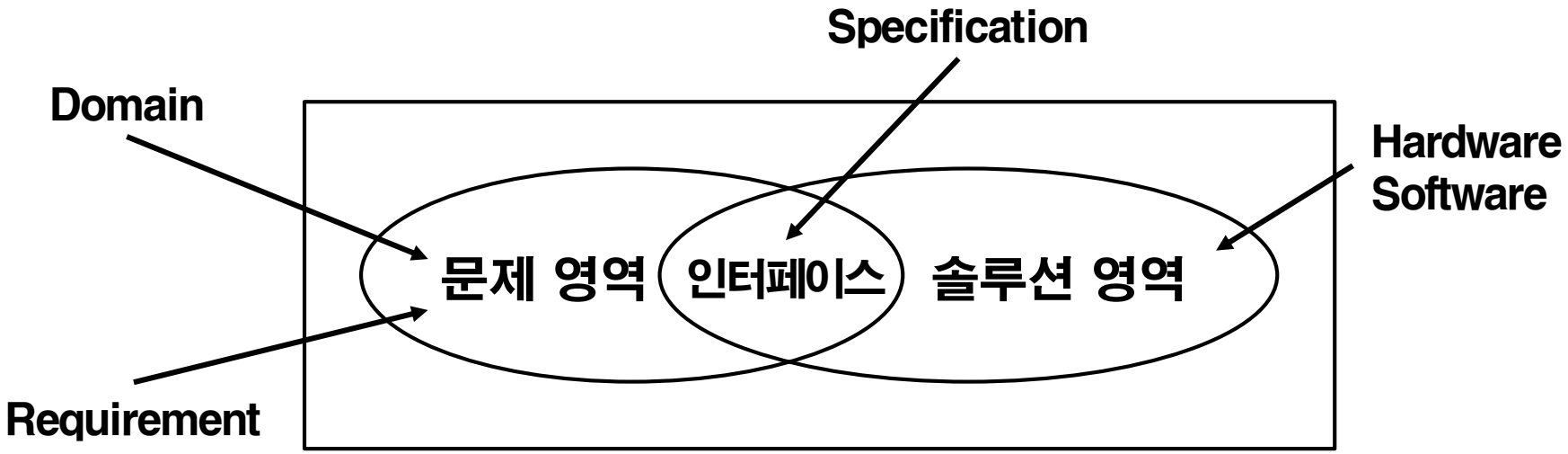
UML Modeling

- 모델링
- 시스템에 대한 다양한 관점
- UML의 모델링 뷰
- UML의 시스템 구조 모델링
- 다이어그램 작성시 유의사항

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

