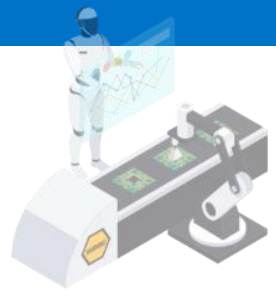
9강. 객체지향 분석과 설계

컴퓨터과학과 김희천 교수

목차



Chapter. 1

객체지향 분석과 설계 개요

1. 객체지향 분석과 설계 (1/2)

- 시스템을 상호작용하는 객체들로 모델링
  - **객체의 협력과 상호작용**을 표현하는 모델을 생성
  - 시스템 구성, 정적 구조, 동적 행위 등을 UML로 표현
- 객체지향 분석(OOA)과 설계(OOD) 과정은 일반적으로 반복적 점증적 프로세스

1. 객체지향 분석과 설계 (2/2)

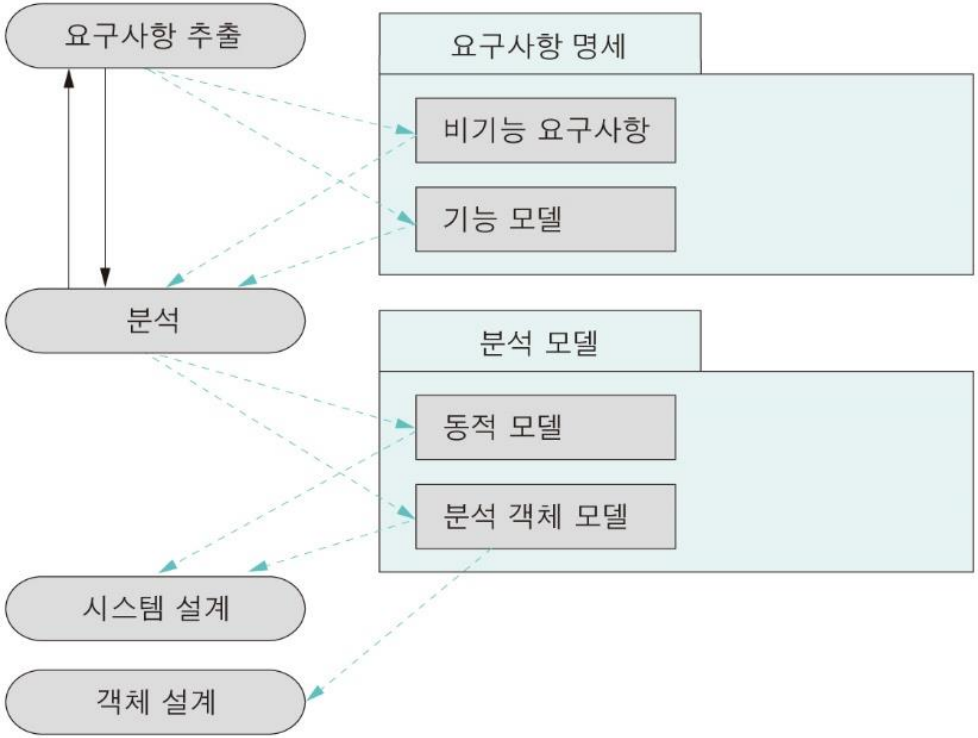
✚ 객체지향 분석

- 결과물은 시스템이 기능적으로 무엇을 수행하는지를 설명한 것
- 유스케이스 다이어그램, 유스케이스 명세
- 문제 도메인을 분석하여 개념 모델을 작성
- 클래스 다이어그램, 상호작용 다이어그램

✕ 객체지향 설계

- 결과물은 시스템을 어떻게 만들 것인가를 설명하는 것
- 분석 과정의 모델을 비기능적 요구사항과 아키텍처를 고려하여 변환
- 개념 클래스는 구현 클래스로 변환됨
- 응답 시간, 처리율, 실행 플랫폼, 개발 환경, 프로그래밍 언어를 고려

