Object-oriented Analysis & Design

프로젝트 정의 및 소프트웨어 개발 프로세스

bangry313@gmail.com

프로젝트 개념

- 정의
  - 유일한 상품 혹은 서비스를 창출하기 이하여 수행되는 한시적인 활용
- 특징
  - 시작과 끝이 존재(Temporary)
  - 전에 수행한 적이 없는 유일한 행위(Unique)
  - 순차적으로 연속되는 작업의 완성(Progressive Elaboration)
- 프로젝트 예
  - 신제품 및 서비스 개발
  - 조직의 구조, 구성원 및 스타일을 효과적으로 변화 시키는 행위
  - 운송수단의 설계
  - 정보시스템을 개발하거나 도입
  - 빌딩 혹은 공장의 건설 등

프로젝트 추진 프로세스

단계	발주사	제안사	고려사항
프로젝트 준비	1. 프로젝트 기안 및 결재		
	2. **RFI(Request For Information)** 작성 및 발송		• 프로젝트 수행을 위한 사전정보(업체, 기술 등) 획득 및 목적 • 10개 업체 내외
		3.(RFI) Proposal	
	4. **RFP(Request For Proposal)** 작성 및 대상업체 선정 및 발송(공공기관은 입찰공고로 대체)		• 프로젝트 추진을 위한 업체선정 목적 • 3~5개 업체 내외 • RFP 작성 시 평가기준 수립
	5. RFP 설명회 개최		
		6. (RFP) Proposal	
		7. 제안 설명회	
	8. 제안 평가		• 기 작성한 평가기준에 의거 시행
	9.우선협상 대상자 선정		• 3개 내외 업체 선정
	10. 기술 및 가격 협상		
	11. 프로젝트 수행업체 선정 및 계약		
프로젝트 계획	12. 프로젝트 계획 수립		• 프로젝트 정의서 • 프로젝트 수행계획서 • 품질 관리 계획서 작성
	13. 프로젝트 추진 승인 획득		

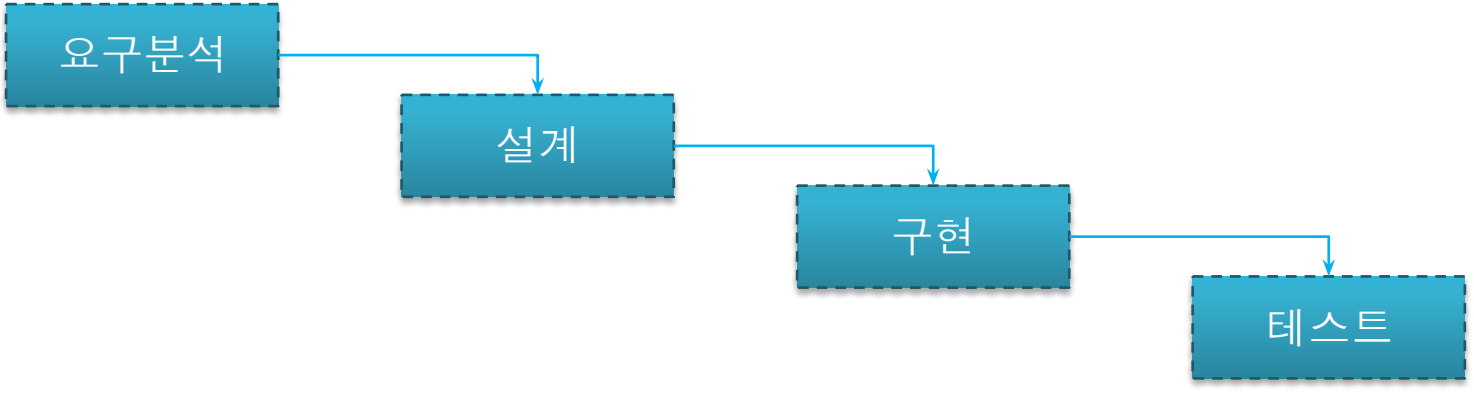
소프트웨어 개발 프로세스

- 요구사항을 충족시키는 소프트웨어 시스템을 구축하기 위하여 수행되는 일련의 활동

단계	설 명
요구분석	개발할 시스템에 대하여 요구되는 기능이 무엇인지, 어떤 제약 사항을 가지고 있는 지를 분석하고 문서화
설계	요구분석 결과를 토대로 시스템에 대한 개발 계획과 구체적인 모습을 고안
구현	설계를 토대로 시스템을 제작
테스트	제작된 시스템이 요구분석 상황에 적합하게 만들어 졌는지 분석

개발 라이프사이클에 따른 개발 모델

- 폭포수 개발 모델
  - 소프트웨어 개발 과정이 단계별로 진행되며, 거슬러 반복되는 경우가 없다
  - 각 단계가 끝난 후 결과를 명확히 정의해야 하며, 결과물의 구조가 표준화되어 있는 경우에 적합



  - 장점
    - 각 단계별로 정형화된 접근 방법과 체계적인 문서화를 할 수 있음
    - 응용분야가 단순하거나 잘 알고 있는 경우에 적합
    - 비전문가가 사용할 시스템을 개발하는 데 적합
  - 단점
    - 문서접근 방식의 문제점을 가짐
    - 개발 단계별 융통성이 없음
    - 실제 동작하는 소프트웨어를 개발 후반부에 확인할 수 있음

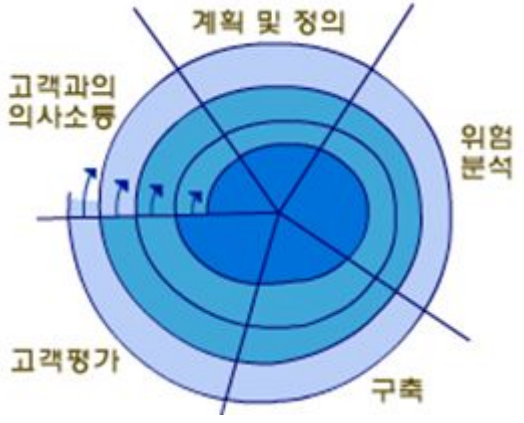
개발 라이프사이클에 따른 개발 모델

- 프로토타입(Prototype) 개발 모델
  - 실제 개발 이전에 사용자나 고객이 시제품을 보고 평가하여 요구 사항을 검증하는 개발 모델
  - 사용자와 시스템간의 인터페이스에 초점을 맞춰 개발되는데, 고객으로부터 피드백을 얻은 후에는 프로토타입을 버리는 경우가 많음
  - 원하는 시스템 기능 중 중요한 부분만 구현하여 피드백을 받은 후 계속 발전시켜 완제품을 만들어 낼 수 있음