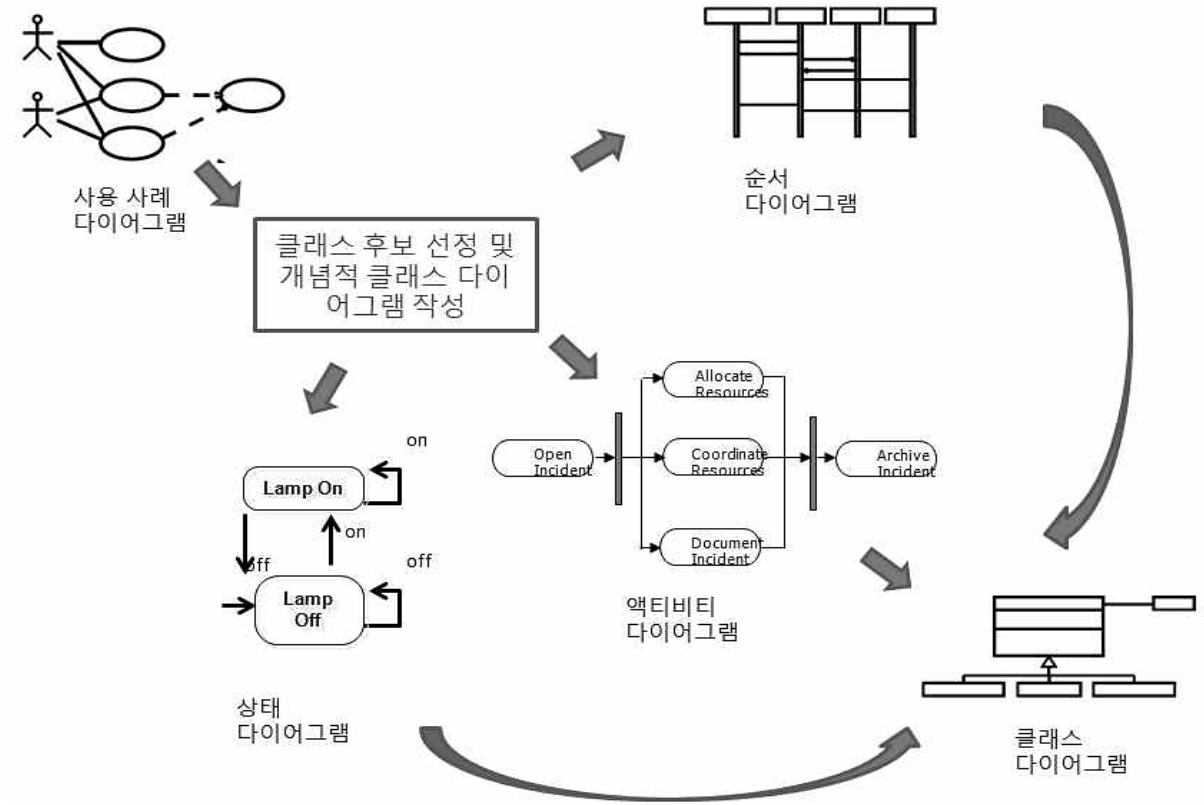
모델링

- 모델링(Modeling)이란?
  - 개발 대상 시스템을 잘 이해하기 위해서 현실 세계의 복잡한 모습을 단순화시키는 작업
  - 모델: 개발 대상 시스템을 단순하게 표현한 문서, 그림, 축소 모형 등