2. 객체지향 분석과 모델



Chapter. 2

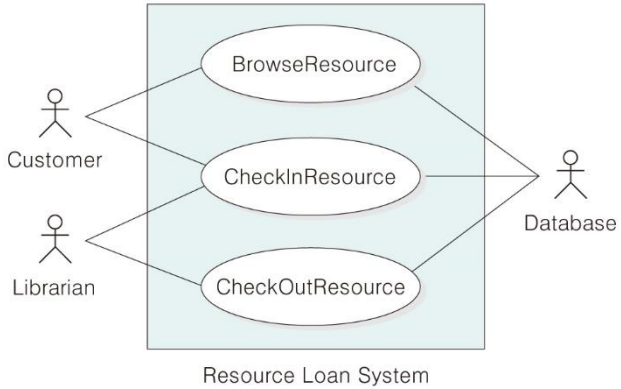
요구사항 추출

1. 요구 공학

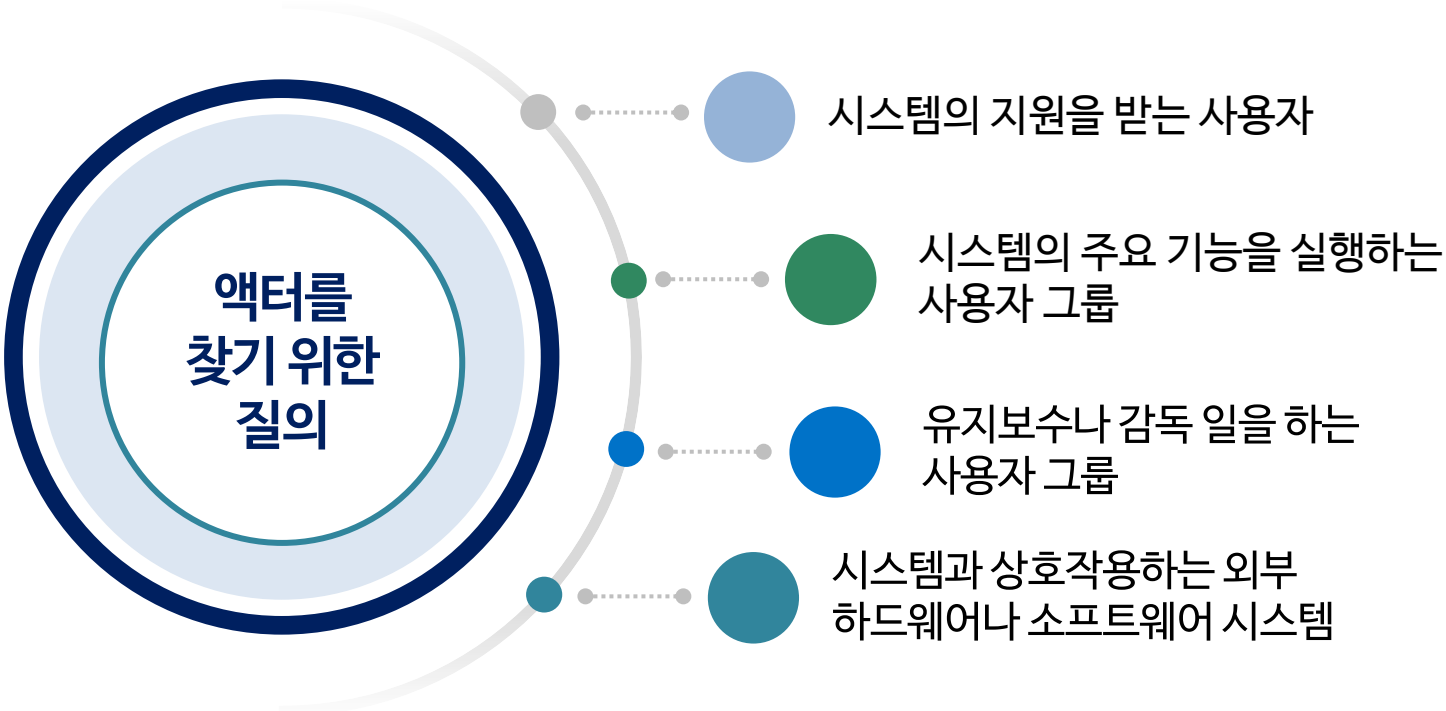
- 고객이 이해할 수 있는 요구사항 명세서를 작성
- 요구사항 명세서는 분석 활동 동안 구조화되고 정형화되어짐
  - 초기 요구사항 명세서는 자연어로 표현됨
  - 같은 정보를 표현하는 것이나 분석 모델은 정형의 표기 기호를 사용함
- 시나리오에 기초한, 요구사항 추출을 위한 활동
  - 액터 찾기, 시나리오 찾기, 유스케이스 찾기, 유스케이스 상세화, 액터와 유스케이스들 간의 관계 찾기, 비기능 요구사항 찾기

2. 액터 찾기 (1/2)

- 액터는 시스템과 상호작용하는 사람이나 외부 시스템
- 액터의 식별은 요구사항 추출의 첫 단계
  - 시스템이 지원해야 하는 사용자를 식별
  - 액터는 시스템 경계의 외부에 존재



2. 액터 찾기 (2/2)


액터를
찾기 위한
질의
시스템의 지원을 받는 사용자
시스템의 주요 기능을 실행하는
사용자 그룹
유지보수나 감독 일을 하는
사용자 그룹
시스템과 상호작용하는 외부
하드웨어나 소프트웨어 시스템

3. 시나리오 찾기 (1/2)

시나리오

- 액터가 보는 시스템의 단일 기능을 이야기 식으로 기술한 것
- 응용 분야의 용어를 사용하여 단일 기능을 표현

+ 시나리오가 유스케이스와 다른 점

× 특정 사례에 초점을 두며 모든 가능한 상황을 기술하는 것은 아님

+ 시나리오를 찾기 위한 질의

× 시스템이 수행해 주길 바라는 작업
× 액터가 엑세스하는 정보
× 액터가 알려 주어야 하는 외부 환경의 변화
× 시스템이 알려 주어야 하는 사건

3. 시나리오 찾기 (2/2)

〈표 9-1〉 학생용 도서 대출 시나리오의 예

시나리오 이름	CheckOutBookForStudent
참여 액터	홍길동: Student 김철수: Librarian
이벤트 흐름	1. 학생 홍길동은 "대지"라는 책을 빌리고자 사서인 김철수에게 책 목록과 도서 카드를 건넨다. 2. 김철수는 작업을 시작한다. 3. 홍길동이 도서 카드를 스캔하거나 학생 ID를 입력하면, 학생 ID가 데이터베이스로 전송된다. 4. 김철수는 데이터베이스로부터 홍길동의 회원 정보를 받고 도서 대출과 관련된 학생의 상태에 문제가 없음을 확인한다. 5. 김철수가 책 "대지"의 바코드를 스캔하면, 책 ID가 데이터베이스에 전송된다. 6. 김철수는 데이터베이스로부터 책 "대지"의 정보를 받는다. 책 "대지"는 대출이 가능한 것으로 나온다. 7. 김철수는 책 "대지"를 대출한다. 8. 학생의 상태에 따라 책 "대지"의 대출 기한이 기록되고, 책의 상태는 대출 중으로 바뀌며, 대출 이력에는 학생 ID가 추가된다.