  - 장점
    - 고객 요구사항이 모호하거나 요구사항을 명세화 할 수 없는 경우 사용하면 유용
    - 여러 테스트 환경을 설정하고 각 테스트 환경 별로 검증을 하여 이후 시스템 테스트에 들어가는 노력 최소화
    - 실제 제품이 만들어져 사용자에게 보급되기 전에 미리 사용자를 교육시키는 데 유용
  - 단점
    - 고객은 프로토타입을 최종 제품으로 오해하여 약간의 수정만 가하면 완제품이 될 것이라 기대할 수 있음
    - 프로토타입 개발 시 사용한 비효율적인 알고리즘, 인터페이스, 성능을 고려하지 않은 구현 등의 개발이 진행되면서 수정되어야 하는 내용이 누락될 수 있음

개발 라이프사이클에 따른 개발 모델

- 나선형 개발 모델
  - 개발 단계를 반복적으로 수행함으로 써 점증적으로 완벽한 소프트웨어를 개발하는 진화적 모델
  - 프로토타입 모델의 장점을 수용하고, 새로운 요소인 위험 분석을 추가한 모델
  - 시스템 개발 시 발생하는 위험을 최소화하여 관리하고자 하는 것이 주목적임

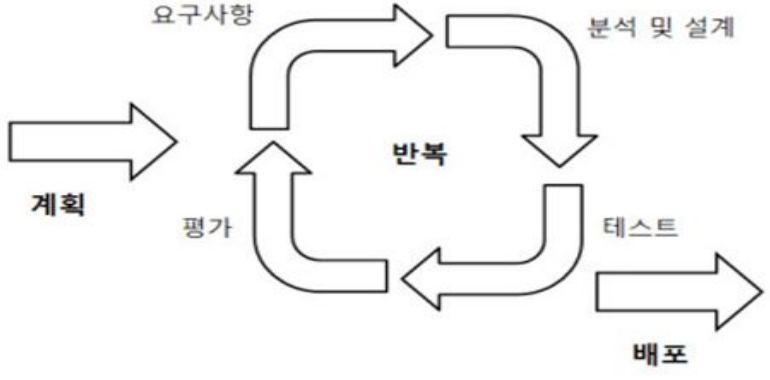


  - 장점
    - 시스템 개발에 많은 비용이 들거나 개발 기간이 큰 시스템을 구축할 때 효과적으로 적용할 수 있음
    - 각 순환주기를 나선형 모델의 특별한 경우로 간주하며, 근본적인 유지보수와 개발간에 큰 차이가 없음
    - 유지보수를 개발과정으로 봄
  - 단점
    - 폭포수 모델이나 프로토타입 모델보다 상대적으로 복잡하여 프로젝트 관리 자체를 어렵게 할 수 있음
    - 개발자가 발생할 수 있는 위험을 정확하게 분석하지 못하거나 파악하는 기술이 부족하면 엄청난 문제 야기할 수 있음

개발 프로세스에 따른 개발 모델

■ RUP(Rational Unified Process) 개발 모델

- Rational 회사에서 제안한 "UML 기반의 객체지향 개발 프로세스"로서 개발 조직 내에서 작업과 책임을 할당하기 위한 규칙을 제안
  - 예정된 일정과 예산의 범위 안에서 고객의 요구사항을 충족시키는 고품질의 소프트웨어 개발 가능
- 객체지향 프로젝트 수행에 맞도록 개발된 표준 소프트웨어 개발 모델로 CBD(Component Based Development) 등의 개발모델로 확장되어질 수 있다.
- 특징
  - 유스케이스 기반(Use Case Driven) 개발 모델
    - Use Case Model을 기반으로 개발자는 유스케이스를 실현함으로써 분석, 설계, 테스트로의 추적성 및 일관성을 유지할 수 있음
  - 아키텍처 중심적(Architecture Centric)
    - 아키텍처(Architecture)를 중심으로 복잡한 프로젝트를 운영하고, 시스템의 무결성을 유지하도록 프로젝트 전체에 대한 통제를 할 수 있음
  - 반복적 & 점진적(Iterative & Incremental)
    - 한 사이클이 끝날 때 마다 테스트가 완료되어 통합 및 수행 가능한 시스템이 산출되는 개발 모델
    - 여러 번의 릴리즈를 거치면서 사용자의 피드백이나 의견에 따라 원하는 시스템에 더욱 근접한 시스템을 만들 수 있다.

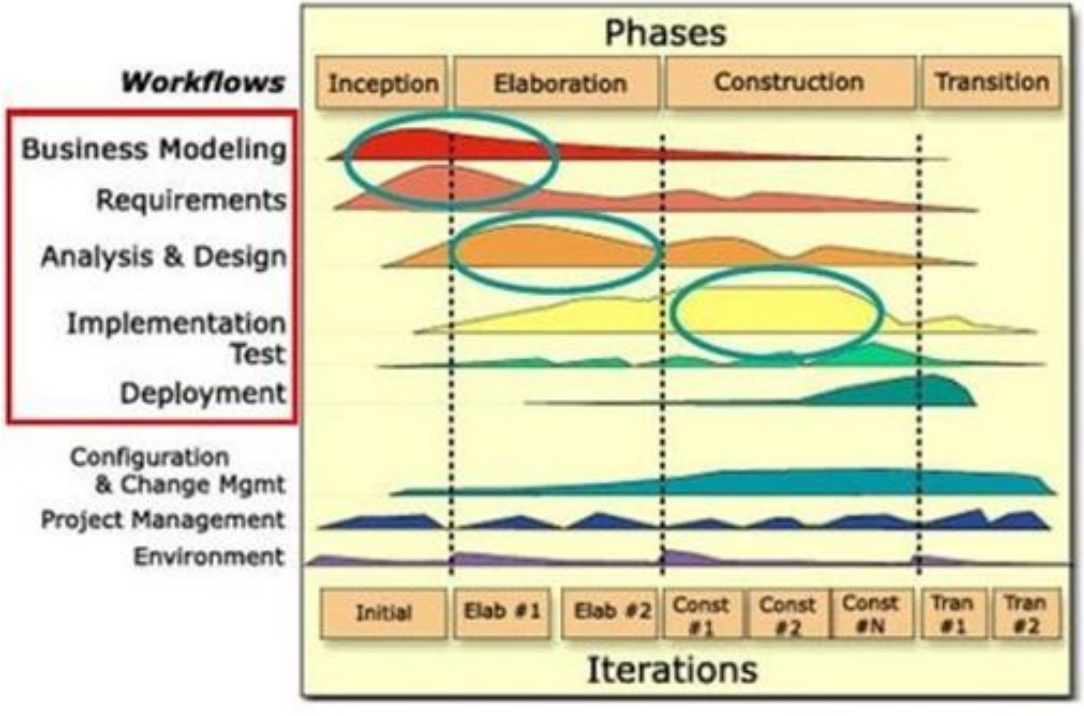


개발 프로세스에 따른 개발 모델

■ RUP(Rational Unified Process) 개발 모델

– 개발 공정

- 도입(Inception) 단계 : 개발의 시작, 대상 요소의 정의
- 정련(Elaboration) 단계 : 소프트웨어 아키텍처, 시스템 뼈대 확립
- 구축(Construction) 단계 : 소프트웨어 작성 및 실행
- 전이(Transition) 단계 : 테스트, 설치, 다음 반복단계 준비