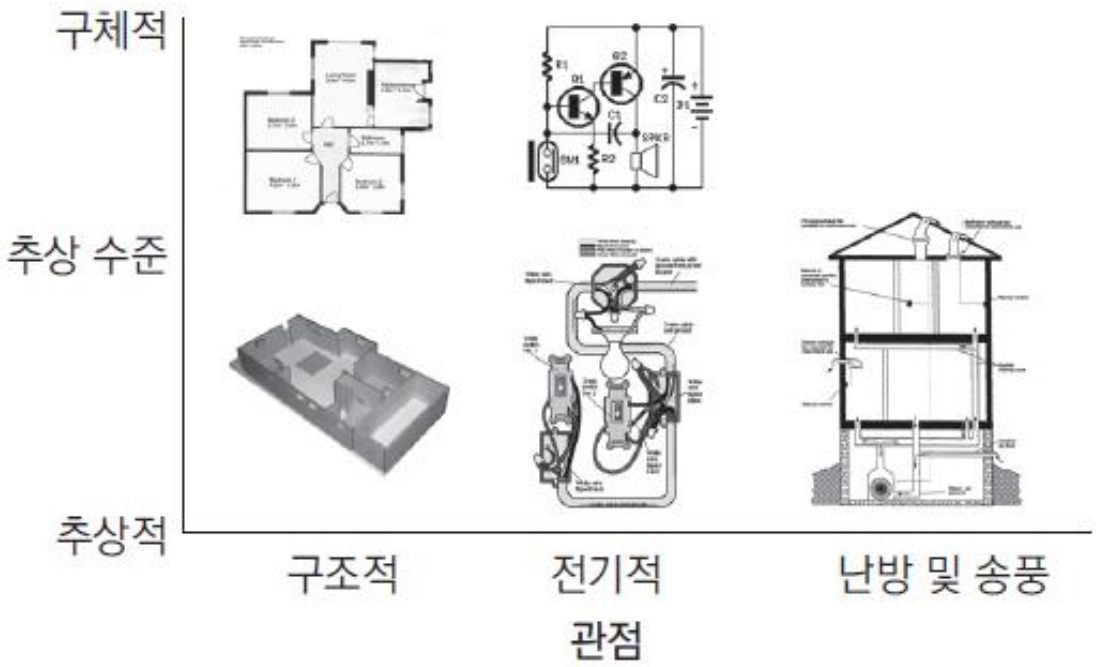
모델링

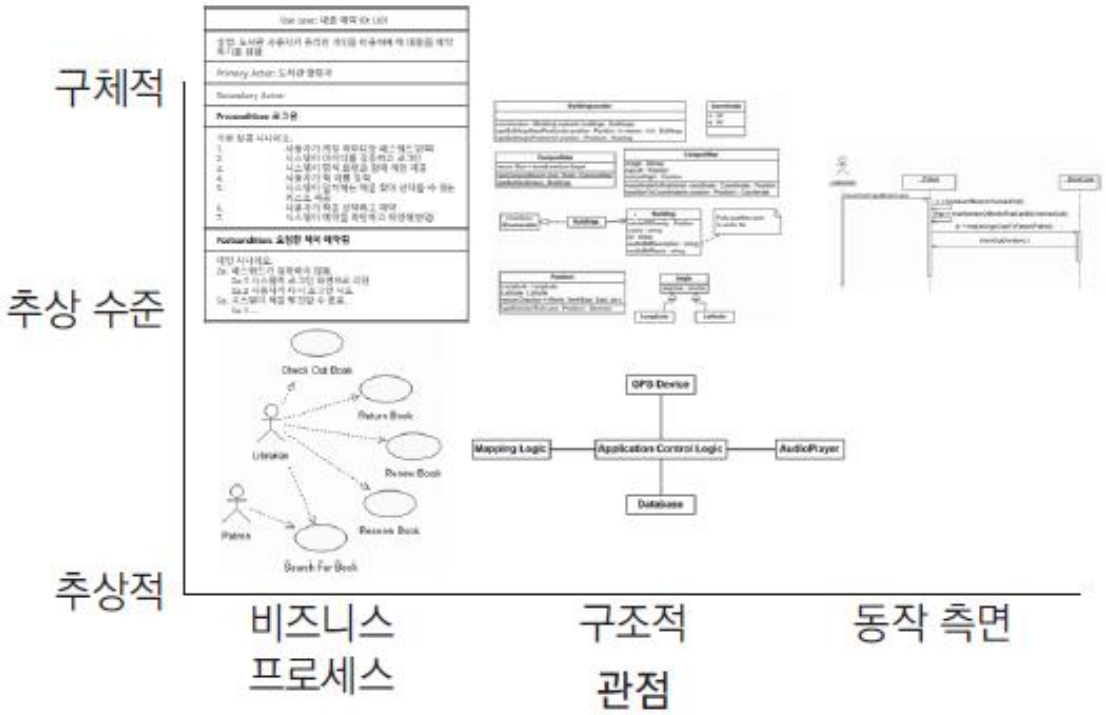
- UML 다이어그램을 사용한 모델링
  - 개발대상 시스템을 모델링하는 도구로써 다이어그램을 이용
  - Projection 기법을 기반으로 복잡한 문제를 단순화
    -> 다이어그램 = 특정 관점에서 본 시스템의 모습
  - 시스템의 전체 모습: 여러 개의 다이어그램으로 표현



시스템에 대한 다양한 관점(1)