4. 유스케이스 찾기 (1/4)

- 시나리오는 유스케이스의 인스탄스
  - 시나리오는 특정한 하나의 사례, 유스케이스는 유사 기능의 모든 사례
  - CheckOutResource는 유스케이스, CheckOutCDForProfessor는 시나리오
- 해당 기능의 모든 가능한 시나리오를 명세한 것
  - 액터에 의해 실행되며 상호작용에 관한 완전한 흐름을 표현
  - 실행 후에는 다른 액터와 상호작용도 함
  - 유스케이스의 이름은 액터 입장에서 동사구로 표현

4. 유스케이스 찾기 (2/4)

✚ 참여 액터

– 유스케이스를 시작시키는 액터와 정보를 제공받는 액터

✕유스케이스 명세의 구성

- 선행 조건과 종료 조건:
  유스케이스가 발생될 수 있는 조건과 유스케이스의 종료 조건을 기술
- 기본 이벤트 흐름: 시스템과 액터의 성공적 상호 작용을 기술
- 대체 이벤트 흐름: 부수적이며 선택적인 상호작용을 기술
- 특수 요구사항: 비기능적 요구사항

4. 유스케이스 찾기 (3/4)

유스케이스 이름	CheckOutResource
참여 액터	Librarian은 유스케이스를 시작시키며, Customer는 정보를 제공하거나 제공받는다.
선행 조건	Librarian은 자료 대출 시스템에 유효한 Customer ID와 암호를 입력하여 성공적으로 로그인한다. Resource와 Customer 정보를 가지고 있는 도서 데이터베이스가 가동 중이다.
기본 이벤트 흐름	1. Librarian이 CheckOutResource 기능을 요청하면, 유스케이스가 시작된다. 2. 시스템은 폼을 제공한다. 3. Customer는 폼을 완성하여 제출하거나, 도서 카드를 스캔한다. 4. 시스템은 ValidateCustomer 유스케이스를 시작시킨다. 5. ValidateCustomer 유스케이스가 성공적으로 끝나면, 시스템은 Librarian에게 알린다. 6. Librarian은 Resource의 바코드를 스캔한다. 7. 시스템은 EnterResource 유스케이스를 시작시킨다. 8. EnterResource 유스케이스가 성공적으로 끝나면, 시스템은 DetermineDueDate 유스케이스를 시작시킨다. 9. 시스템은 대출된 Resource의 정보를 Customer에게 디스플레이하고 유스케이스가 종료된다.
대체 흐름	5a. ValidateCustomer 유스케이스에서 유효한 Customer ID가 확인되지 않는다면, 시스템은 대출 요청을 취소시키고 유스케이스가 종료된다. 5b. 유효한 Customer가 18세 이하라면, Librarian은 CheckOutResourceForChild 유스케이스를 시작시킨다. 6a. 바코드가 인식되지 않으면 Resource의 바코드를 직접 입력한다. 8a. EnterResource 유스케이스에서 유효한 Resource ID가 확인되지 않으면, 경고 메시지를 내보내고 DetermineDueDate를 건너뛰고 다음 EnterResource 유스케이스를 시작한다.
종료 조건	ValidateCustomer 유스케이스에서 유효한 Customer ID가 확인되지 않는다면, 시스템은 변경되지 않고 유스케이스가 종료된다. 유효한 Customer ID가 입력되면 Customer 객체에 Resource ID와 대출 기한이 기록된다. Resource 객체에 대출한 Customer ID와 대출 중 상태가 기록된다.
특수 요구사항	Librarian은 30초 내에 Customer의 요청을 처리한다.

4. 유스케이스 찾기 (4/4)

대체 흐름	5a. ValidateCustomer 유스케이스에서 유효한 Customer ID가 확인되지 않는다면, 시스템은 대출 요청을 취소시키고 유스케이스가 종료된다. 5b. 유효한 Customer가 18세 이하라면, Librarian은 CheckOut-ResourceForChild 유스케이스를 시작시킨다. 6a. 바코드가 인식되지 않으면 Resource의 바코드를 직접 입력한다. 8a. EnterResource 유스케이스에서 유효한 Resource ID가 확인되지 않으면, 경고 메시지를 내보내고 DetermineDueDate를 건너뛰고 다음 EnterResource 유스케이스를 시작한다.
종료 조건	ValidateCustomer 유스케이스에서 유효한 Customer ID가 확인되지 않는다면, 시스템은 변경되지 않고 유스케이스가 종료된다. 유효한 Customer ID가 입력되면 Customer 객체에 Resource ID와 대출 기한이 기록된다. Resource 객체에 대출한 Customer ID와 대출 중 상태가 기록된다.
특수 요구사항	Librarian은 30초 내에 Customer의 요청을 처리한다.

5. 유스케이스 상세화

✚ 요구사항을 상세히 명세함

- 시스템이 제공하는 기능을 자세히 기술
- 유스케이스의 완전성과 정확성을 높이는 작업
- 시나리오에서 포함되지 못했던 기능을 파악하여 예외 사항으로 기술하거나 새로운 유스케이스를 추가함

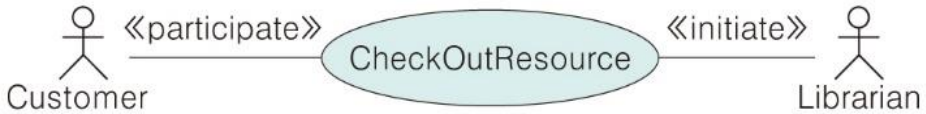
✕ 고려 사항

- 시스템이 다루는 대상를 상세히 함
- 저수준의 상호작용, 예외 사항의 파악과 처리를 명시
- 유스케이스들에서 공통 기능의 추출

6. 액터와 유스케이스 간의 관계 찾기

- 모델의 복잡성을 줄이고 이해도를 높이기 위한 것으로 **통신, 확장, 포함, 상속 관계**를 찾아 정리함

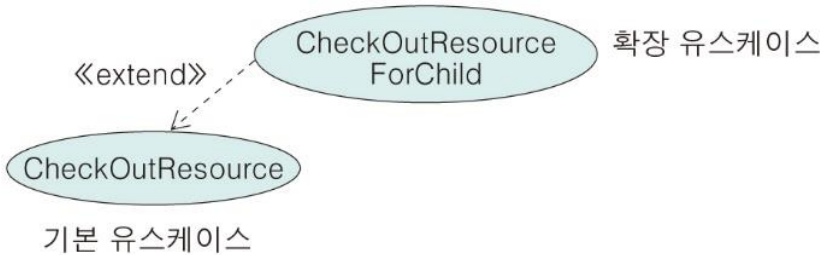
- 통신 관계
  - 유스케이스와 이것을 시작시키는 액터 또는 유스케이스와 통신하는 액터 사이의 관계
  - 시스템의 접근 권한을 보여줌



