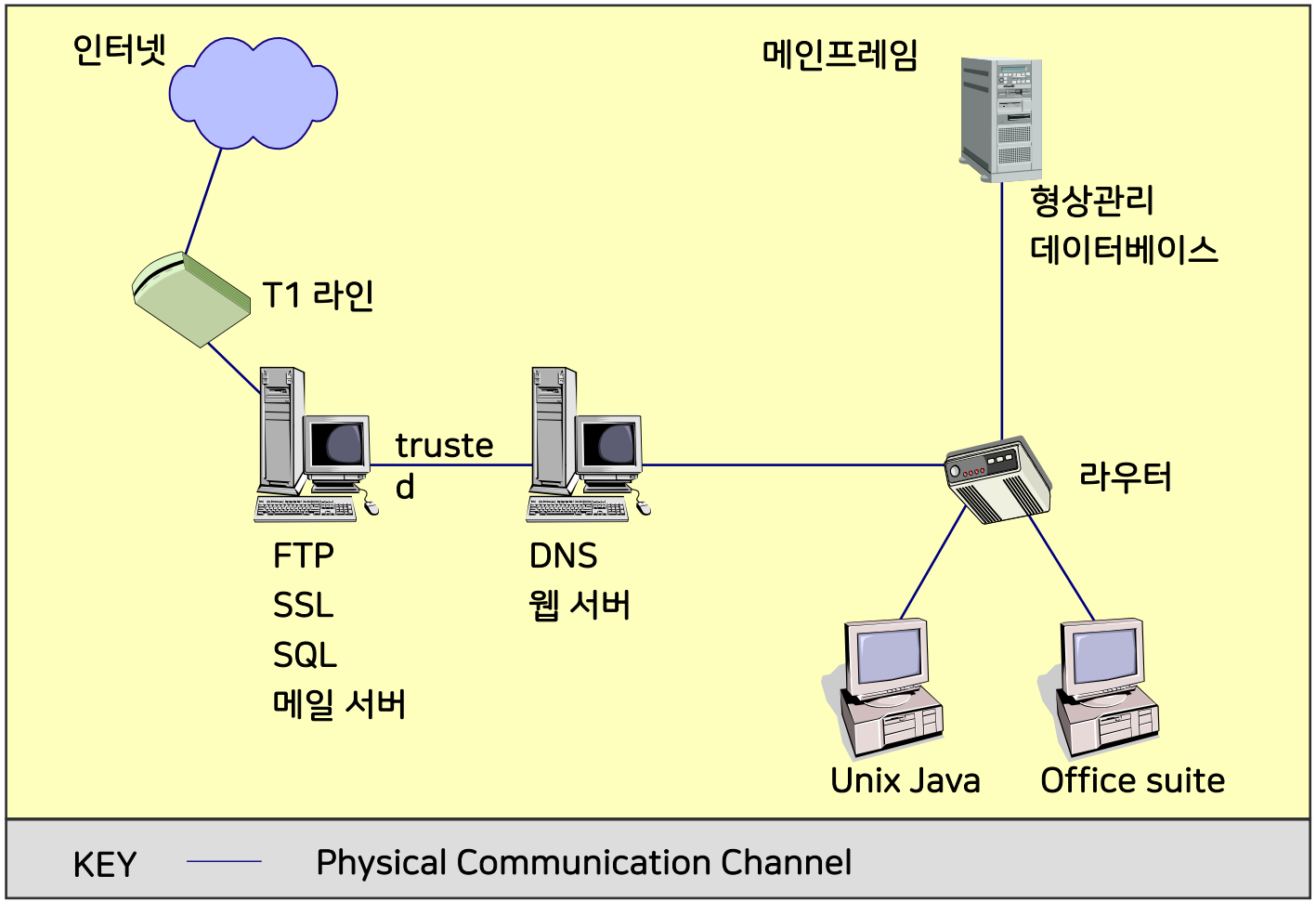


Deployment Style

- 컴포넌트-커넥터 스타일의 요소를 실행 플랫폼에 할당
  - 일반적으로 통신 프로세스 스타일**(Communicating-Processes Style)**요소를 할당
- 제약사항
  - 소프트웨어 요소의 요구사항
  - 적합한 하드웨어 요소의 특징에 의해 요구사항이 만족되는 방법