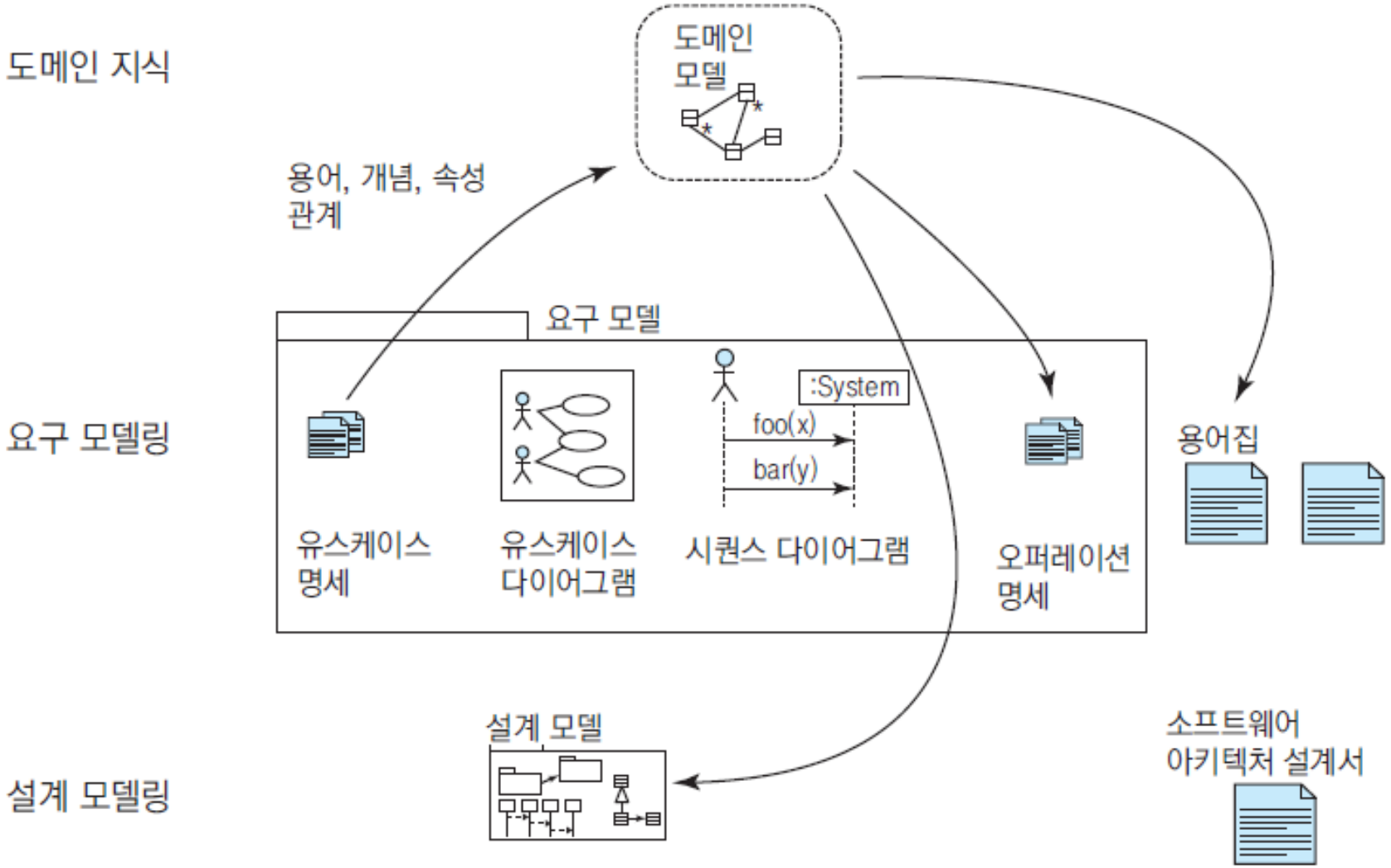
- 시스템과 관련된 이해 당사자
  - end user, technical writers, project manager, analyst, developers, system integrators, testers
  - 서로 다른 시기에 서로 다른 관점에서 시스템을 바라봄
  - 시스템에 대한 모델은 다양한 관점(View)에서 작성 되어야 함





시스템에 대한 다양한 관점(2)