1) 수평축 : 시간에 따른 변화, 동적인 생명 주기 측면을 나타내며, 시간의 흐름에 따라 단계(phase)로 나누고 단계별 이정표(milestone)를 제시
2) 수직축 : 핵심적인 작업흐름(workflow)을 표현하며, 활동(activity)에 대한 논리적인 그룹핑이라 할 수 있음. 정적인 측면에서 작업자(worker/누가), 활동(activity/어떻게), 작업흐름(workflow/언제), 산출물(artifact/무엇을)로 구성
   - 6개 핵심 엔지니어링 작업흐름 : 업무모델링, 요구사항, 분석/설계, 구현, 시험, 배치
   - 3개 지원 엔지니어링 작업흐름 : 프로젝트관리, 구성/변경관리, 환경

각 단계를 반복적으로 순환하면서 프로젝트의 위험요소(Risk Element)를 줄일 수 있음.

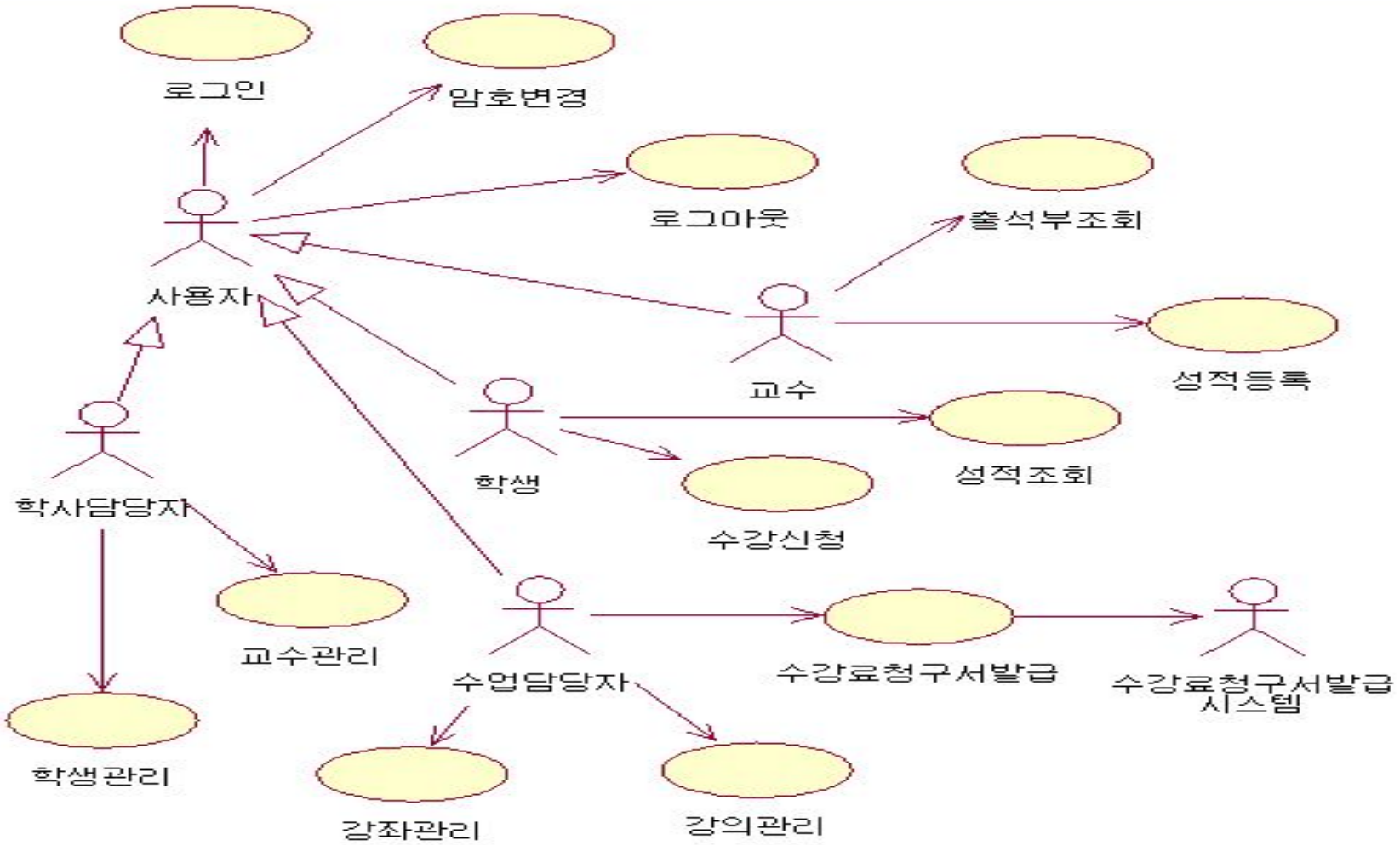
요구분석 활동

- 정의
  - 개발될 시스템에 대한 요구사항을 도출하고, 정의하여, 명세하는 활동이다.
- 특징
  - 요구분석 단계에서는 앞으로 구축될 시스템을 Black Box로 본다.
  - 구축될 시스템 내부에 어떤 것들이 있는지는 관심 밖이며 다만, 사용자 관점에서 시스템이 어떤 기능을 제공해야 하는 가에만 집중한다.
- 요구사항의 분류
  - 기능적 요구사항 : 시스템이 제공해야 하는 기능(동작/행위)에 대한 요구사항
  - 비기능적 요구사항 : 시스템의 동작/행위 자체 보다는 성능, 유지 보수성, 신뢰도와 같은 품질 요소에 대한 요구사항

- 요구분석 활동의 주요 산출물
  - 요구사항 정의서(Document)
  - 요구사항 명세서(Document)
  - 유스케이스 모델
    - Usecase Diagram(UML), 유스케이스 명세서(Document)
  - 이벤트흐름 모델 – Sequence Diagram(UML)
  - 화면 흐름 모델 – Activity Diagram(UML)
  - 화면 설계서(스토리보드)
  - 분석 객체 모델 – Class Diagram(UML)
  - 동적 모델(유스케이스 실현 모델) – Sequence Diagram(UML)

