소프트웨어공학

UML 및 UseCase 다이어그램 PART 1

수업 내용

- 수업 내용
  - UML 및 UseCase 다이어그램 보다 명확하고 상세한 개념과 실제 사용 예시
    - 연관 관계 (Association), 포함 관계 (Include), 확장 관계 (Extend), 일반화 관계 (Generalization)
  - Enterprise Architect (trial edition) 도구

설계나 구현도 마찬가지이지만

소프트웨어를 구현하는 방법이 수학처럼 딱하나의 정답만 있는 것은 아니고,

또 개발자의 생각대로 달라질 수 있는 부분이고,

나중에 설계 및 구현 과정에서도 수정이 가능하기 때문에

처음부터 너무 부담 갖거나, 너무 완벽히 하려고 하지 않아도 괜찮습니다.

수업 내용

- 수업 내용
  - UML 및 UseCase 다이어그램 보다 명확하고 상세한 개념과 실제 사용 예시
    - 연관 관계 (Association), 포함 관계 (Include), 확장 관계 (Extend), 일반화 관계 (Generalization)
  - Enterprise Architect (trial edition) 도구

설계나 구현도 마찬가지이지만

소프트웨어를 구현하는 방법이 수학처럼 딱하나의 정답만 있는 것은 아니고,

또 개발자의 생각대로 달라질 수 있는 부분이고,

나중에 설계 및 구현 과정에서도 수정이 가능하기 때문에

처음부터 너무 부담 갖거나, 너무 완벽히 하려고 하지 않아도 괜찮습니다.

요구사항의 표현

- 표현과 모델의 이해
  - 표현
    - 생각하거나 느낀 것을 표현하는 방법은 다양
    - 수학 공식은 (글로 표현하면 너무 길고 불분명해질 수 있는 내용을) 공통으로 이해할 수 있고, 오해의 소지가 없는 특정 기호로 표현
  - 모델
    - 표현을 위해 사용하는 악보나 수학 공식은 하나의 모델
    - 장난감 자동차나 로봇, 생명 과학자들이 사용하는 DNA 분자 모형 등 또한 모델의 종류

요구사항의 표현

- 모델의 정의와 필요성

  - ‘복잡한 대상의 핵심 특징만 선별해 일정한 관점으로 단순화한 뒤 기호나 그림 등을 사용해 체계적으로 표현한 것

  - 모델이 필요한 가장 큰 이유는 실제 모습을 미리 확인하기 위함(예: 모델하우스)

  - 하나의 사물도 여러 모델이 필요(예: 건물 건설시 여러 관점과 사용 목적에 따라 다수의 도면이 필요)

- 소프트웨어 개발 모델

  - 건축에 여러 도면이 필요하듯이 소프트웨어를 개발할 때도 여러 종류의 모델이 필요

  - UML 등의 다양한 다이어그램을 통해 개발 소프트웨어의 범위나 개략적인 구조와 기능을 이해

소프트웨어개발과 모델

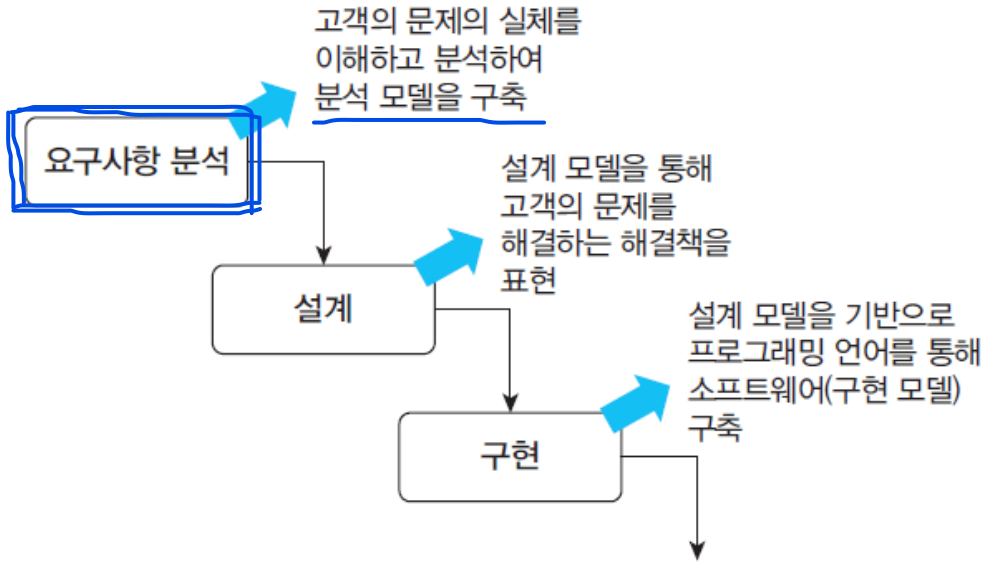


그림 2-1 소프트웨어 개발과 모델

- 요구사항 모델의 역할
  - ✓ 서로의 해석을 공유해 합의를 이루거나 타당성을 검토
  - ✓ 현재 시스템 또는 앞으로 개발할 시스템의 원하는 모습을 가시화
  - ✓ 시스템의 구조와 행위를 명세
  - ✓ 시스템을 구축하는 틀 제공

모델링

- 모델링
  - 모델링의 개요
    - 앞서 설명한 모델을 만드는 과정(작업), 구체화 · 가시화 · 최적화 가능(원활한 요구사항 분석)
    - 연필로 종이에 간단한 콘티를 작성하거나 스토리보드같이 자기가 표현하고 싶은 것을 문장으로 서술하는 것도 한 종류
    - 각 전문 분야의 모델링은 기호나 다이어그램, 표기법 등을 사용해서 차이가 없이 일관성 있는 해석이 가능하도록 표현

주로 다이어그램을 많이 활용

- 모델링, 모델을 만드는 것(모델 : 진짜가 아닌데 진짜 같이 보이는 것)
- 소프트웨어를 직접 짜기 오래걸리고 힘드니까 모델을 만드는 것
- 가장 처음 하는게 => 요구사항 분석 모델링 -> 설계 모델링
  => 실제 실행되는 모습이 눈에 딱 들어와야 한다.
- 또한 요구사항 분석 모델을 보는 순간, 소스코드가 보여야한다.
- 작명도 요구사항 분석할 때 50%는 결정된다.