❑ 모델 사이의 관계

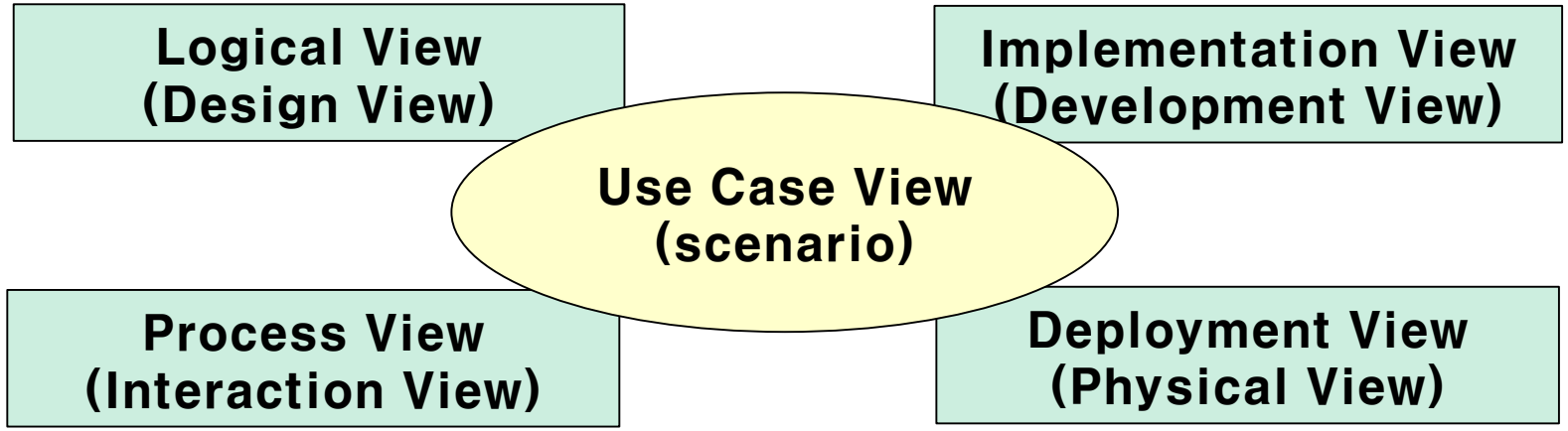


시스템에 대한 다양한 관점(3)

- 시스템의 소프트웨어 구조(Software Architecture)
  - 정의: 소프트웨어 컴포넌트들과 그들의 외부적으로 보여지는 특성,
    그리고 그들 상호간의 관계들로 구성되는 해당 시스템의 구조 또는 구조들
  - 다양한 관점을 관리, 통제함에 있어 핵심적 역할 담당
  - 소프트웨어 아키텍쳐 패턴
    - 계층화 패턴 (Layered pattern)
    - 클라이언트-서버 패턴 (Client-server pattern)
    - 마스터-슬레이브 패턴 (Master-slave pattern)
    - 파이프-필터 패턴 (Pipe-filter pattern)
    - 브로커 패턴 (Broker pattern)
    - 피어 투 피어 패턴 (Peer-to-peer pattern)
    - 이벤트-버스 패턴 (Event-bus pattern)
    - MVC 패턴 (Model-view-controller pattern)
    - 블랙보드 패턴 (Blackboard- pattern)
    - 인터프리터 패턴 (Interpreter pattern)

UML의 모델링 뷰(1)

❑ 4+1 View : UML을 이용한 소프트웨어 시스템 아키텍처



뷰	주요 관심 사항
Use case view	시스템의 기능적 요구
Logical view	시스템의 성능, scalability, 효율
Implementation view	시스템 구성 요소의 선택, 각각의 역할 및 기능
Process view	구성 요소의 조립 방법 및 형상 관리
Deployment view	구성 요소의 배치 방법, 납품 및 설치

UML의 모델링 뷰(2)

- Use case View
  - 사용자, 분석가, 검사자(tester)에게 보여지는 유즈케이스를 표현한 뷰
    - 정적인 측면: use case 다이어그램
    - 동적인 측면: interaction, state, activity 다이어그램

- Design View
  - 기능적 요구를 충족시키는데 필요한 일을 수행하는 시스템의 구성요소를 표현
    - 정적인 측면: class, object 다이어그램
    - 동적인 측면: interaction, state, activity 다이어그램
  - 표현대상 구조물
    - 클래스 및 그의 인터페이스, Collaboration, …

UML의 모델링 뷰(3)

❑ Interaction View

- 시스템 구성요소 사이의 제어흐름을 표현
  - 정적인 측면: class, object 다이어그램
  - 동적인 측면: interaction, state, activity 다이어그램