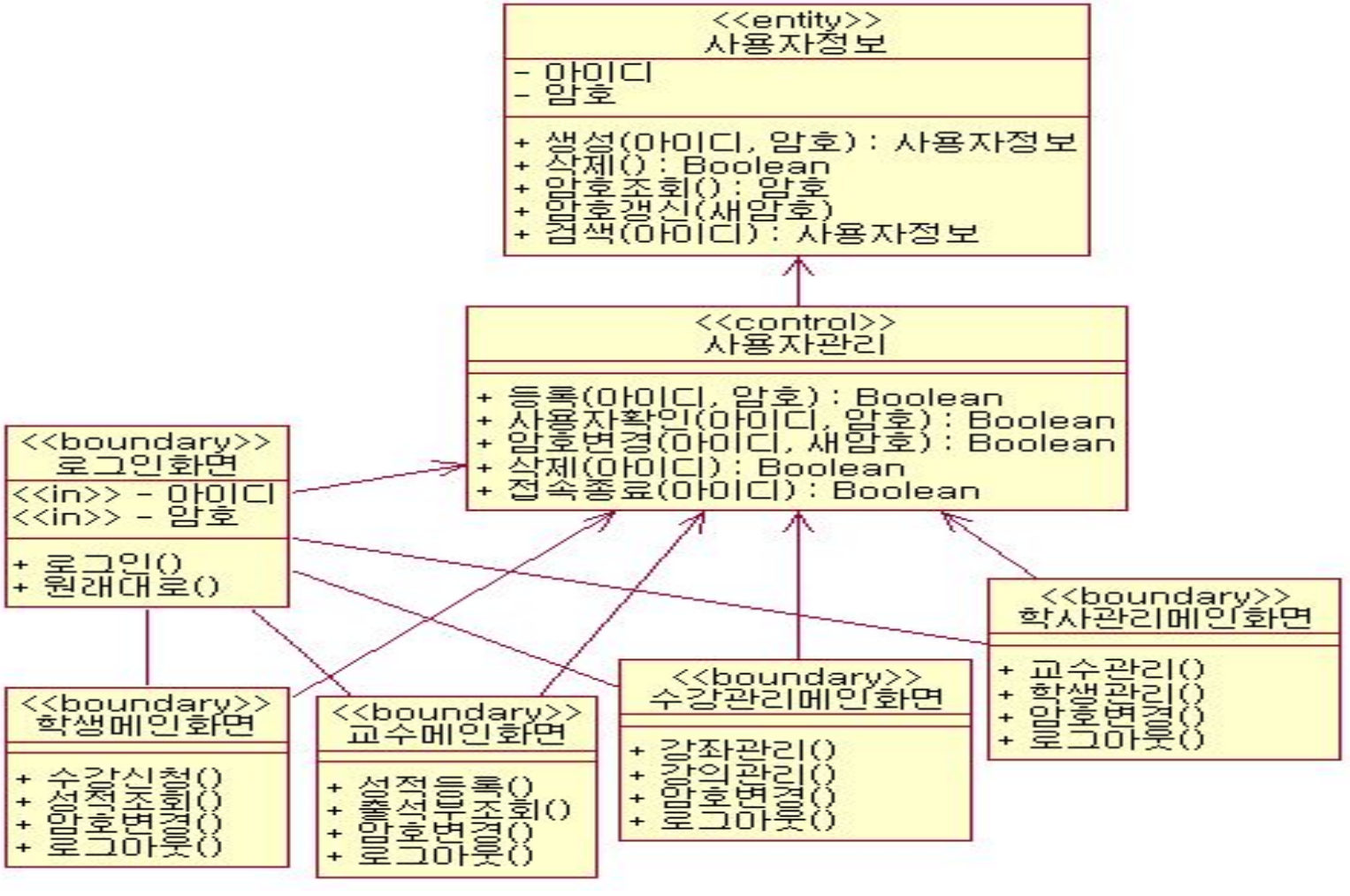
요구분석 활동의 UML 주요 다이어그램 예시

■ Usecase Diagram(유스케이스 모델)



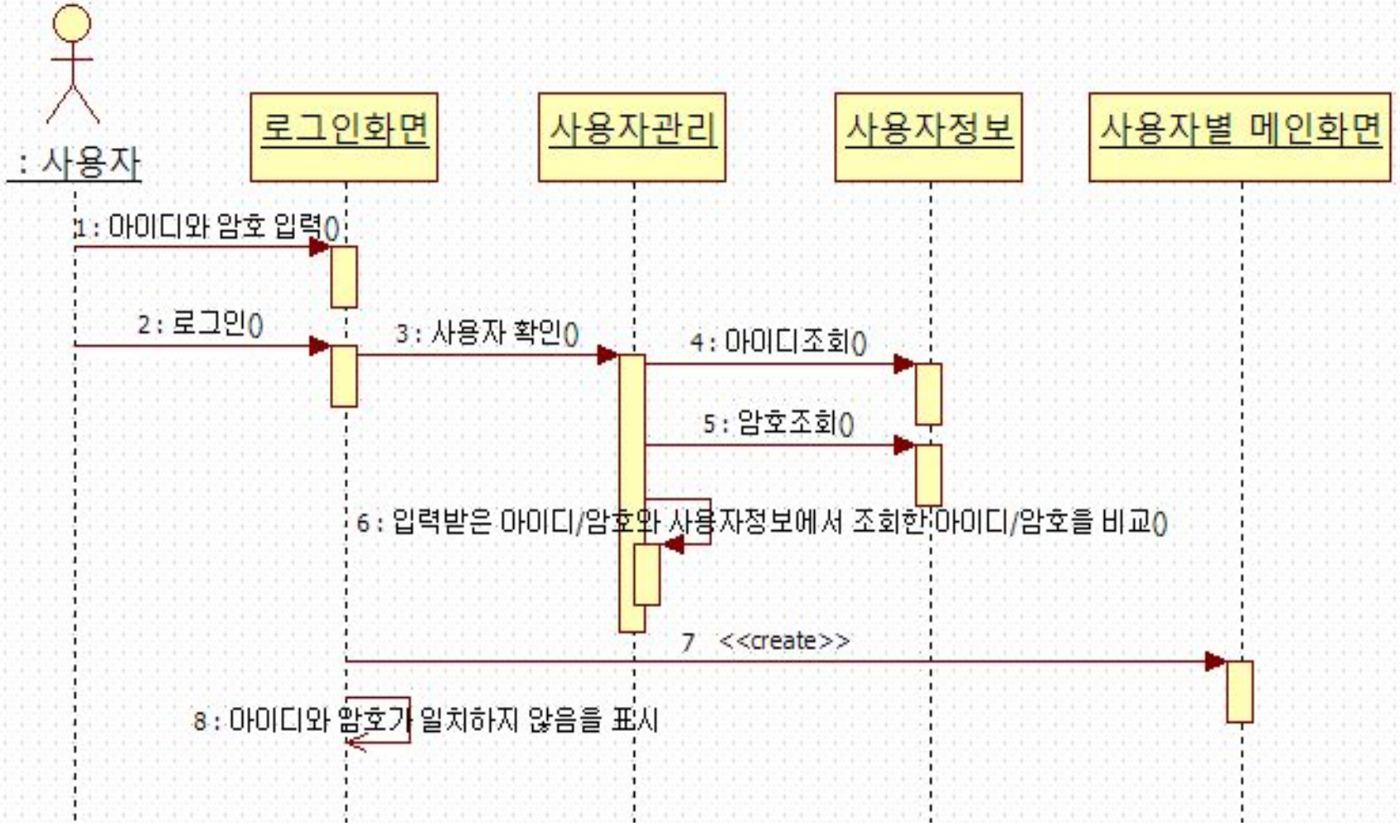
요구분석 활동의 UML 주요 다이어그램 예시

■ Class Diagram(분석객체 모델)