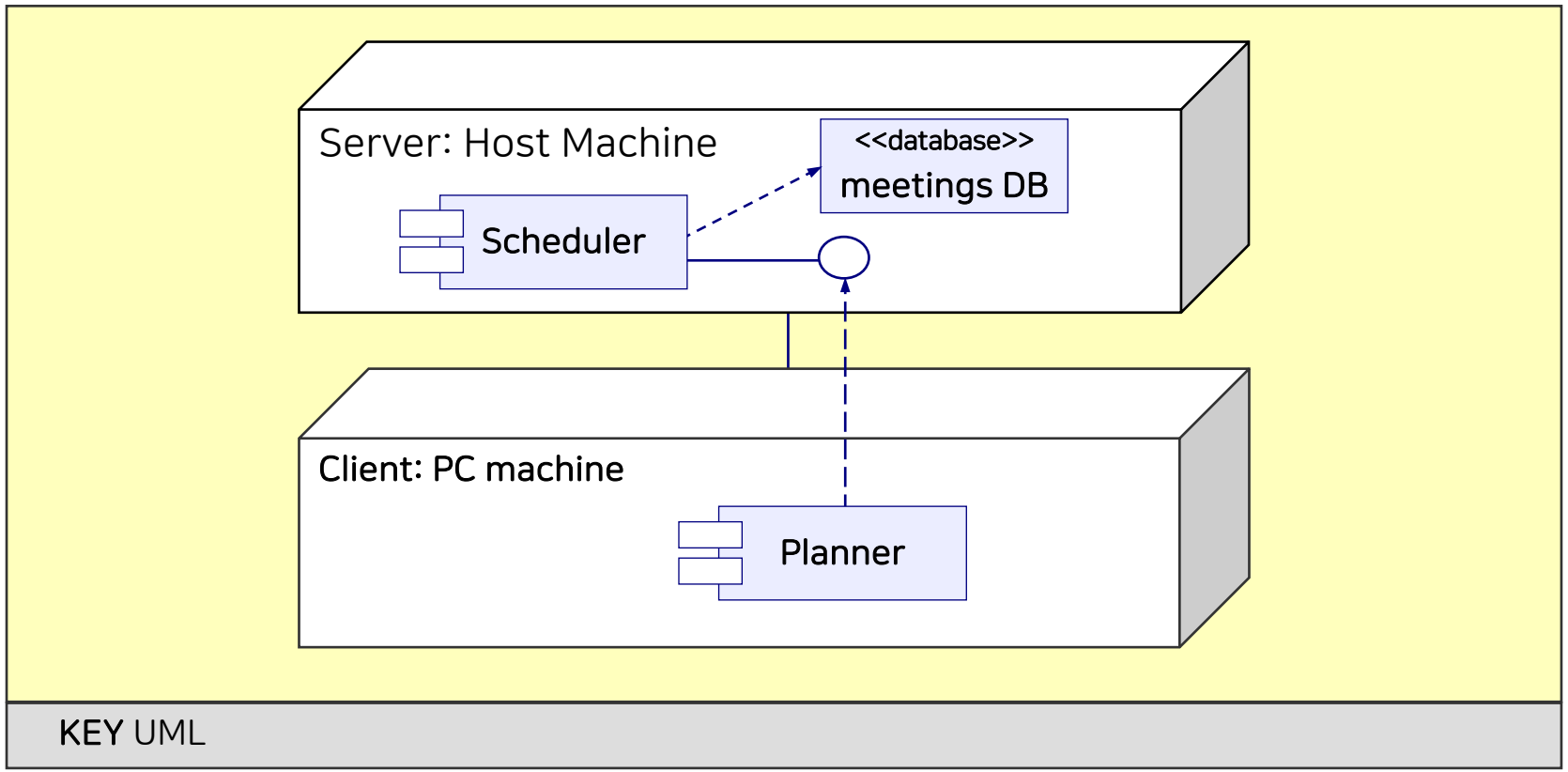
Graphical Notations

- 비정형적 표기법



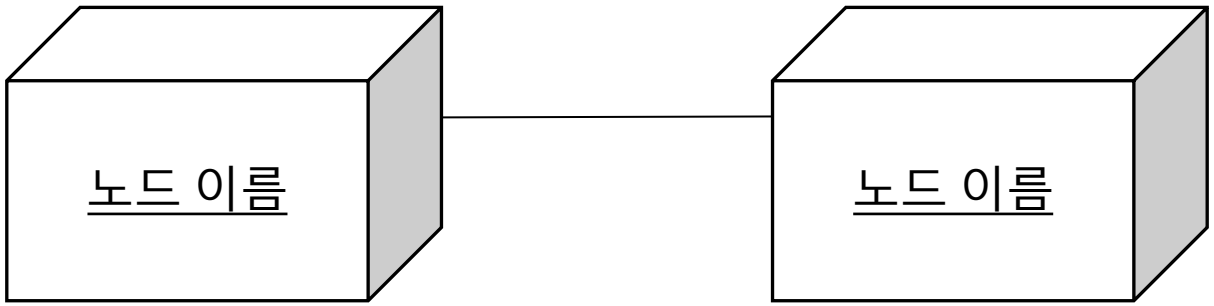
Graphical Notations

- **UML 표기법**



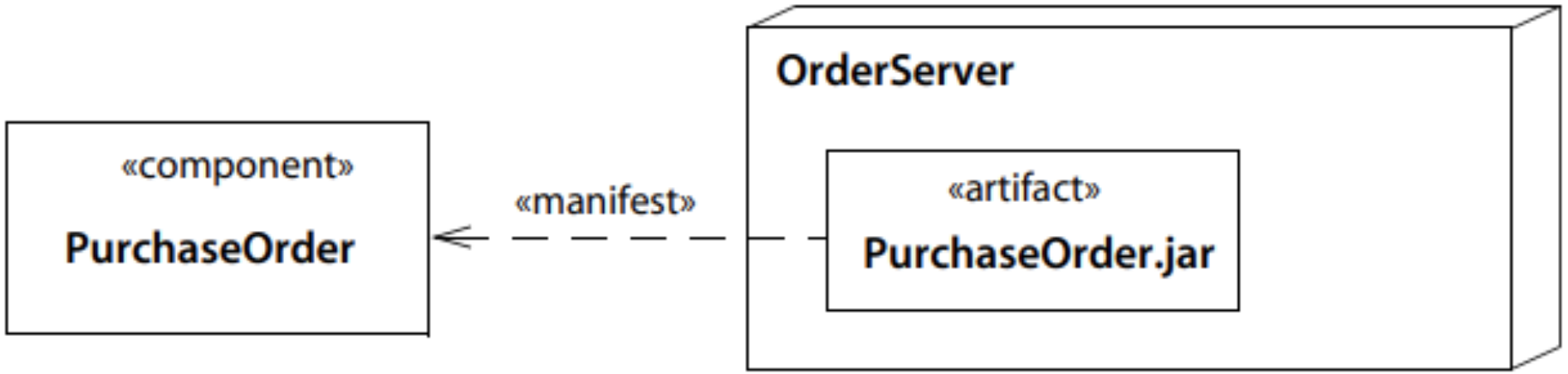
Graphical Notations

- 노드
  - 실시간 계산하는 자원을 모델링함
    - 메모리와 계산 능력을 가지고 있음,
    - **CPU**, 기기, 메모리 등을 구분하는 스테레오 타입을 가질 수 있음
    - 산출물을 내포할 수 있음
- 관계: 통신 경로를 표현