  ※ 디자인 뷰와 같은 다이어그램을 사용하나,
  액티브(active) 클래스와 메시지에 초점을 두는 것이 차이점

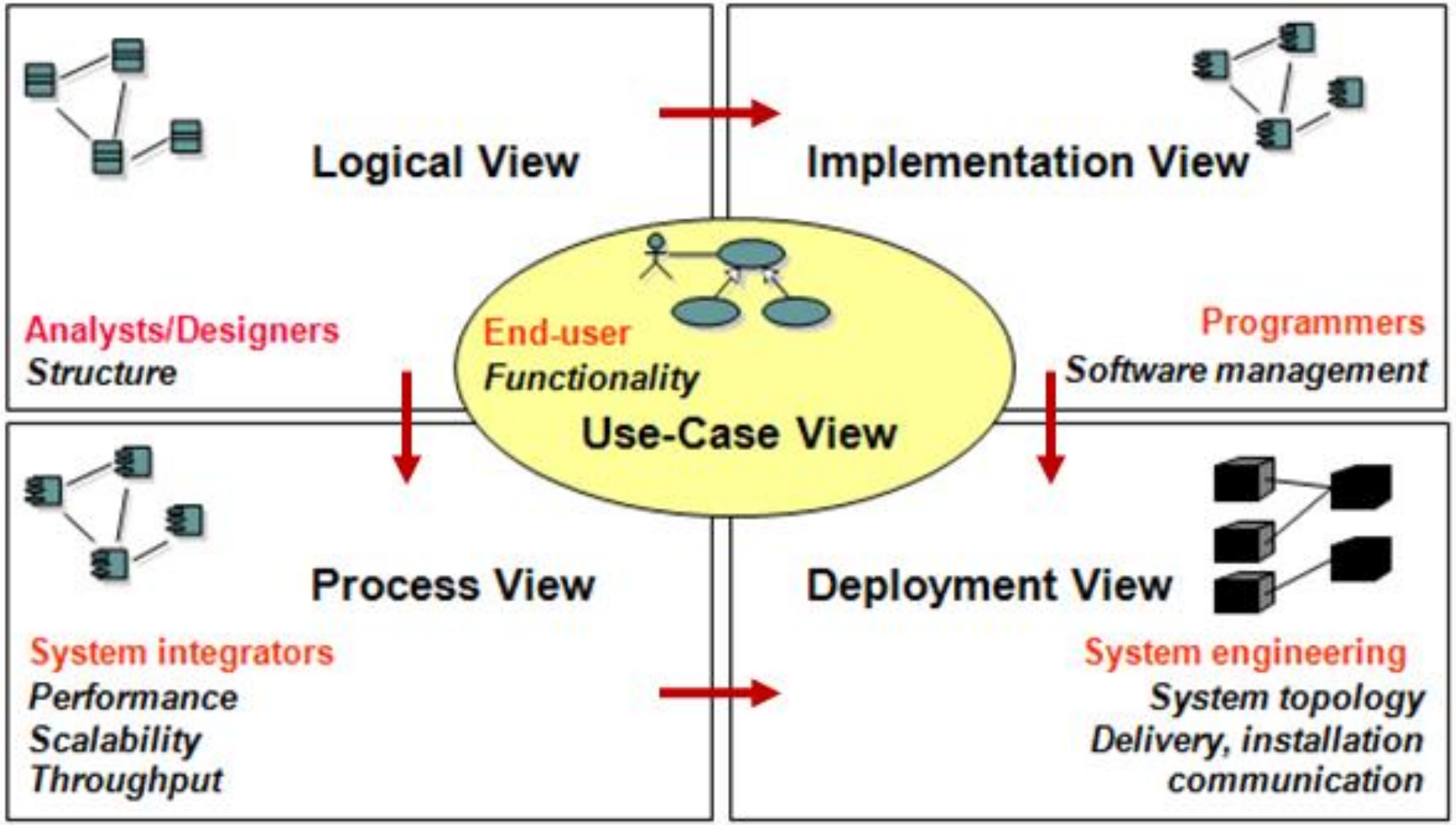
- 표현대상 구조물
  - 작업의 처리 순서
  - 병렬 처리를 수행할 쓰레드(thread)
  - 동기화에 참여하는 프로세스(process)

UML의 모델링 뷰(4)