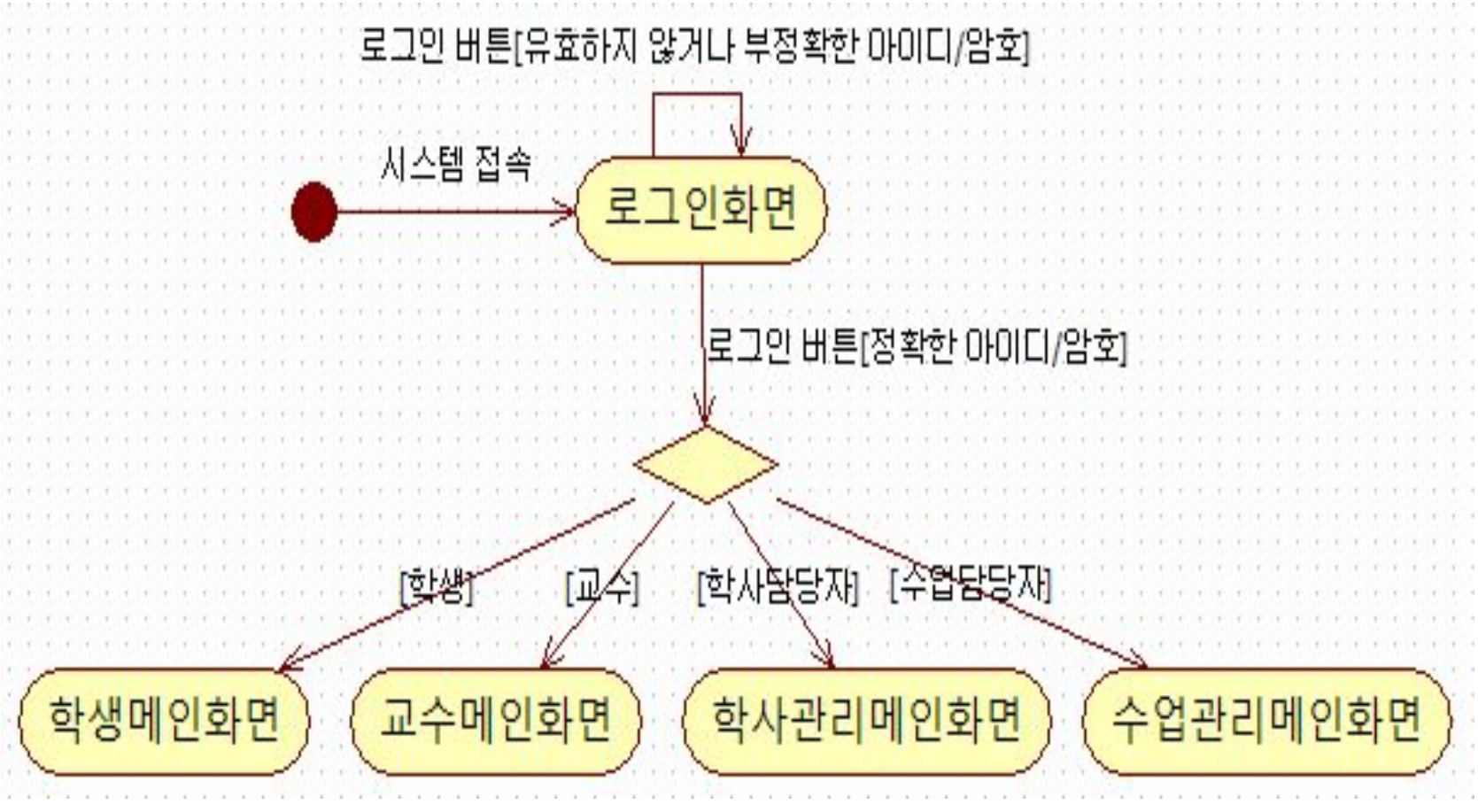
요구분석 활동의 UML 주요 다이어그램 예시

■ Sequence Diagram(동적모델)

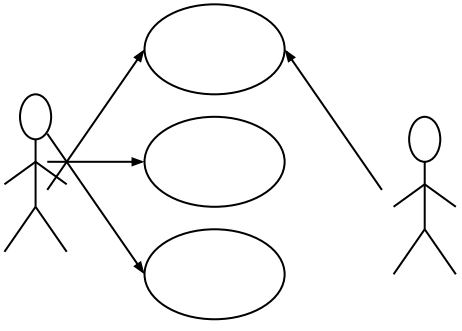


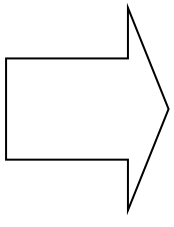
요구분석 활동의 UML 주요 다이어그램 예시

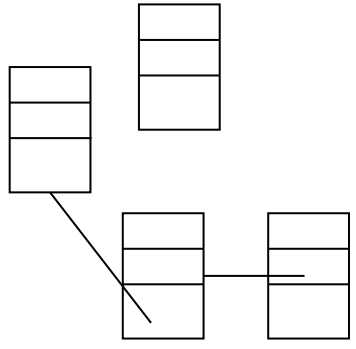
■ Activity Diagram(화면흐름 모델)



요구분석 활동의 UML 주요 다이어그램 예시



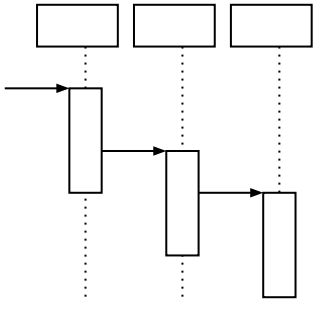




분석 객체 모델

분석 객체 도출

요구사항을 실현하기 위하여 필요한 객체는 ?



동적 모델

분석 객체 상세화

요구사항을 실현하기 위하여 파악된 객체들이 어떻게 상호작용을 해야 하는가 ?