모델링

- 모델링 표현 방법
  - 자연어를 사용한 표현
    - 사용자 요구사항을 자연어로 표현(모델링)하는 것
    - 도구나 기술을 따로 익힐 필요가 없고 사용자와 대화할 때도 이해하기 쉬움
    - 표현이 길어질 수 있고 해석이 달라질 수 있으며 표현된 내용을 검증하기가 어려움(모호성 존재 가능)

  - UML 다이어그램을 사용한 표현
    - UML 다이어그램을 사용하면 개발할 소프트웨어를 가시적으로 볼 수 있고, 개발할 소프트 웨어에 대한 문서화도 가능
    - 이 문서들은 분석 및 설계 과정에서 유용하며 검증 자료로도 활용 가능

※ [참고] 형식 언어를 사용한 표현도 있음
- 친숙하지는 않지만 문법과 의미가 수학을 기초로 작성되어서 간결하고 정확하게 표현할 수 있음
- 표기법을 별도로 공부해야 하므로 일반 개발자가 사용하기에는 조금 어려운 편
  .관계형 표기법: 함의 방정식, 순환 관계, 대수 공리, 정규 표현법을 사용
  .상태 위주 표기법: 의사결정 테이블, 이벤트 테이블 , 전이 테이블, 유한 상태 기계, 페트리 넷을 사용

UML(Unified Modeling Language)

- 객체지향 모델링을 위해 개발된 대표적인 표준 모델링 방법
- UML은 객체 지향 방식에서 진화했으며, 객체 지향 프로그래밍 스타일을 지원하는 많은 기능을 가지고 있음, 객체 지향 스타일의 모델링, 설계, 프로그래밍을 지원
- 따라서 객체 지향 프로그래밍에 익숙하다면 UML의 기능을 더 쉽게 이해할 수 있음

- UML을 통한 객체 지향 모델링의 장점은
  요구사항 분석 모델과 설계 모델을 표현하는 방식이 다르지 않다는 점

❖ 요구사항 분석, 설계 구현 등의 시스템 개발 과정에서 의사소통이 원활하게 이루어지도록 표준화한 모델링 언어

UML(Unified Modeling Language)

- 소프트웨어 모델링에 사용되는 실질적인 표준
- UML에는 여러가지 다이어그램 존재
  - 소프트웨어 모델링 언어로 시스템의 동작을 모형화

  - 이번 주, 5주차 수업에서는 유스케이스 다이어그램 학습 (6주차 수업에서는 클래스 다이어그램)

분류	다이어그램 유형		목적
구조 다이어그램 (structure diagram)	클래스 다이어그램 (class diagram)		시스템을 구성하는 클래스들 사이의 관계를 표현한다.
	객체 다이어그램 (object diagram)		객체 정보를 보여준다.
	복합체 구조 다이어그램 (composite structure diagram)		복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	배치 다이어그램 (deployment diagram)		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	컴포넌트 다이어그램 (component diagram)		컴포넌트 구조 사이의 관계를 표현한다.
	패키지 다이어그램 (package diagram)		클래스나 유즈 케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램 (behavior diagram)	활동 다이어그램 (activity diagram)		업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	상태 머신 다이어그램 (state machine diagram)		객체의 생명주기를 표현한다.
	유즈 케이스 다이어그램 (use case diagram)		사용자 관점에서 시스템 행위를 표현한다.
	상호작용 다이어그램 (interaction diagram)	순차 다이어그램 (sequence diagram)	시간 흐름에 따른 객체 사이의 상호작용을 표현한다.
		상호작용 개요 다이어그램 (interaction overview diagram)	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
		통신 다이어그램 (communication diagram)	객체 사이의 관계를 중심으로 상호작용을 표현한다.
		타이밍 다이어그램 (timing diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

유스케이스 다이어그램을 사용한 모델링

유스케이스 다이어그램을 사용한 모델링

☞ 기능 요구사항 분석 모델링에 유용하며 전체 범위와 방향 파악에 용이 시각적인 표현방법

- 사용자의 관점에서 시스템의 서비스와 기능 및 그와 관련된 외부 요소를 보여주는 다이어그램
- 시스템을 사용하는 목적들, 즉 기능들을 사용자 관점에서 기술한 다이어그램이며

  이 목적 달성을 위한 사용자와 시스템 사이의 상호작용을 보여준다.

  - 유스케이스 다이어그램을 통해 시스템이 제공하는 기능이나 서비스 등을 정의하고, 시스템의 범위를 결정

- 구성요소
  - 시스템 범위(경계)
  - 액터
  - 유스케이스
  - 관계

※ 표준화 그룹 중의 하나인 OMG (Object Management Group)에서 시스템의 분석 및 설계를 위한 도구로 재정한 M=UMLdml 유스케이스를 기반으로 시스템을 분석 및 명세화하는 방법

다이어그램으로 표현하지 못한 유스케이스들의 구체적인 내용을 각각 서술하는데 사용

※ 최종적으로 완성되는 것은 기본적으로 **유스케이스 다이어그램**과 **유스케이스 기술서 (유스케이스별 명세서)**

유스케이스 모델링 과정

- 유스케이스 모델링(분석) 과정

  1) 시스템 경계 (System Boundary) 정하기

  - 시스템 경계를 설정한다.