- Implementation View
  - 구현된 시스템을 구성하는 물리적 산출물을 표현
    - 정적인 측면: artifact 다이어그램
    - 동적인 측면: interaction, state, activity 다이어그램
  - 표현대상 구조물
    - 설계 클래스 및 컴포넌트를 구현한 Artifact(산출물)

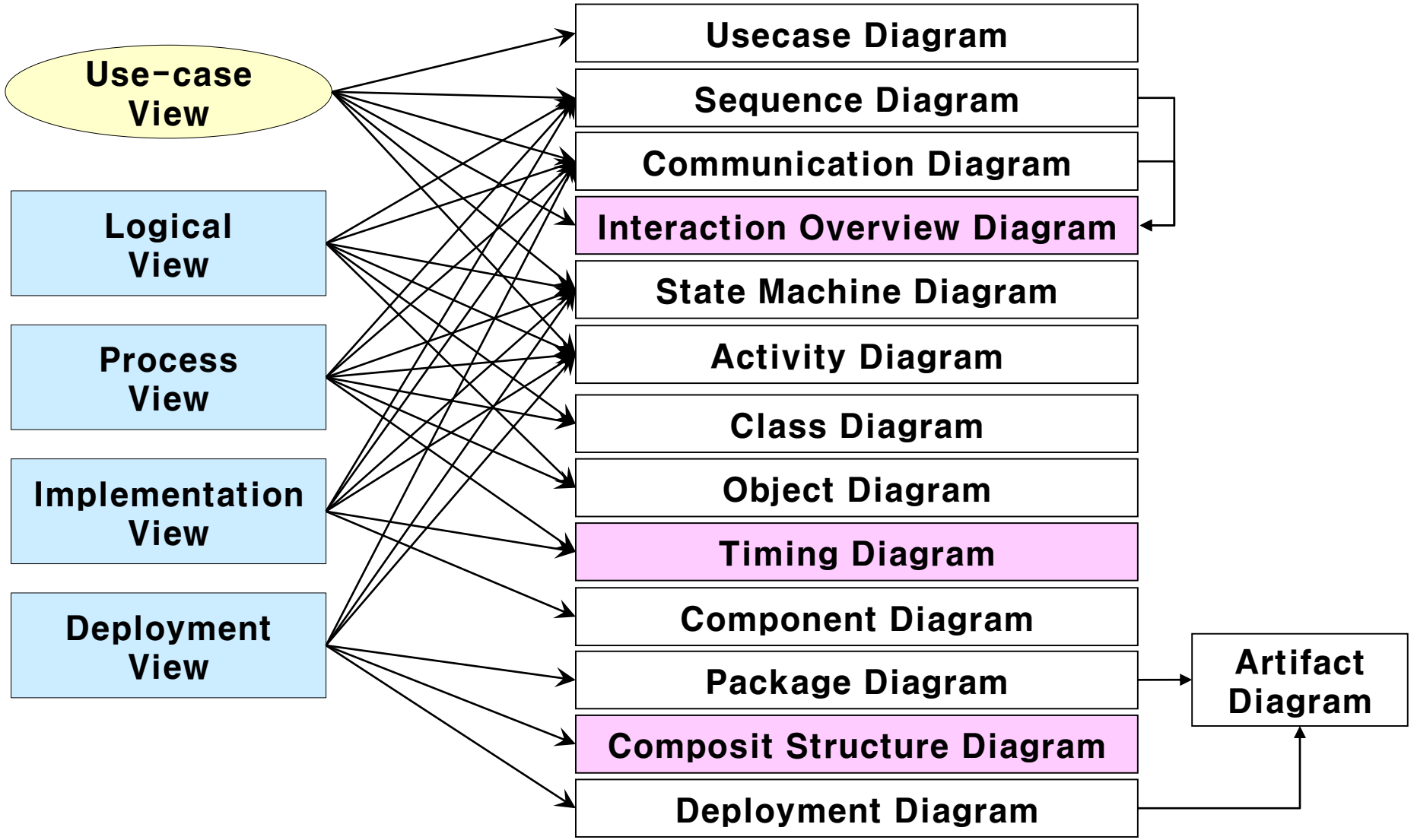
- Deployment View
  - 하드웨어 시스템을 구성하는 노드와 이에 탑재된 산출물을 표현
    - 정적인 측면: deployment 다이어그램
    - 동적인 측면: interaction, state, activity 다이어그램