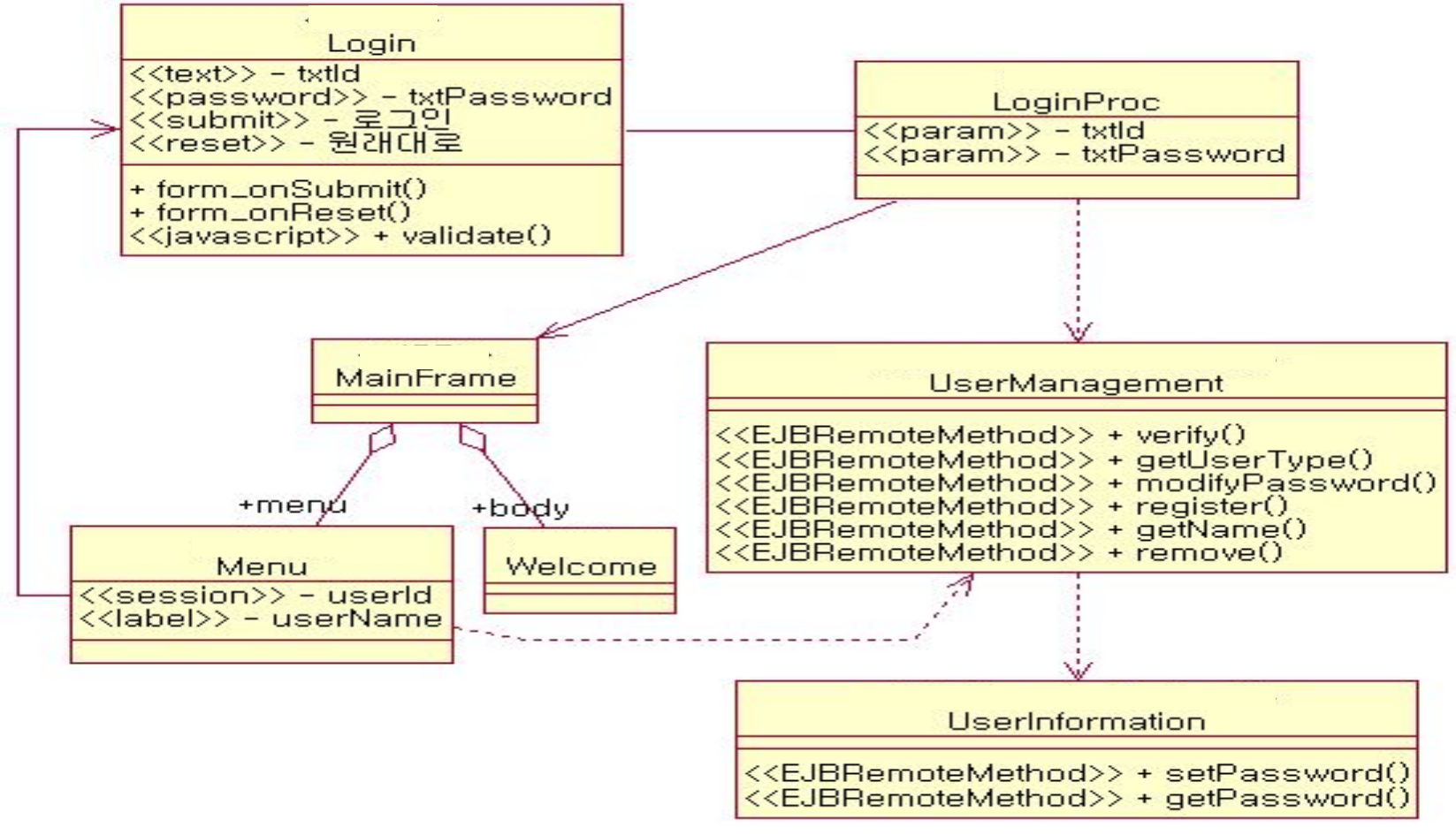
설계 활동

- 정의
  - 분석 결과를 바탕으로 개발될 시스템에 대한 구체적인 모습 정의하는 활동이다.
- 특징
  - 플랫폼 고려 : H/W, 운영체제, 미들웨어, DBMS, 개발 언어, 프레임워크 등
  - 비기능적인 요구사항 고려 : 성능, 확장성, 유지보수성, 보안, 신뢰도 등

- 설계 활동의 주요 산출물
  - 설계 객체 모델 – Class Diagram(UML)
  - 데이터 모델 – ERD
  - 설계 유스케이스 실현 모델 – Sequence Diagram(UML)
  - 컴포넌트 모델 – Component Diagram(UML)
  - 배치 모델 – Deployment Diagram(UML)