  2) 액터 (Actor) 찾기

  - 시스템을 사용하는 사람들을 추출한다. 예) 사용자, 유지보수자, 외부 연동 시스템 등

  3) 유스케이스 찾기

  - 시스템의 기능을 유스케이스 단위로 분리한다

  4)유스케이스 사이의 관계 찾기

  - 연관 관계, 포함 관계, 확장 관계, 일반화 관계

시스템 범위(경계), System Boundary

사각형의 틀을 그리고 그 안의 상단에 시스템 명칭을 기술

- 시스템 안과 밖의 구성요소를 명확히 분리

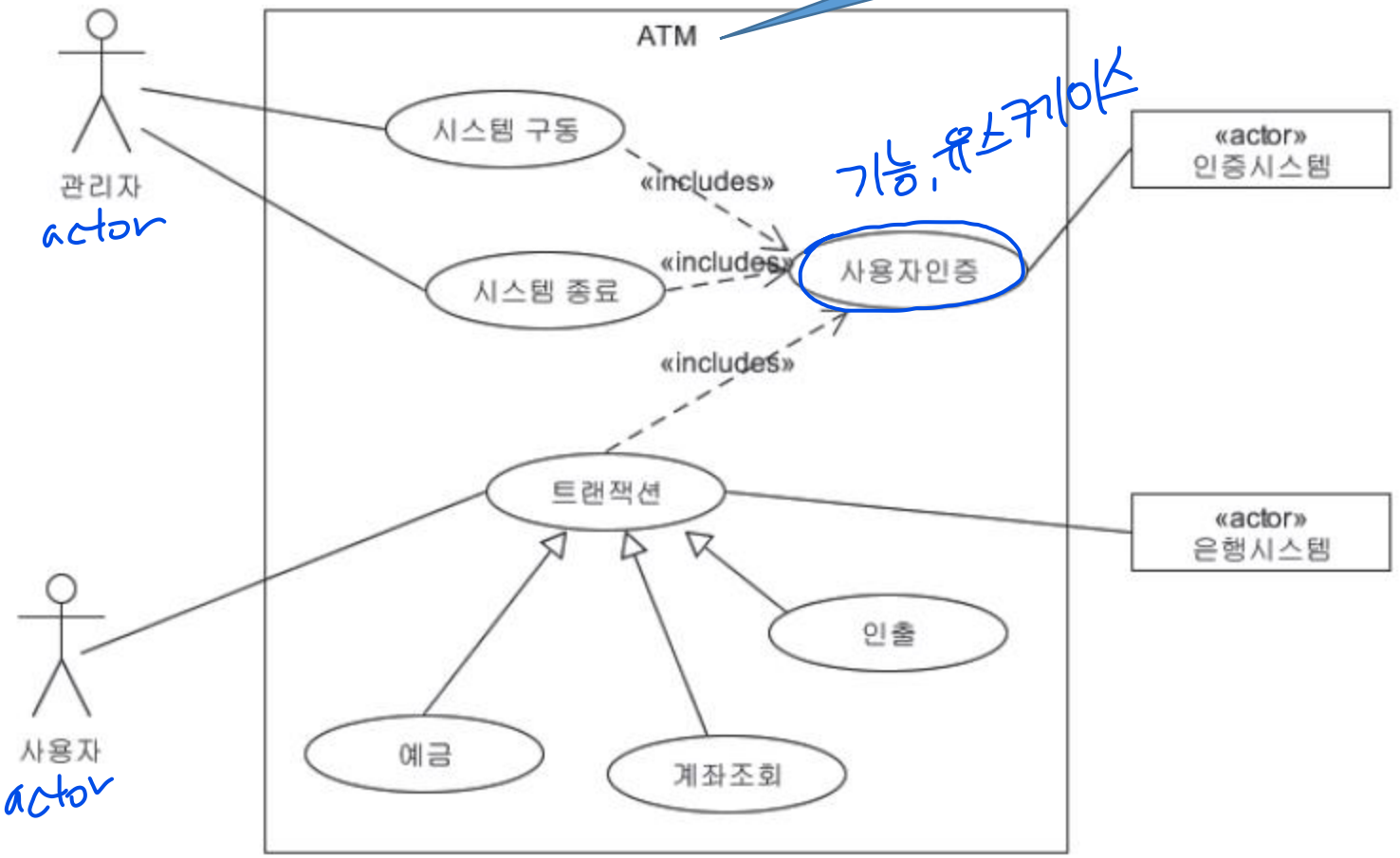
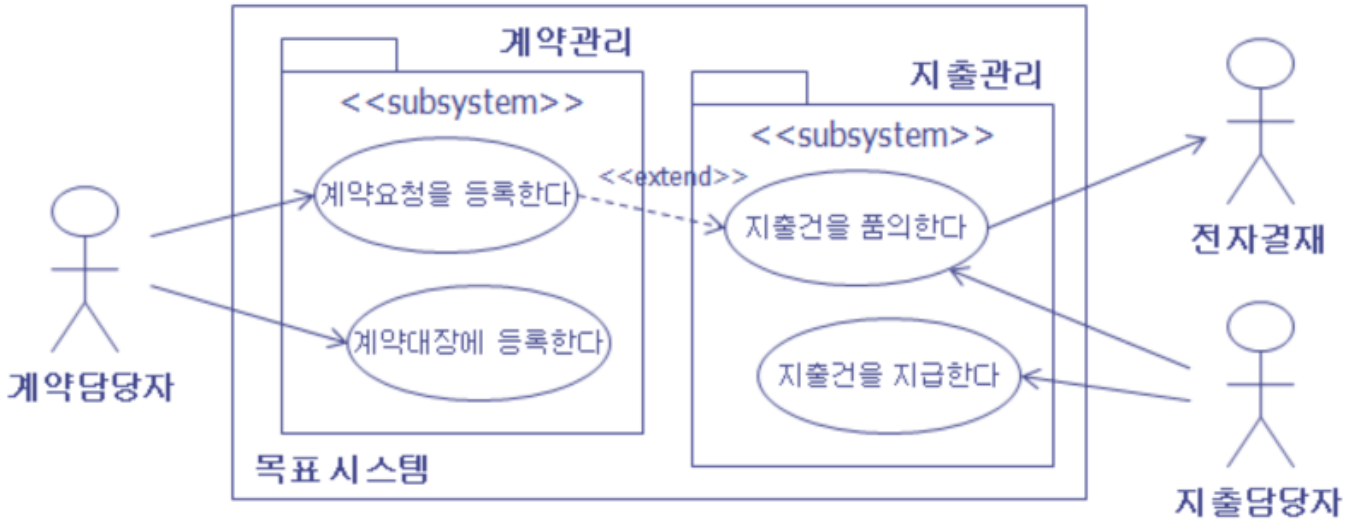


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

시스템 범위(경계), System Boundary

- 2개의 서브시스템을 갖는 유스케이스 모델 여러개도 가능

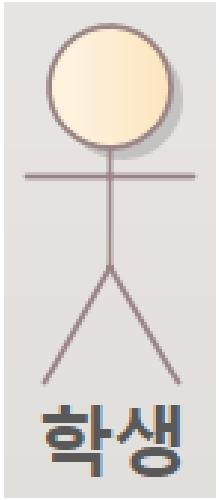


(그림 4) 두 개의 서브젝트를 갖는 유스케이스 모델

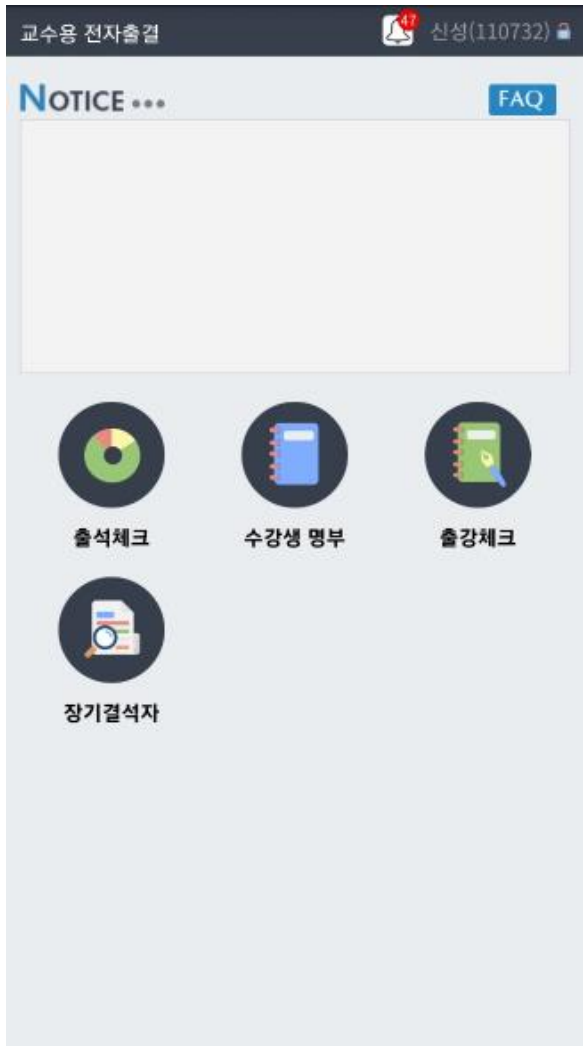
☞ *조준수, 정기원, '서비스 지향 컴퓨팅을 위한 확장 유스케이스 모델링', 한국 인터넷 정보학회 논문지(10권5호)*