7. 유스케이스들 간의 관계 찾기 (1/3)

+ 확장 관계

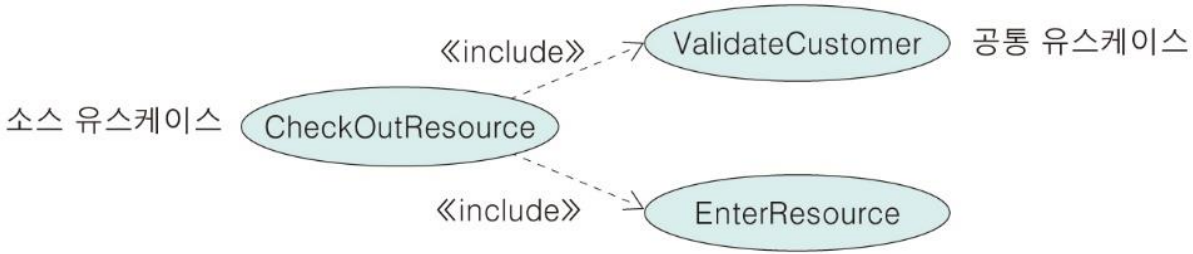
- × 특정 조건하에서 확장된 행위를 포함하면 확장 유스케이스로 분리
- × 확장 유스케이스는 기본 유스케이스에서 예외적이며 선택적인 사건의 흐름을 떼어 내는 것
  - * 기본 유스케이스가 짧아지고 이해가 쉬워짐
  - * 보통의 사례와 예외 사례에 다른 처리를 할 수 있음
- × 기본 유스케이스는 그 자체로 완전한 유스케이스임



7. 유스케이스들 간의 관계 찾기 (2/3)

+ 포함 관계

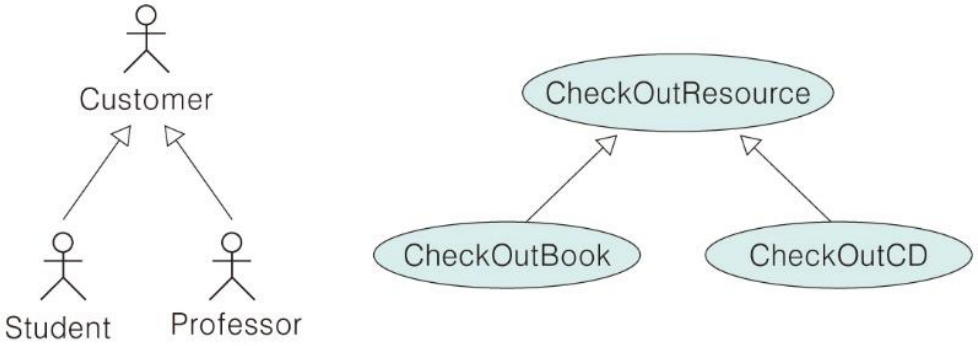
- 유스케이스들에서 존재하는 공통의 기능을 분리
- 소스 유스케이스와 재사용 가능한 공통 유스케이스로 분리
- 중복성을 줄이기 위해 공통 유스케이스로 분리하는 것
- 소스 유스케이스에는 공통의 유스케이스가 반드시 필요함
- 화살표 방향에 유의



7. 유스케이스들 간의 관계 찾기 (3/3)

✚ 상속 관계

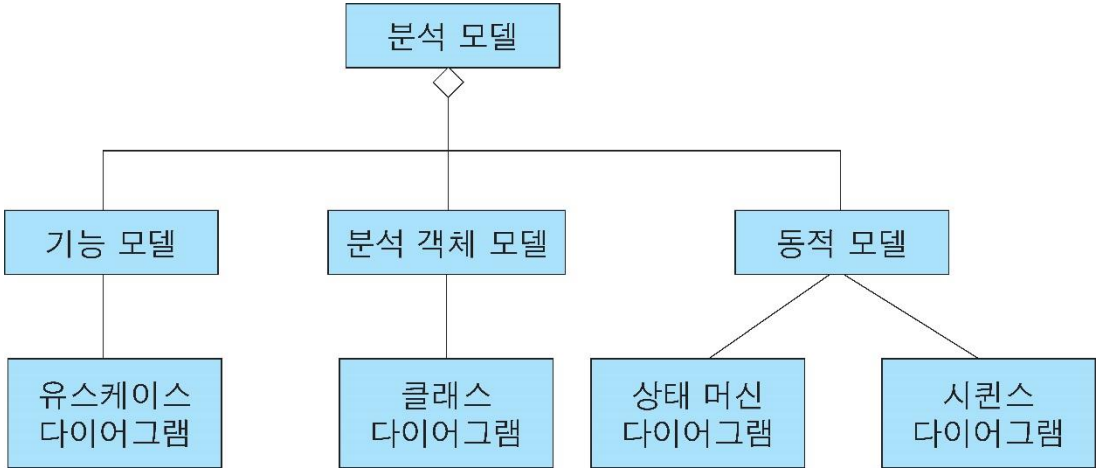
- ✕액터들 사이나 유스케이스들 사이에도 상속 관계가 존재



Chapter. 3

분석

1. 분석 모델 (1/2)


분석 모델
기능 모델
분석 객체 모델
동적 모델
유스케이스
다이어그램
클래스
다이어그램
상태 머신
다이어그램
시퀀스
다이어그램

1. 분석 모델 (2/2)

✕기능 모델

- 요구 추출에 유스케이스와 사용자 스토리 방법을 사용
- 결과물로 유스케이스 다이어그램과 유스케이스 명세를 작성
- 계속적으로 수정 보완됨

+ 분석 객체 모델

- 응용 도메인을 설명하는 개념 모델로서 클래스 다이어그램으로 표현

+ 동적 모델

- 개념 모델을 확장하여 액터와 시스템의 상호작용을 표현(시퀀스 다이어그램)
- 단일 객체의 상태 변화를 모델링(상태 머신 다이어그램)