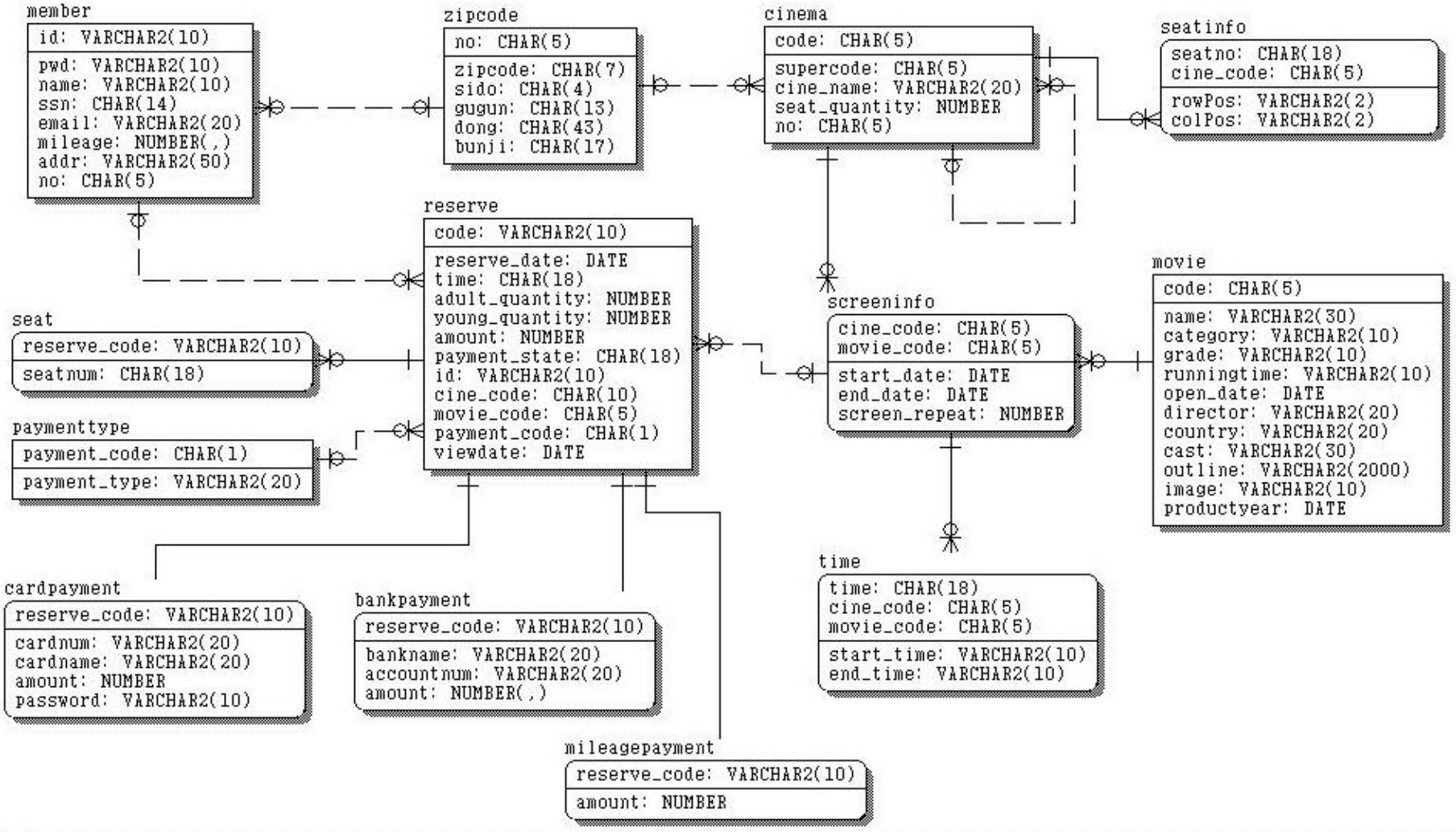
설계 활동의 UML 주요 다이어그램 예시

■ Class Diagram(설계 객체 모델)



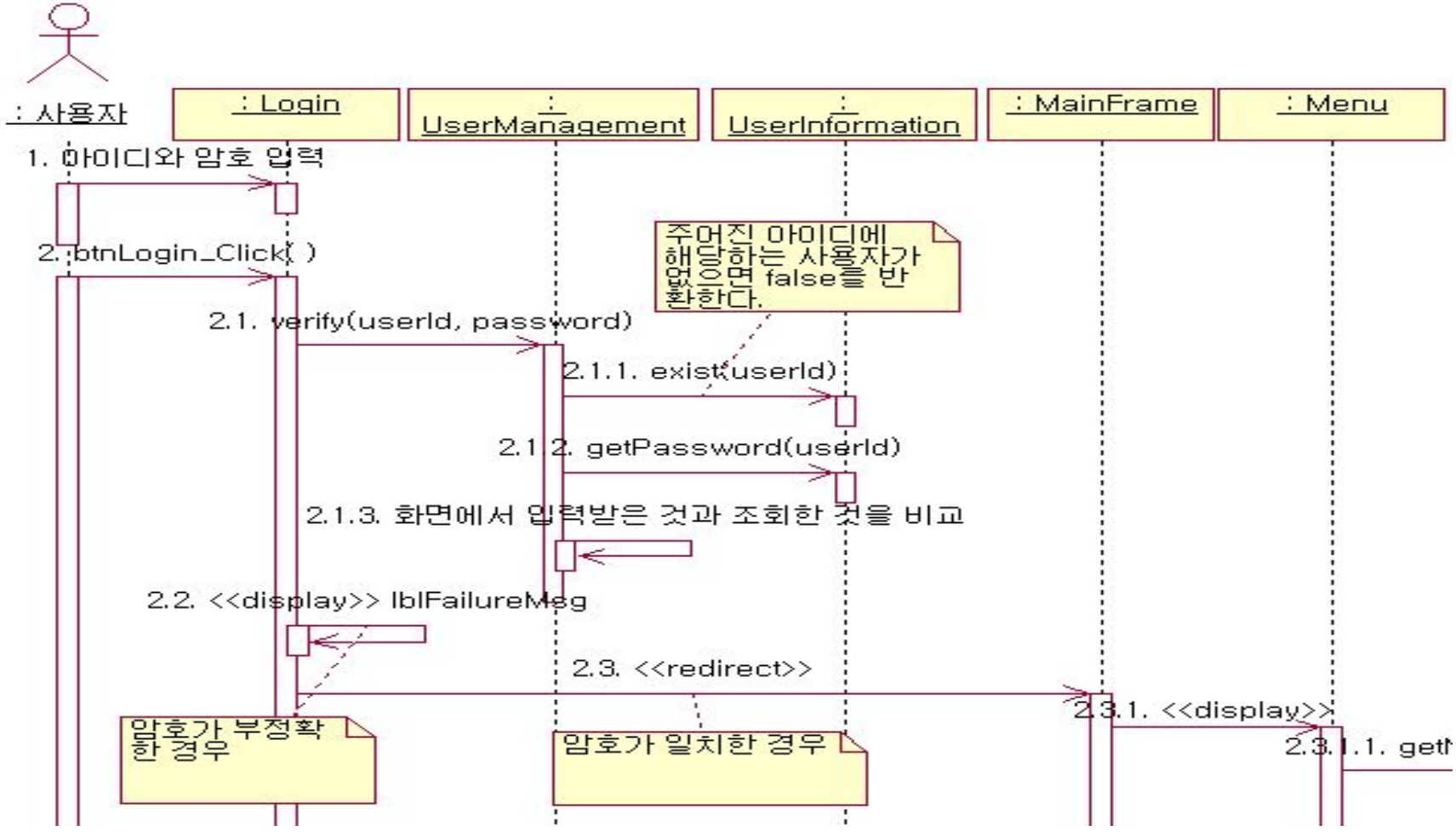
설계 활동의 UML 주요 다이어그램 예시

■ 데이터 모델(ERD)