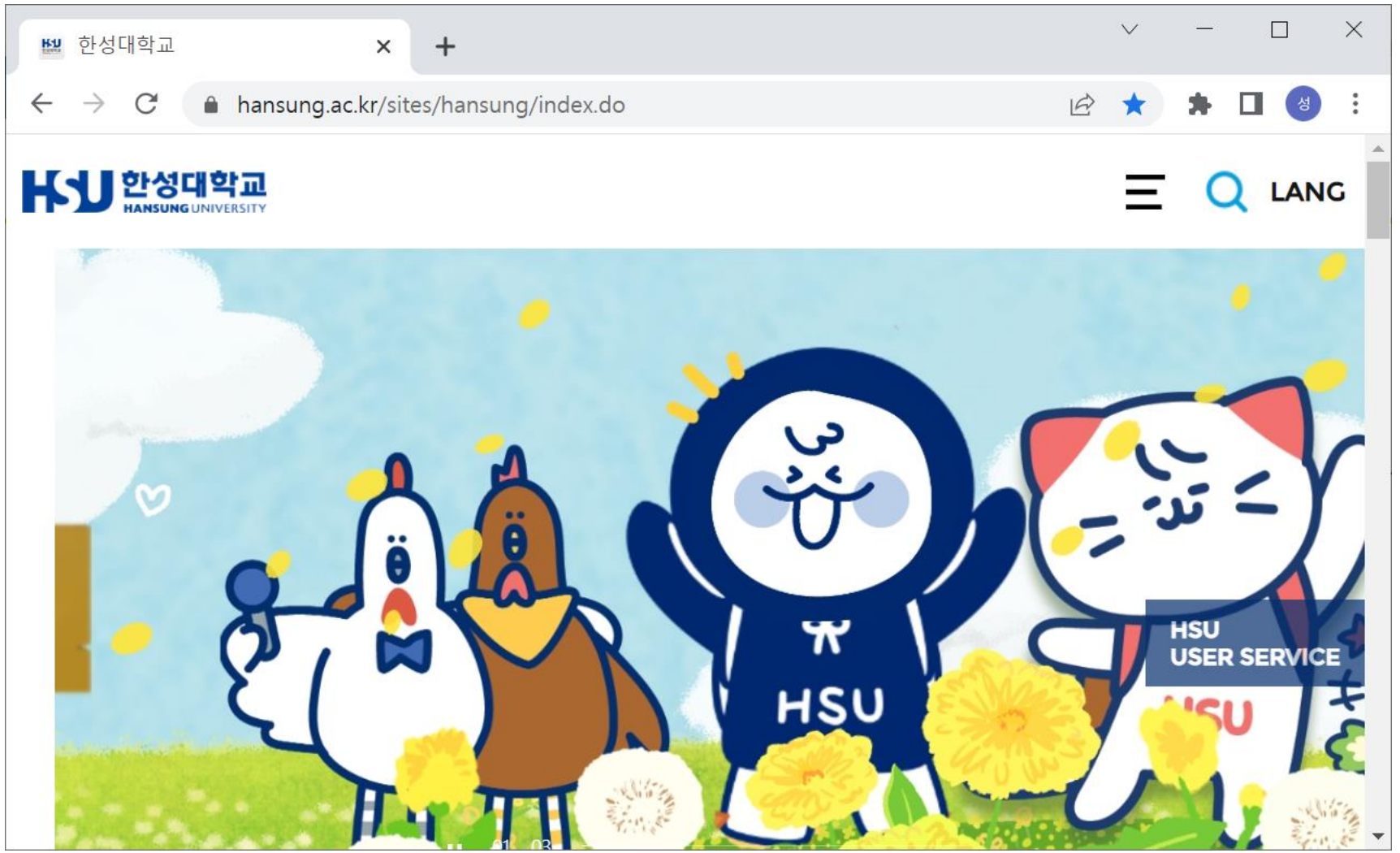
액터(Actor)

- 시스템 외부에 존재하며 시스템과 상호작용을 하는 모든 것들을 나타낸다.
  - 액터는 사람이 될 수도 있고(이해 관계자), 외부 시스템이 될 수도 있다.
- 시스템과 상호 작용하는 외부 개체가 사람이라면 막대 사람 모양으로 표현하고,
  시스템의 경우 박스에 《actor》 표기 [단, 시스템도 사람 모양으로 표현 가능]
- 위 또는 아래에 액터명을 표시