2. 기능 모델(1/3)

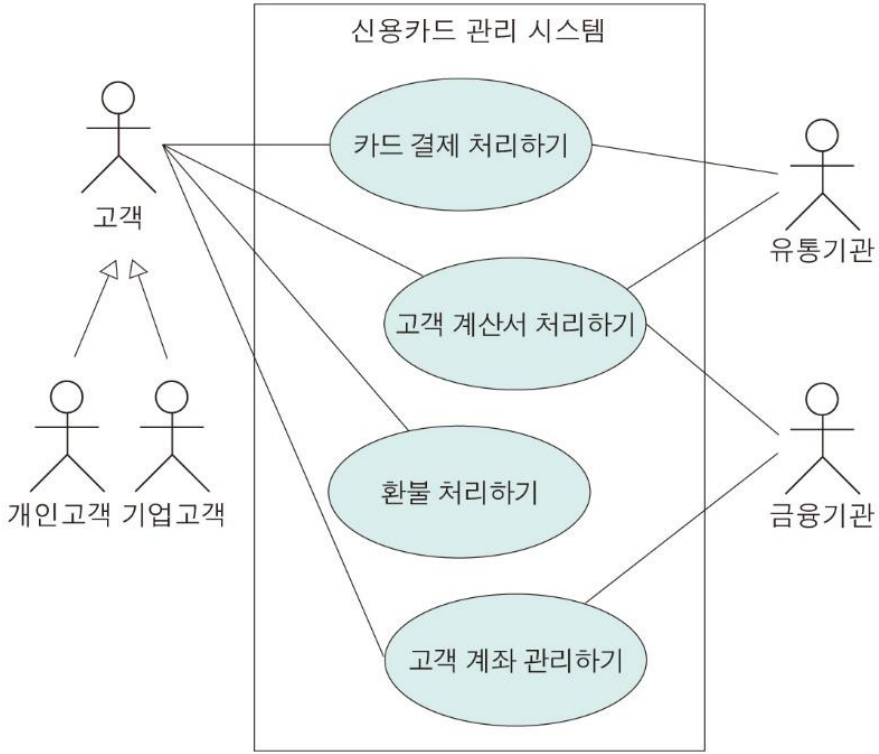
✚ 기능 모델 - 유스케이스와 사용자 스토리

- 객체지향 기반의 요구사항 추출 과정에서 작성되고 분석 과정에서 수정될 수 있음
- 외부에서 접근할 수 있는 기능을 유스케이스로 표현
- 사용자 스토리를 사용하여 요구사항을 모델링 하는 방법도 있음

✖ 유스케이스 다이어그램

- 소프트웨어 시스템과 외부 환경과의 상호 작용을 표현함
- 막대 인간은 액터로 시스템과 상호 작용하는 외부 개체(사람이나 외부 시스템)
- 사각 박스로 시스템의 경계를 표시
- 타원은 유스케이스로 액터가 이용하는 기능
- 직선은 액터가 유스케이스를 시작시키거나, 유스케이스의 결과를 받는 것을 표현

2. 기능 모델(2/3)


신용카드 관리 시스템
카드 결제 처리하기
고객 계산서 처리하기
환불 처리하기
고객 계좌 관리하기
고객
개인고객
기업고객
유통기관
금융기관

2. 기능 모델(3/3)

✚ 유스케이스

- 사용자와 시스템 사이의 일반적 상호작용, 예외적 상호작용 및 사전 조건, 사후 조건을 기술한 구조화된 텍스트
- 반복적 개발 프로세스에서 유스케이스는 점차적으로 구체화됨
- 하나의 유스케이스에서 정의된 행위를 기술하기 위해 시퀀스 다이어그램을 작성하게 됨

✖ 사용자 스토리

- 초기 요구사항 발견과 프로젝트 계획에 사용되는 짧은 대화형의 텍스트
- 애자일 방법에서 널리 사용됨
- 2~4 문장 정도로 시스템이 해야 하는 일을 3x5 인치 카드에 작성

3. 분석 객체 모델

✚ 사용자 관점에서 시스템을 표현

- 다루어야 하는 개별 개념들과 그들 간의 관계를 표현
- 클래스 다이어그램으로 표현
- 개념은 클래스로 표현되며 속성과 오퍼레이션을 가짐
- 분석 모델에서 클래스는 고수준으로 추상화되어 표현됨

✖엔터티/ 경계/ 제어 객체

- 3개의 유형으로 구분함
- 유형을 구분함으로써 작고 특화된 객체를 만들게 되고 변경이 용이해 짐

4. 동적 모델

시퀀스 다이어그램

- 단일 유스케이스에서 객체들의 상호작용을 표현
- 개별 클래스에 책임을 할당
- 설계 과정을 거치면 하나의 클래스는 소스 코드 상에서 하나 이상의 구현 클래스에 대응됨

상태 머신 다이어그램

- 단일 객체 관점에서 시스템의 행위를 표현
- 긴 생명주기와 상태 종속적 행위를 하는 객체들만을 고려하는 것이 좋음
- 대개 제어 객체를 대상으로 작성함
- 작성 과정에서 객체의 행위를 정형화하며 빠뜨린 오퍼레이션을 발견할 수 있음

5. 분석 활동 (1/9)

✚ 자연언어 분석 방법

– 명사는 클래스, have 동사는 집합 관계,
be 동사는 상속 관계, 형용사는 속성이 될 수 있음

✖ 주요 활동

- 엔터티/경계/제어 객체 찾기
- 유스케이스를 시퀀스 다이어그램으로 변환하기
- 연관 찾기, 집합체 연관 찾기
- 속성 찾기
- 객체의 상태 종속적 행위를 모델링 하기
- 객체 간의 상속 관계를 모델링 하기