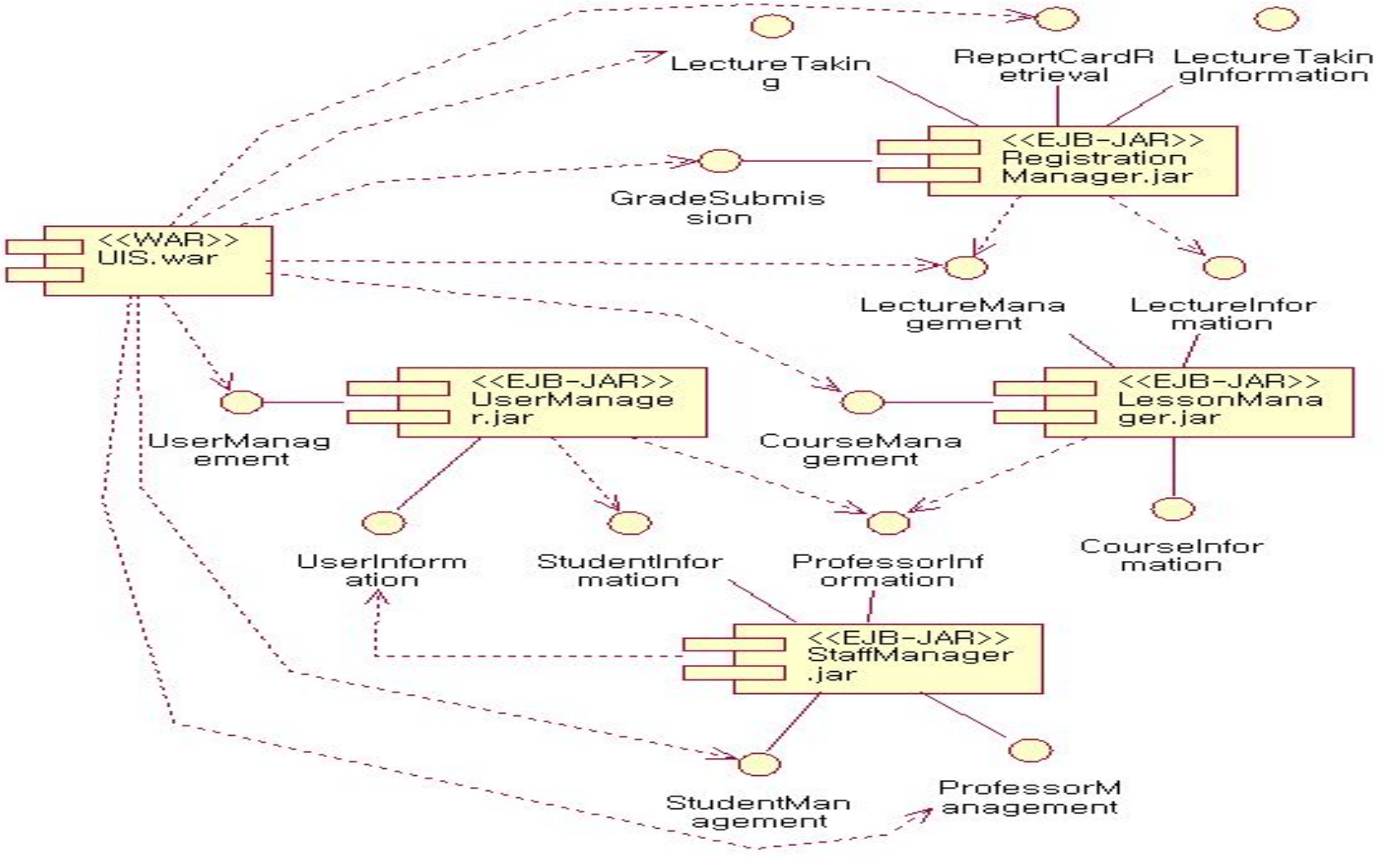
설계 활동의 UML 주요 다이어그램 예시

■ Sequence Diagram(설계 유스케이스 실현 모델)



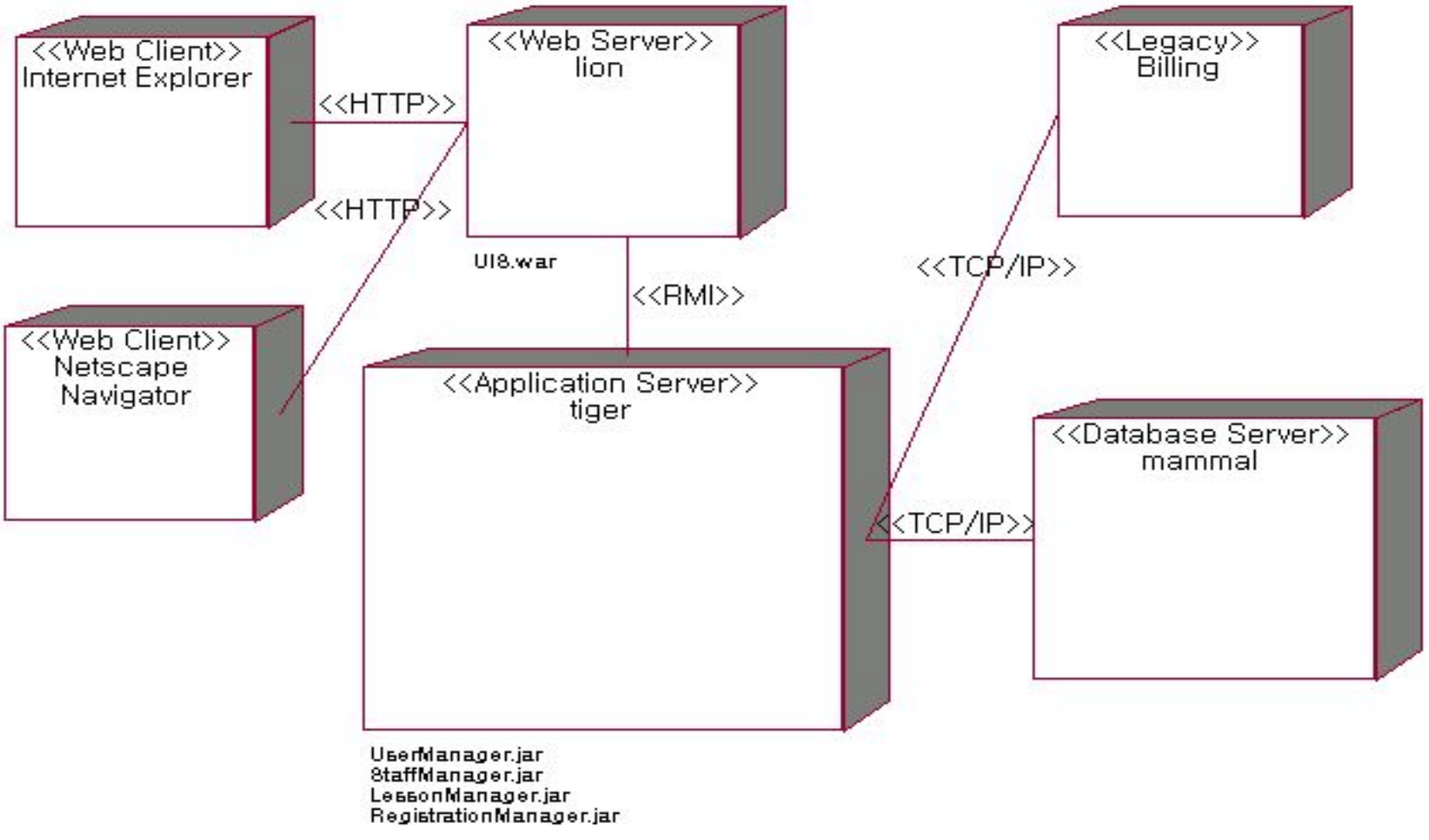
설계 활동의 UML 주요 다이어그램 예시

■ Component Diagram(컴포넌트 모델)



설계 활동의 UML 주요 다이어그램 예시

■ Deployment Diagram(배치 모델)



구현 활동

- 정의
  - 실제 프로그램 소스 코드를 작성하고, 컴파일하여 시스템을 구현하는 활동