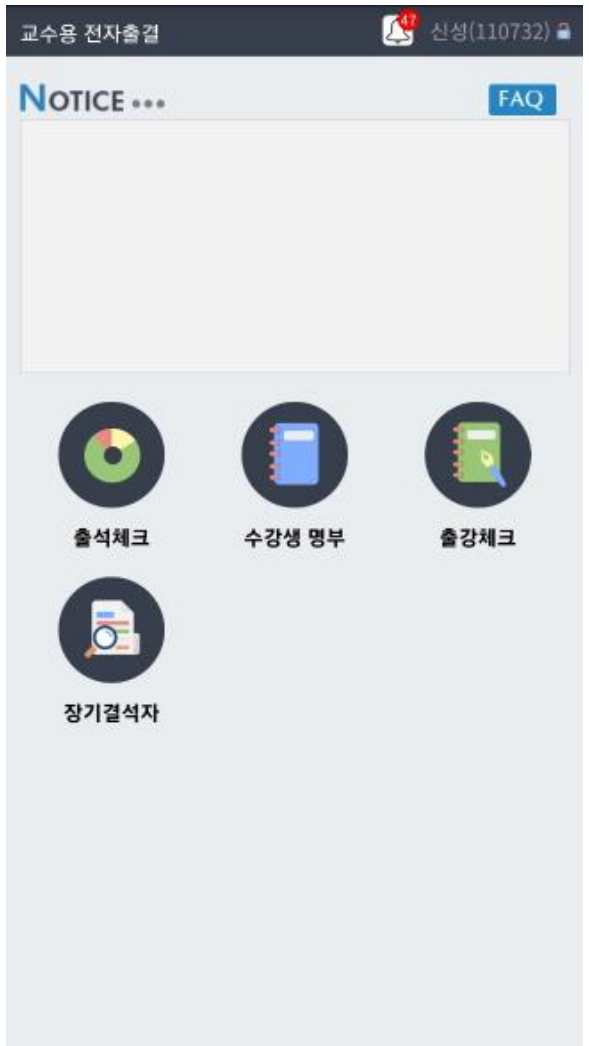


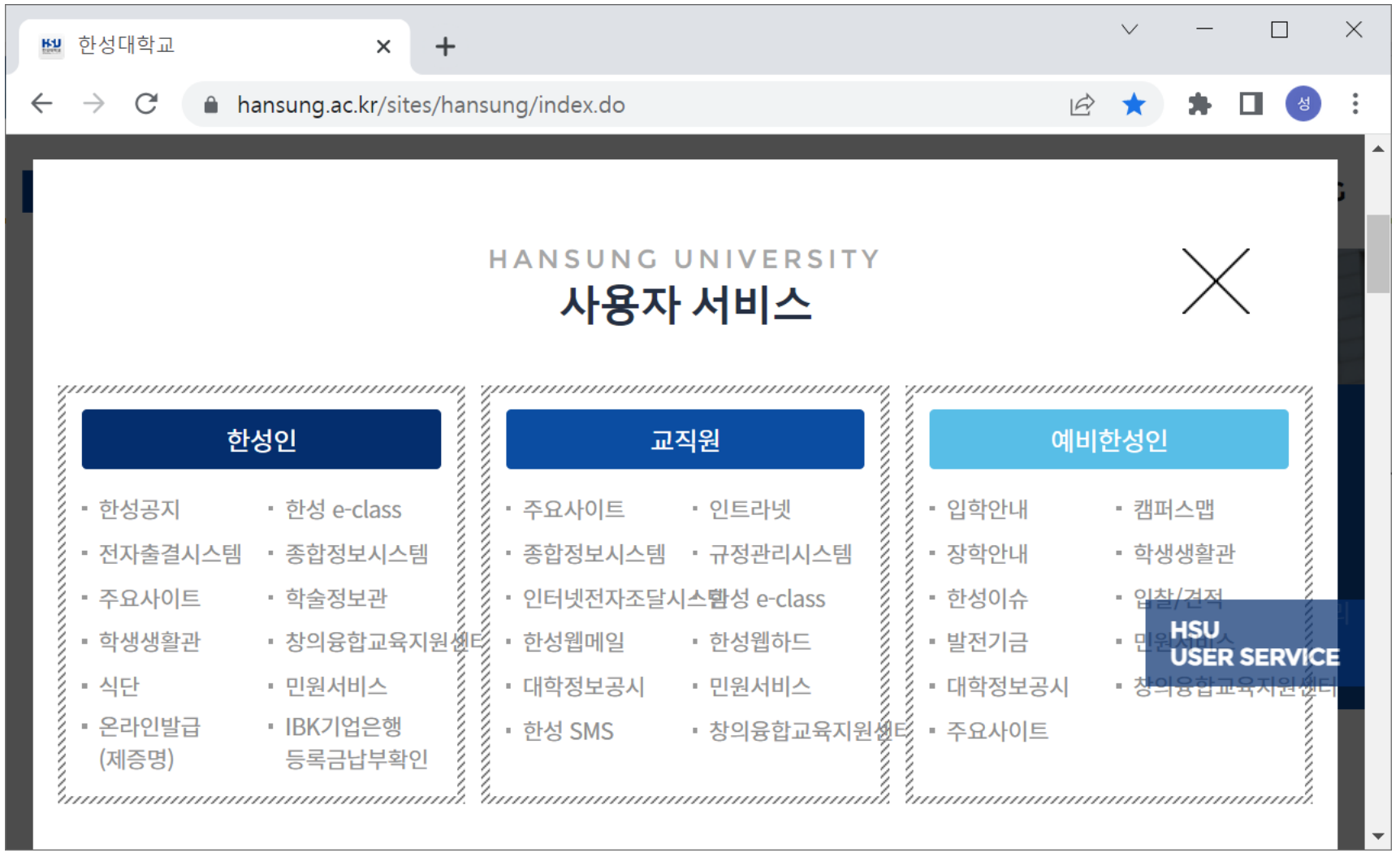
☞ 엑터는 대상 시스템에게 서비스를 제공하거나 제공받는다.


교수용 전자출결
신성(110732)
NOTICE
FAQ
출석체크
수강생 명부
출강체크
장기결석자


한성대학교
hansung.ac.kr/sites/hansung/index.do
HSU 한성대학교
HANSUNG UNIVERSITY
LANG
HSU
USER SERVICE

외부 시스템


교수용 전자출결
신성(110732)
NOTICE
FAQ
출석체크
수강생 명부
출강체크
장기결석자


한성대학교
hansung.ac.kr/sites/hansung/index.do
HANSUNG UNIVERSITY
사용자 서비스
한성인
한성공지
한성 e-class
전자출결시스템
종합정보시스템
주요사이트
학술정보관
학생생활관
창의융합교육지원센터
식단
민원서비스
온라인발급 (제증명)
IBK기업은행 등록금납부확인
교직원
주요사이트
인트라넷
종합정보시스템
규정관리시스템
인터넷전자조달시스템
한성 e-class
한성웹메일
한성웹하드
대학정보공시
민원서비스
한성 SMS
창의융합교육지원센터
예비한성인
입학안내
캠퍼스맵
장학안내
학생생활관
한성이슈
입찰/견적
발전기금
민원서비스
대학정보공시
창의융합교육지원센터
주요사이트
HSU
USER SERVICE

외부 시스템

액터(Actor)