5. 분석 활동 (2/9)

✚ 엔터티 객체 찾기

- ✕엔터티 객체는 시스템이 유지하는 정보를 표현
- ✕유스케이스 작성을 위해 명확히 규정한 용어, 반복적으로 등장하는 명사
- ✕실세계의 개체나 유지해야 하는 행위 정보, 데이터 소스나 종착지 등

✚ 경계 객체 찾기

- ✕사용자 인터페이스를 모델링 한 것
- ✕액터에게 받은 정보를 엔터티 객체나 제어 객체가 사용할 수 있는 형태로 변환함
- ✕각 액터는 적어도 하나의 경계 객체를 통해 상호작용 함
- ✕사용자 인터페이스 컨트롤, 데이터 입력 폼, 사용자에게 제공하는 메시지 등이 후보가 될 수 있음

5. 분석 활동 (3/9)

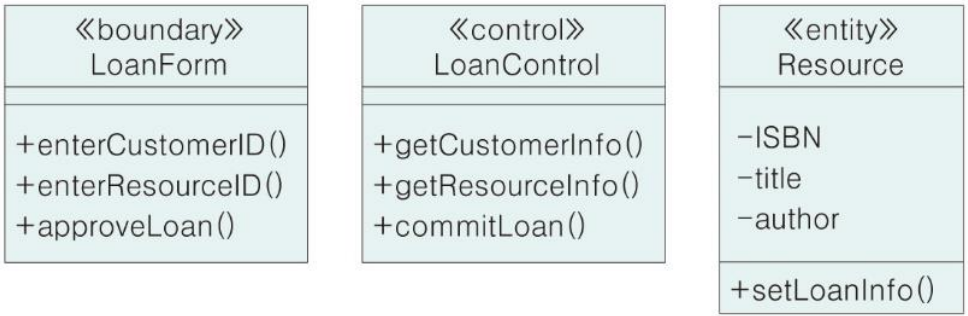
제어 객체 찾기

- 실제적 기능을 수행하는 부분으로 경계 객체와 엔터티 객체 사이의 조정자
- 경계 객체로부터 정보를 받아 엔터티 객체로 보내는 역할
- 유스케이스당 하나 또는 유스케이스의 액터당 하나의 제어 객체가 필요
- 유스케이스가 시작될 때 만들어지고 유스케이스가 끝날 때 없어짐

5. 분석 활동 (4/9)

객체의 종류를 구분하여 표기하는 방법

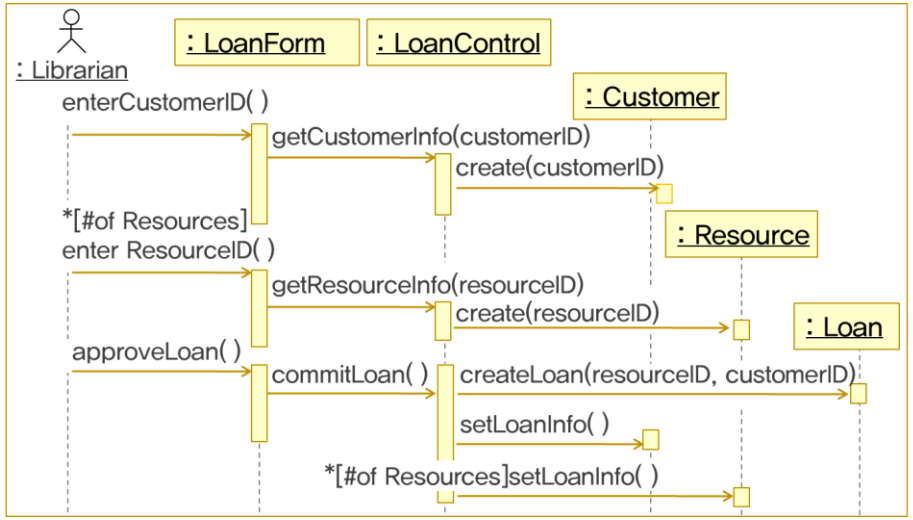
- 스테레오타입을 사용하여 메타 정보를 추가
  - 예) 《boundary》, 《control》, 《entity》
- 이름에 접미사를 사용
  - 예) Loan은 엔터티 객체, LoanForm은 경계 객체, LoanControl은 제어 객체



5. 분석 활동 (5/9)

✚ 유스케이스를 시퀀스 다이어그램으로 변환하기

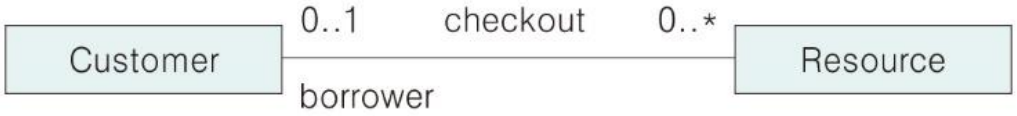
- ✕ 유스케이스를 실현하는 객체들의 협력을 표현
- ✕ 이 과정에서 요구사항 명세서에 빠져 있는 객체나 행위들을 발견할 수 있음



5. 분석 활동 (6/9)

+ 연관 찾기

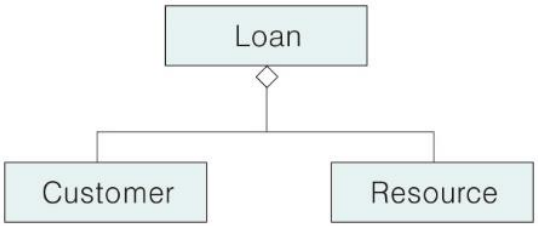
- × 동사나 동사구를 조사
- × 객체들 간의 관계나 상호의존성을 표현
- × 연관의 이름, 연관에 참여하는 클래스의 역할, 관계의 다중성을 표현