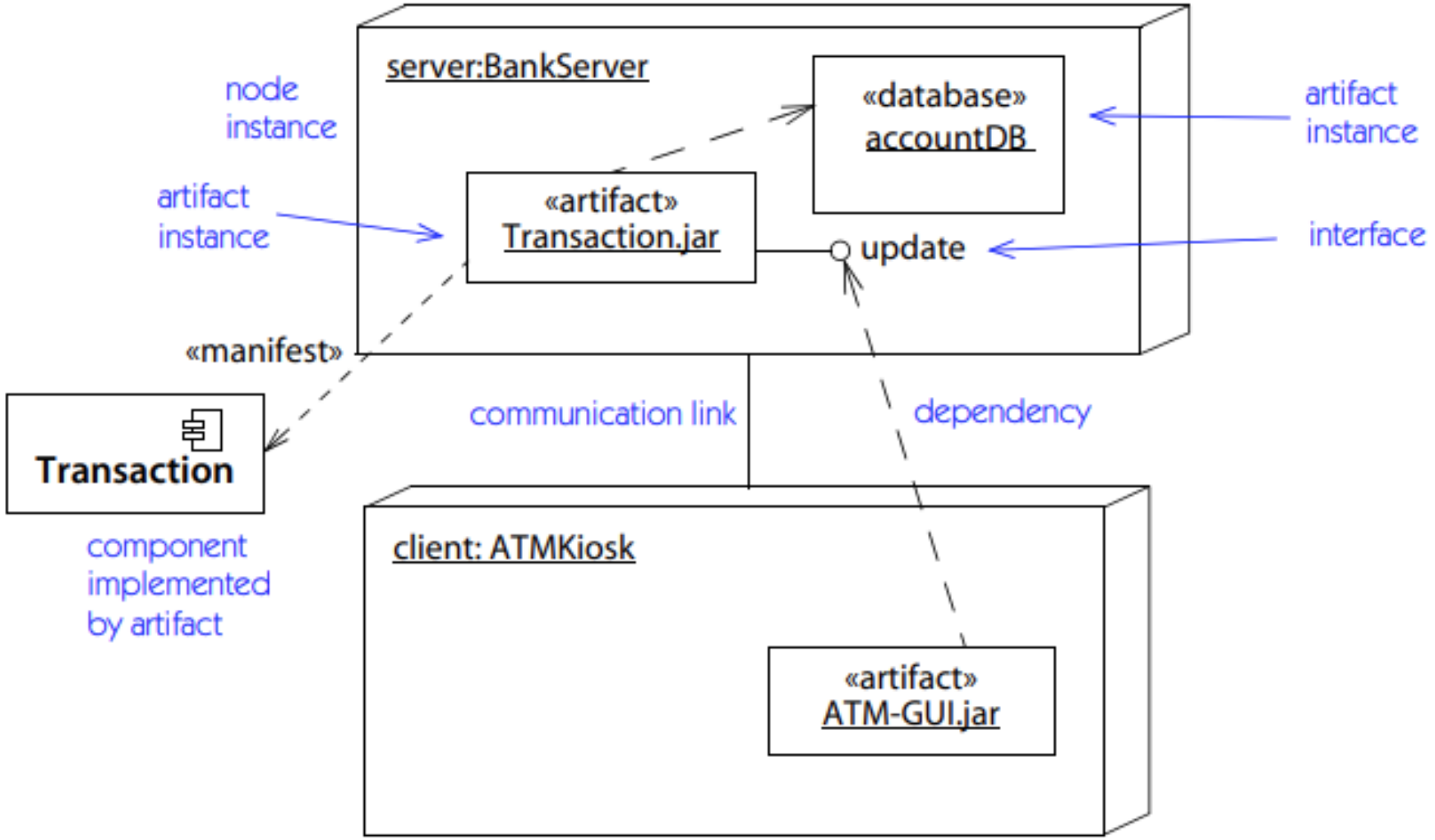


Graphical Notations

- 산출물 (Artifact)
  - 파일과 같은 물리적 개체를 모델링
  - 사각형 모양과 **<<artifact>>** 키워드와 함께 사각형으로 표현
  - 노드 심볼 내에 산출물 심볼을 배치하여 진행
    - 데이터베이스, 웹 페이지, 실행 가능한 개체, 스크립트 등이 될 수 있음



Graphical Notations



Usages

- 성능(performance) 분석
  - 하드웨어에 대한 소프트웨어의 할당 변경으로 성능 향상
    - 프로세서 상의 병목 현상 제거
    - 프로세서 이용도 향상을 위해 작업을 균등하게 할당
- 신뢰성(reliability) 분석
  - 신뢰성은 프로세서나 채널의 실패에 직면한 시스템의 행위에 의해 영향을 받음
    - 경고 없이 프로세서나 채널이 실패하는 경우
      - 배치 단위의 복사본을 개별 프로세서에 배치
  - 프로세서나 채널 실패 전에 경고를 받는 경우
    - 배치 단위는 실행 시간에 이동 가능

Usages

- 비용(**cost**) 예측
  - 비용은 시스템의 하드웨어 요소에 의존
  - 특정한 구성의 하드웨어 요소와 요소들의 목적을 표시하기 위해 배치 뷰 사용
- 전사적 배치(**Enterprise deployment**) 확보
  - 서브시스템 할당은 조직에 의존

Question?

Seonah Lee

saleese@gmail.com