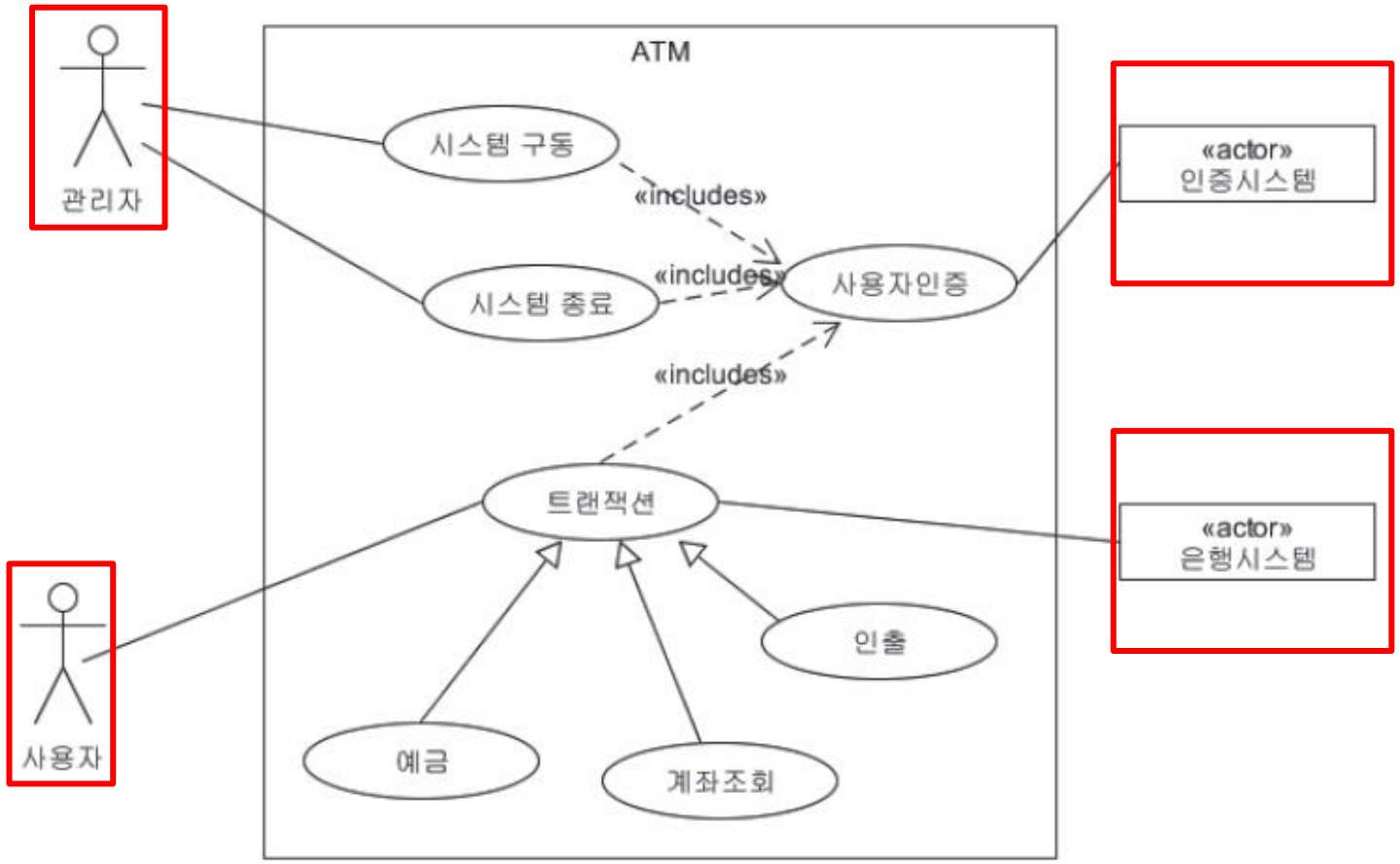


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

※ 참고

- 액터의 종류에도 주(primary) 액터와 부(secondary) 액터가 있다. 주 액터는 목적 달성을 위해 시스템의 서비스를 필요로 하는 액터를 의미하며
- 부 액터는 주 액터의 목적달성을 위해 시스템에 서비스를 제공하는 외부 개체를 의미한다.

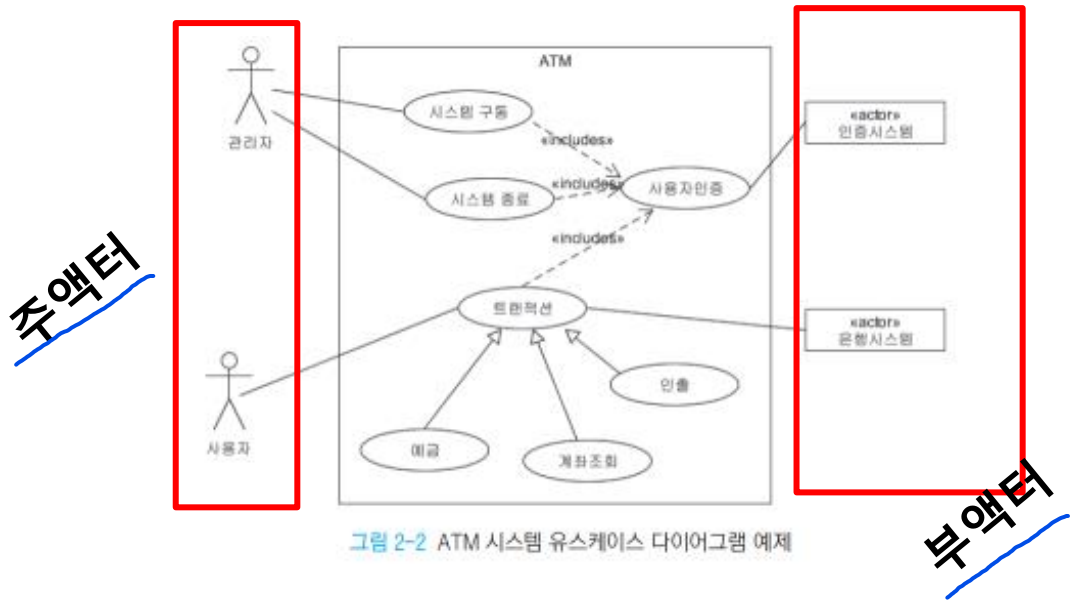


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

T h a n k y o u

TECHNOLOGY

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Velit ex
ullicabo ipsum, labore sed tempora ratione asperiores des
cipserat bore sed tempora rati jgert one bore sed tem!

소프트웨어공학

UML 및 UseCase 다이어그램 PART 2

유스케이스 (UseCase)

- 유스케이스는 엑터에게 보이는 시스템의 기능 (기능 요구사항)
  - 시스템이 실행하는 일련의 동작으로 특정 액터에게 관찰 가능한 실행 결과를 제공
- 유스케이스 이름은 상호작용을 수행하는 목적을 단순하게 명료하게 기술
  - 시스템에 존재하는 어휘라면 제약이 없으나, 비교적 간단하고 이해하기 쉽게 만드는 것이 바람직

☞ 타원형으로 표시, 시스템 안의 각 단위 기능을 의미

유스케이스 (UseCase)