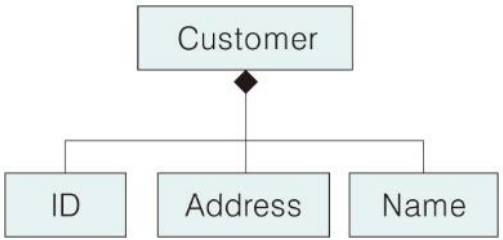


5. 분석 활동 (7/9)

✚ 집합체 연관 찾기

- ✕ 전체-부품 관계(또는 포함 관계)를 표현하는 특별한 유형의 연관
- ✕ 구성 집합체는 부품의 존재가 전체에 의존하는 경우
- ✕ 공유 집합체 연관은 독립적으로 존재하거나 여러 집합체에 의해 공유될 수 있음





5. 분석 활동 (8/9)

속성 찾기

- 객체의 특성을 표현
- 소유의 의미를 가지는 단어나 형용사를 조사
- 구현할 때는 Customer와 Resource 사이의 연관 관계에서 ResourceList가 Customer의 속성으로 표현될 수 있음

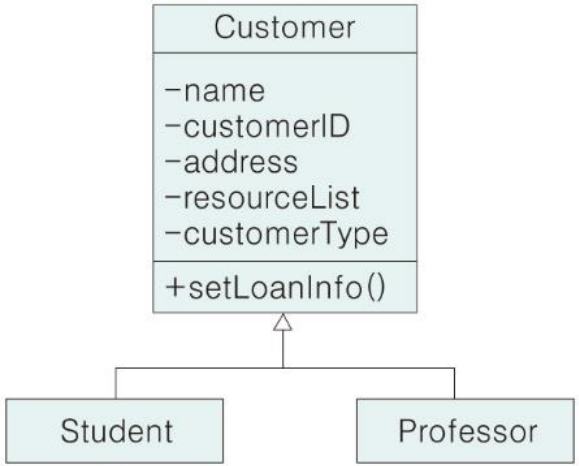
객체의 상태 종속적 행위를 모델링하기

- 상태 머신 다이어그램은 단일 객체의 행위를 기술
- 긴 생명주기와 상태 종속적 행위를 가지는 제어 객체들을 표현

5. 분석 활동 (9/9)

✚ 객체 간의 상속 관계 모델링하기

- ✕상속 관계를 이용하여 모델에 등장하는 중복성을 제거할 수 있음



Chapter. 4

시스템 설계

1. 시스템 설계

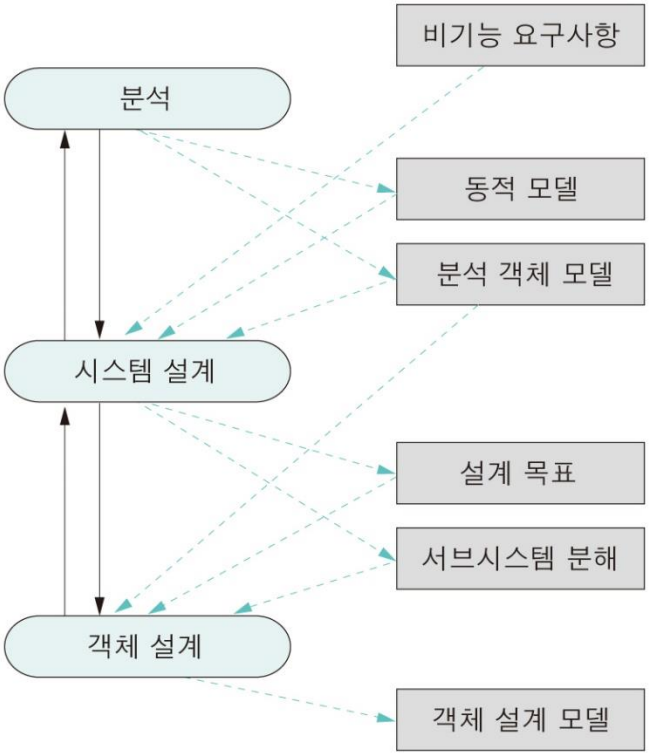
✚ 분석 모델을 시스템 설계 모델로 변환하는 작업

- 분석 모델은 시스템의 내부 구조, 하드웨어 형상 및 구현 방법 등에 관한 정보를 포함하지 않음

✕ 시스템 설계의 결과

- 설계 목표의 정의: 시스템의 품질을 기술
- **소프트웨어 아키텍처의 결정**
- 경계 조건의 기술: 시스템의 설치, 시작, 종료 및 예외 처리 사항

2. 시스템 설계와 모델



3. 설계 목표의 설정 (1/2)

✕ 설계 목표의 설정은 시스템 설계의 시작으로
대개 비기능 요구사항과 응용 도메인으로부터 유도됨

✚ 성능 요인
- 반응 속도, 처리량, 메모리 용량 등

✚ 의존성 요인
- 강건성: 부정확한 사용자 입력에 견디는 능력
- 신뢰도: 고장 없이 명세된대로 행위하는 능력
- 가용성: 시스템을 사용할 수 있는 시간의 백분율
- 결함 내성: 잘못된 조건하에서 동작할 수 있는 능력
- 보안: 악의적 공격을 이길 수 있는 능력
- 안전: 오류나 고장이 생겨도 생명의 위협을 피할 수 있는 능력

3. 설계 목표의 설정 (2/2)

비용 요인

- 개발과 배포에 드는 비용
- 유지보수 및 관리에 드는 비용
- 새 제품으로의 전환 비용이나 하위 호환성을 보장하는 비용

유지보수 요인

- 확장성, 수정성, 적응성, 이식성, 가독성, 요구사항 추적성 등

사용자 요인

- 유용성, 사용성 등

4. 아키텍처 설계에서 다루는 문제 (1/2)