UML의 모델링 뷰(5)


Logical View
Implementation View
Analysts/Designers
Structure
End-user
Functionality
Use-Case View
Programmers
Software management
Process View
Deployment View
System integrators
Performance
Scalability
Throughput
System engineering
System topology
Delivery, installation
communication

UML의 모델링 뷰(6)

❑ 4+1 View와 다이어그램 상관 관계



UML의 시스템 구조 모델링(1)

- 시스템의 구조는 서로 맞물린 5개의 뷰로 표현
  - 각각의 뷰는 특정한 관점(perspective)에서 시스템을 관찰하여, 해당 관점에서 중요한 사항을 포착
    - 대상 시스템에 따라 적용할 뷰의 선택은 매우 중요
  - 각각의 뷰에 대해서 작성할 다이어그램을 선택
    - 시스템의 정적 측면: structural 다이어그램으로 표현
    - 시스템의 동적 측면: behavioral 다이어그램으로 표현
    - 검토회의를 거칠 것과 보존할 것을 사전에 결정
  - 이용자에 따라 다이어그램의 상세도를 조절
    - 고객, 분석가: 세부 내역 생략
    - 개발자: 세부 내역 포함

UML의 시스템 구조 모델링(2)

- 개발대상 시스템에 따른 다이어그램 선택 예시
  - 단순 애플리케이션: Use case, Class, Interaction
  - Reactive 시스템: Use case, Class, Interaction, State, Activity
  - 클라이언트/서버 시스템: Use case, Class, Interaction, Component, Deployment
  - 복잡한 분산 시스템: 모든 다이어그램을 사용

다이어그램 작성시 유의사항(1)

- 다이어그램에 대한 기본 인식이 중요
  - 다이어그램은 Things와 이들 사이의 관계를 표현
  - 좋은 다이어그램: 대상 시스템을 쉽게 이해할 수 있도록 작성된 것
  - 대상 시스템에 따라 적절한 다이어그램을 선택하는 것은 매우 중요한 의사결정
  - 다이어그램은 점진적이고, 반복적으로 작성

- 체계가 잘 갖춰진 다이어그램을 작성하려면?
  - 다이어그램의 용도에 맞도록 관점을 설정
  - 설정한 관점에서 바라 본 시스템의 모습을 표현
  - 다이어그램의 상세도는 필요한 수준으로 유지
  - 과도한 단순화로 필요한 정보가 누락되지 않도록!

다이어그램 작성시 유의사항(2)

- 다이어그램 작성시 준수 사항
  - 다이어그램은 대상 시스템을 가시적으로 기술하고, 구축하며, 문서화하는 것임을 명심할 것!
  - 개발 및 유지보수에 필요한 다이어그램만 보존
  - 꼭 필요한 다이어그램 외에는 작성하지 말 것!
  - 세부 내역은 개발 단계별로 필요한 만큼만 포함
  - 필수적인 내용을 생략하는 것은 절대 불가!!
  - Structural/Behavioral 다이어그램 사이에 균형 유지
  - 다이어그램이 너무 크거나 작아지지 않도록 유의
  - 다이어그램에 의미 있는 이름을 부여할 것!
  - 다이어그램을 패키지 단위로 분류
  - 다이어그램의 포맷에 너무 구애 받지 않도록!

다이어그램 작성시 유의사항(2)

- 복잡한 다이어그램의 작성
  - 시스템의 분할을 잘하더라도 크고 복잡한 다이어그램은 항상 존재
    ex> 100개 이상의 DB 테이블을 포함하는 시스템
  - 복잡한 다이어그램 작성시 유의사항
    - 상대적으로 중요하지 않은 상세 내역은 생략
    - 구성 요소를 적절하게 분류하여 패키지로 그룹화
    - 분류된 그룹별로 출력하여 넓은 벽면에 부착하여 편하게 참조할 수 있도록 할 것
    - 노트와 색상을 적절하게 사용하여 식별하기 쉽도록 할 것