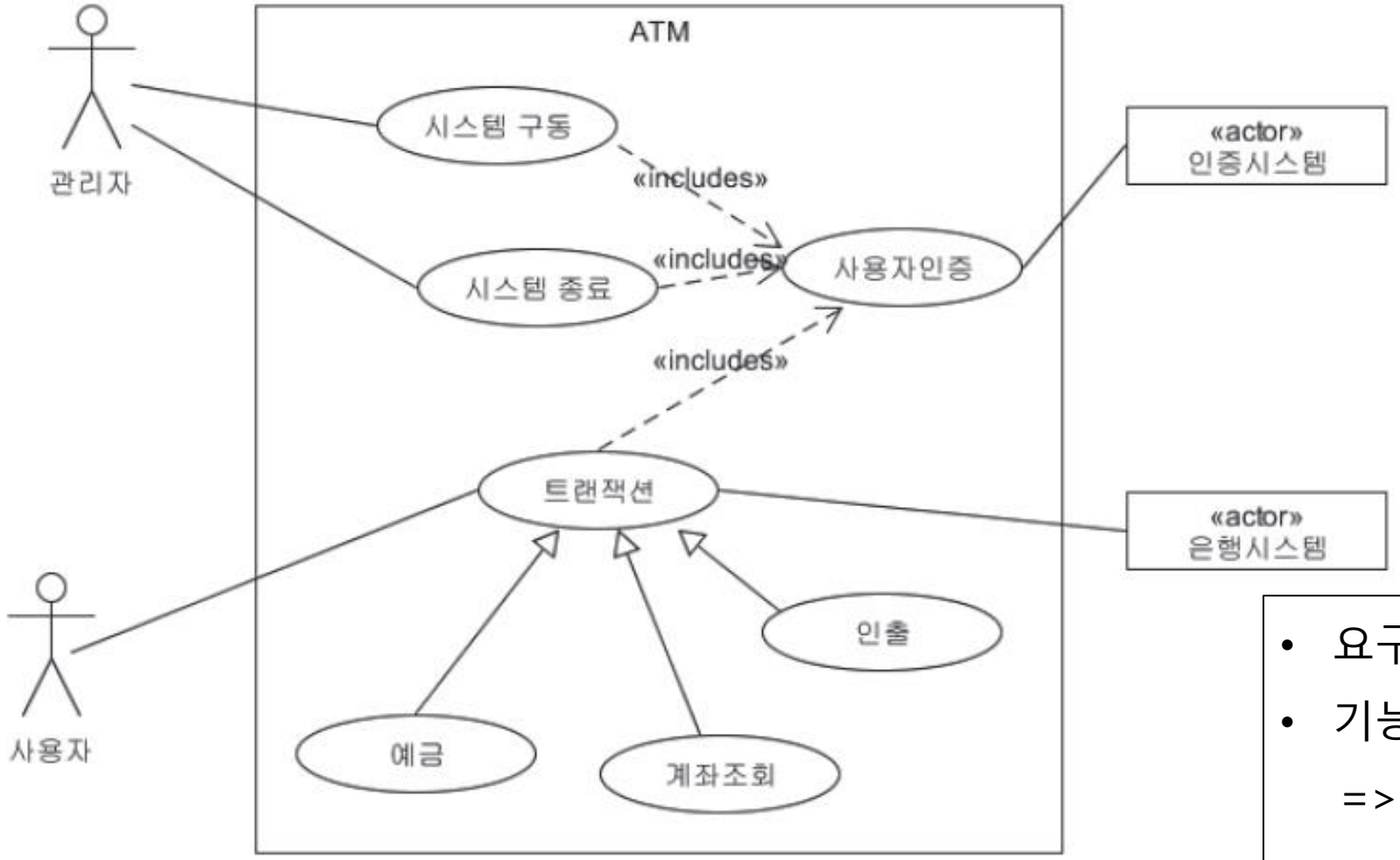


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

- 요구사항 정의서 각각의 기능
- 기능들에서 공통적인 부분이 있으면 분리

  => 모델링 과정(분석 과정)

  => 전체 시스템의 분석 및 설계가 용이

  => 각 기능의 재사용성도 향상

유스케이스 (UseCase)

유스케이스 모델을 이용해서 **기능 요구 사항 분석 모델링**
굉장히 많이 사용하는 모델

액터라고 하는 사용자 중심 요구사항 모델링
액터와 유스케이스 사이의 상호작용을 통해 시스템 분석
(비기능적 요구사항은 표현이 어려움)

액터와 시스템 사이에 어떤 인터렉션이 있는지 작성하는 것

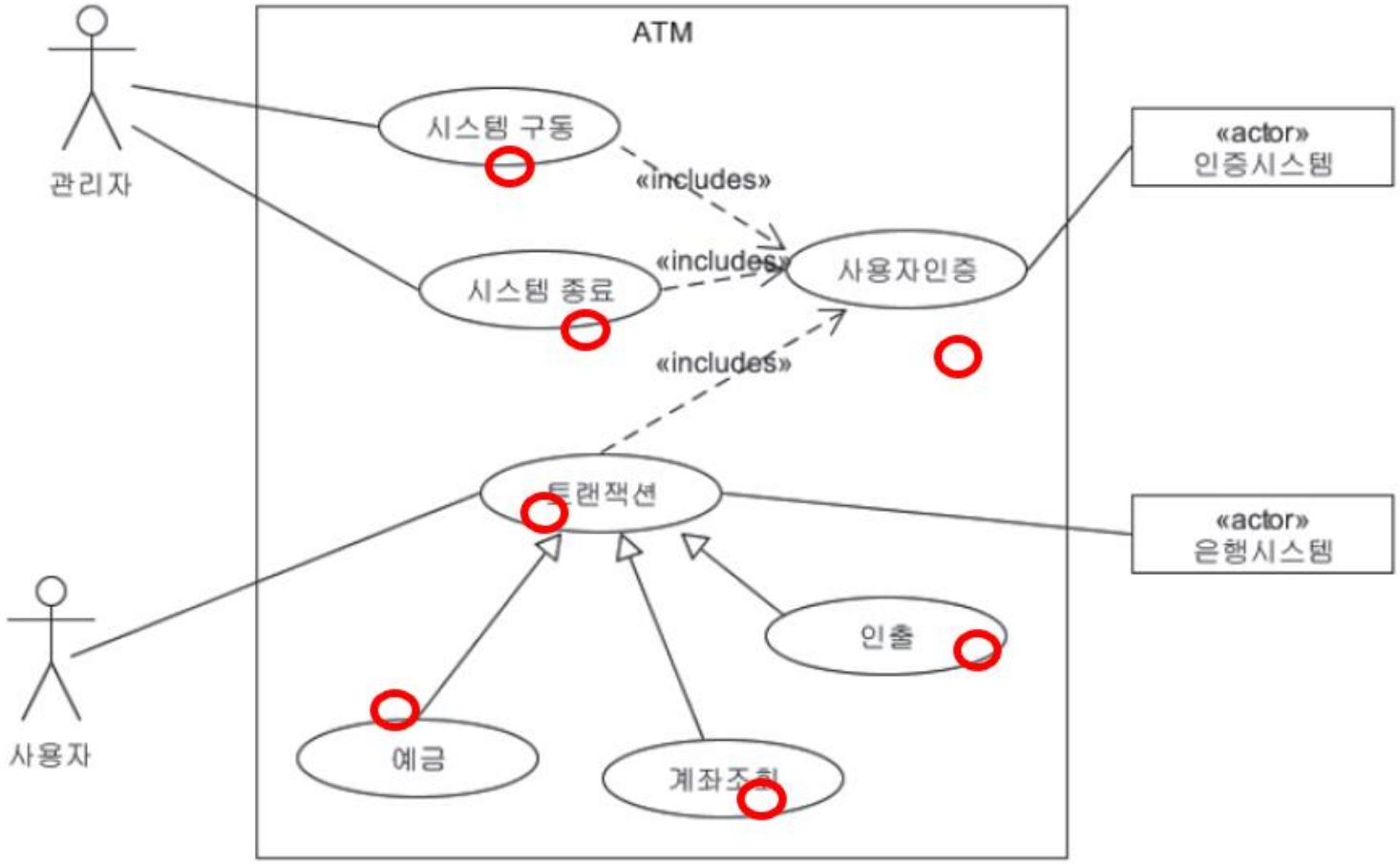


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

관계 (Relationship)

- 액터와 유스케이스 사이에 나타날 수 있으며, 유스케이스들 사이에서도 나타난다.
- 유스케이스간의 관계
  - 연관 관계 (Association)
  - 포함 관계 (Include)
  - 확장 관계 (Extend)
  - 일반화 관계 (Generalization)