✕ 설계 목표에 맞는 아키텍처를 결정할 때의 고려 사항

- 표준 아키텍처 스타일의 활용도 고려해야 함

✚ 하드웨어와 소프트웨어의 연결

- 어떤 하드웨어에 어느 기능을 배치할 것인가
- 컴퓨터들 간의 통신을 어떻게 구현할 것인가
- 기성 컴포넌트나 COTS를 이용할 수 있는가

✚ 데이터 관리

- 데이터베이스 관리 시스템이나 데이터 관리를 전담하는 서브시스템의 선정

4. 아키텍처 설계에서 다루는 문제 (2/2)

접근 제어

- 데이터 접근에 관한 정책을 정하고 어떻게 구현할지 정함

제어 흐름

- 시스템의 동작 순서
- 이벤트 중심 제어나 동시 사용자의 허용 여부를 결정

경계 조건

- 시스템이 어떻게 시작되고 종료되는가
- 예외 상황을 어떻게 처리할 것인가

Chapter. 5

통합 프로세스(Unified Process)

1. 통합 프로세스(UP)

✚ UP는 객체지향 분석과 설계를 위한 방법론

- 여러 개발 방법론의 장점을 통합
- 반복적이고 점증적인 프로세스를 위한 프레임워크
- 개발 조직이나 프로젝트에 맞추어 사용될 수 있음
- RUP는 UP를 상세히 다듬고 HTML로 문서화한 제품

✖ UP가 강조하는 사항

- 반복적 개발
- 유스케이스 기반의 개발
- 아키텍처를 중요시 함
- 프로젝트 초기에 중요한 위험을 다룰 것

2. UP 생명주기 (1/2)

✚ RUP는 작업을 프로세스 분야(discipline)에 따라 구분함

- 6개의 공학 분야(비즈니스 모델링, 요구사항, 분석과 설계, 구현, 테스트, 배포)
- 3개의 지원 분야(형상 관리, 프로젝트 관리, 환경)

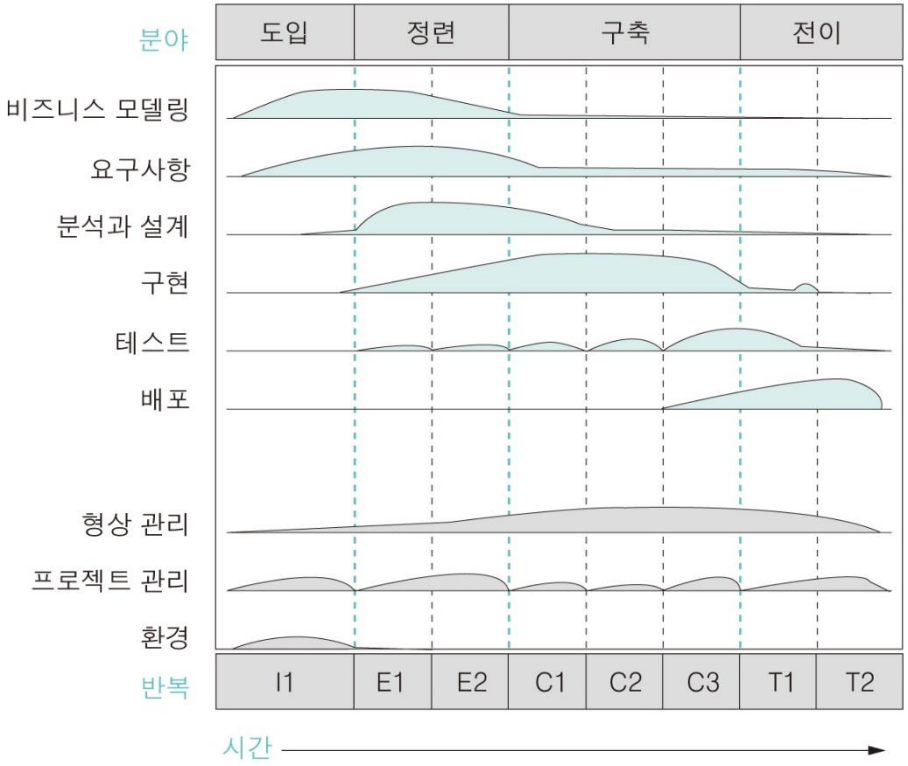
✚ UP 생명주기

- 도입, 정련, 구축, 전이의 4 단계로 구성됨
- 정련, 구축, 전이는 각각 일련의 반복으로 구성됨

✖도입, 정련, 구축, 전이의 반복 단계에서 각 분야의 비중이 다름

- 초기에는 요구사항, 분석과 설계가 비중이 큼
- 후반에는 구현, 테스트의 비중이 높아짐

2. UP 생명주기 (2/2)


분야
도입
정련
구축
전이
비즈니스 모델링
요구사항
분석과 설계
구현
테스트
배포
형상 관리
프로젝트 관리
환경
반복
I1
E1
E2
C1
C2
C3
T1
T2
시간

3. RUP 4단계 (1/2)

도입(Inception)

- 1주 정도의 짧은 기간에 수행
- 시스템의 범위를 정하며, 비즈니스 사례를 파악하고 비전을 세움
- 주요 요구사항을 나열. 10% 정도의 유스케이스를 상세히 작성
- 가능성 있는 해결 방안과 아키텍처를 검토하고 위험 요소를 식별
- 일정과 비용의 개략적 추정. 실현 가능성을 조사

정련(Elaboration)

- 핵심 아키텍처를 구축하고 대부분의 요구사항을 명확히 정의
- 정련의 초기에 30%, 종료까지 80% 정도의 유스케이스를 상세히 작성
- 높은 위험 요소를 해결하고, 일정과 자원을 상세히 추정
- 2~6주 기간의 반복을 2~4회 수행함
- 중요 요구사항을 설계하고 구현. 전체적으로 15% 정도를 구현

3. RUP 4단계 (2/2)

구축(Construction)

- 남아 있는 부분을 설계하고 구현하여 통합함
- 최종적으로 고객에게 인도할 준비를 함

전이(Transition)

- 사용자 환경으로 시스템을 옮김
- 반복은 사용자 피드백을 받아 보수하는 작업

다음강의

10강. 유스케이스 다이어그램 및 명세