많다고 좋은 건 아님

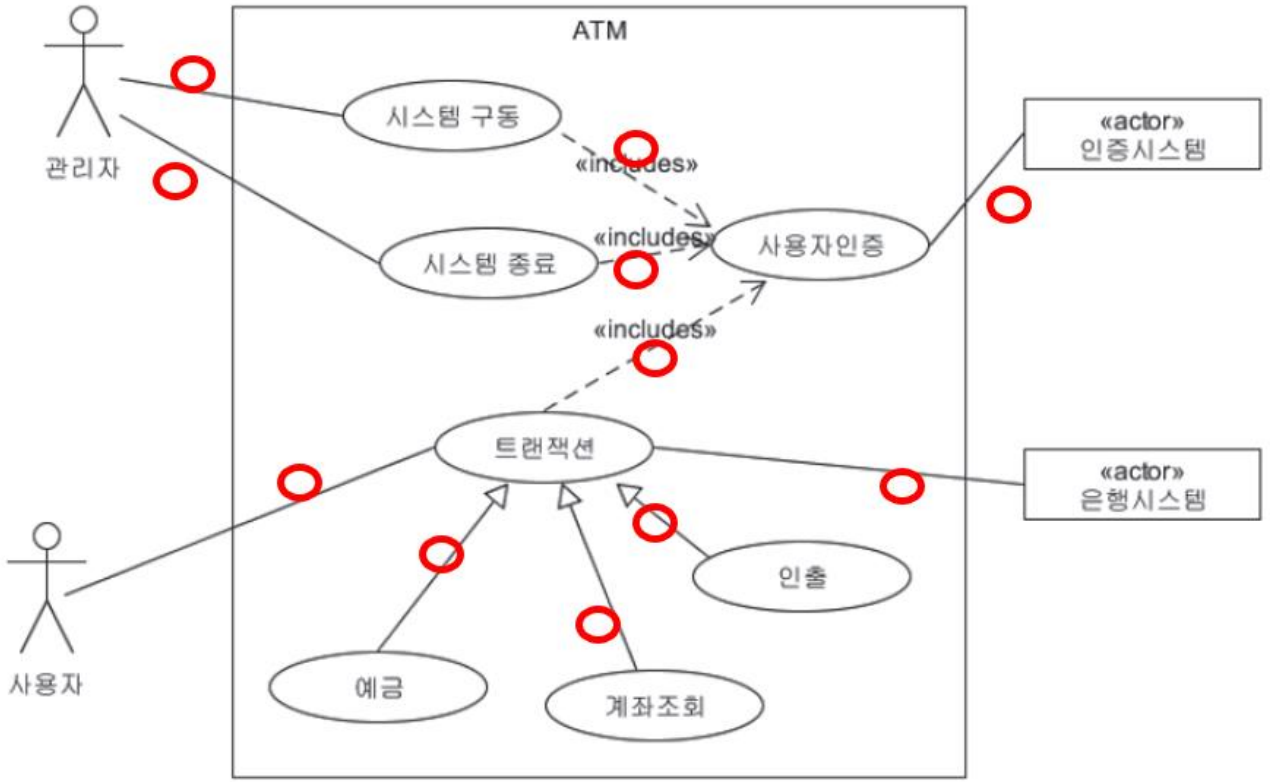


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

관계 (Relationship)

- **연관 관계 (Association)**

  - 엑터와 유스케이스 간의 상호작용이 있음을 표현해 주며, 유스케이스와 액터를 실선으로 연결

- **포함 관계 (Include)**

  - 필수적관계이며 여러 유스케이스에서 중복되는 단계가 있는 경우 이를 따로 떼어내어 새로운 유스케이스를 만든다. 원래의 유스케이스(들)에서 새롭게 만들어진 유스케이스 방향으로 (포함하는 쪽에서 포함되는 쪽으로) 화살표를 점선으로 연결하여 표현하고《includes》를 표기한다.

- **확장 관계 (Extend)**

  - 유스케이스가 특정한 조건이 만족되는 경우에만 실행되는 단계가 있는 경우에 확장관계를 사용하여 표기한다. 이를 위해 특정한 조건이 만족되는 경우에만 실행되는 단계를 따로 독립하여 유스케이스를 만들고, (포함 관계와 다르게) 확장 유스케이스에서 원래의 기본 유스케이스 방향으로 화살표를 점선으로 연결하고《extends》를 표기한다.

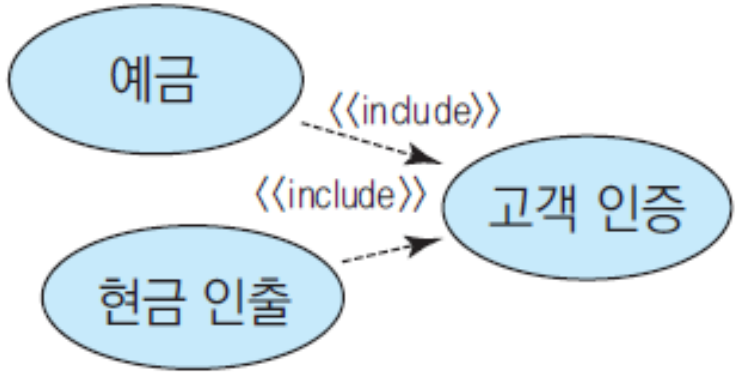
- **일반화 관계 (Generalization)**

  - 보편적인 유스케이스와 구체적인 유스케이스 사이에 존재

  - 해당 UseCase를 상속받는 관계

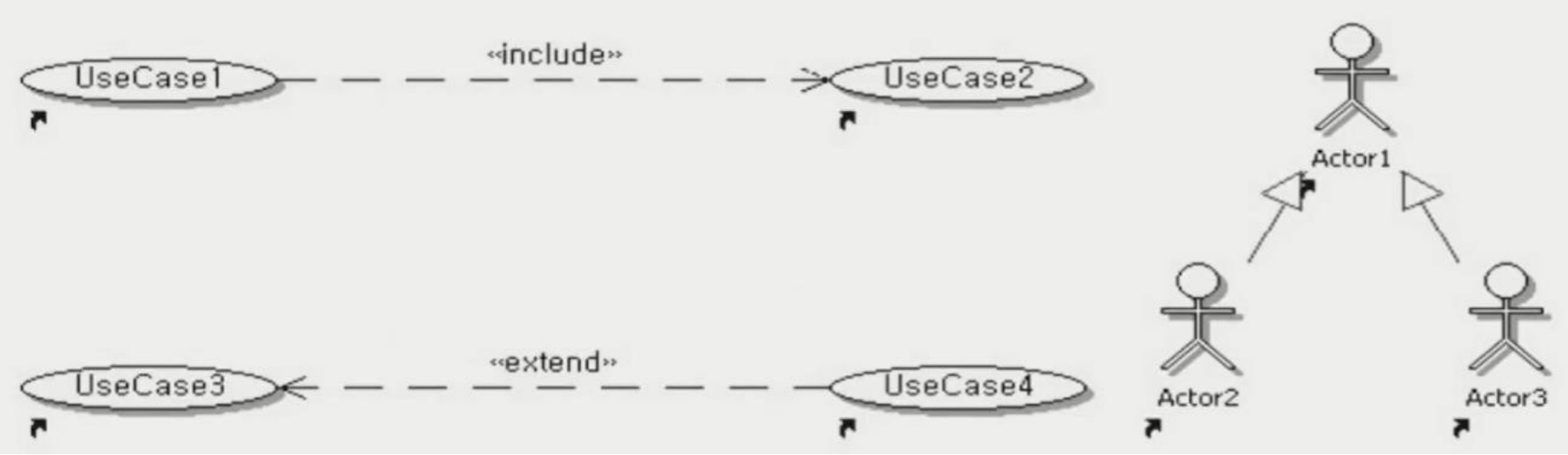
포함 관계 (Include)

- 어떤 유스케이스가 다른 유스케이스를 포함하는 관계(필수적 관계)
- 공통된 동작을 분리, 유스케이스 사이의 중복을 제거함
- UseCase1이 실행되면 UseCase2는 반드시 실행되는 것 (필수적)



〈예〉 포함 관계의 예

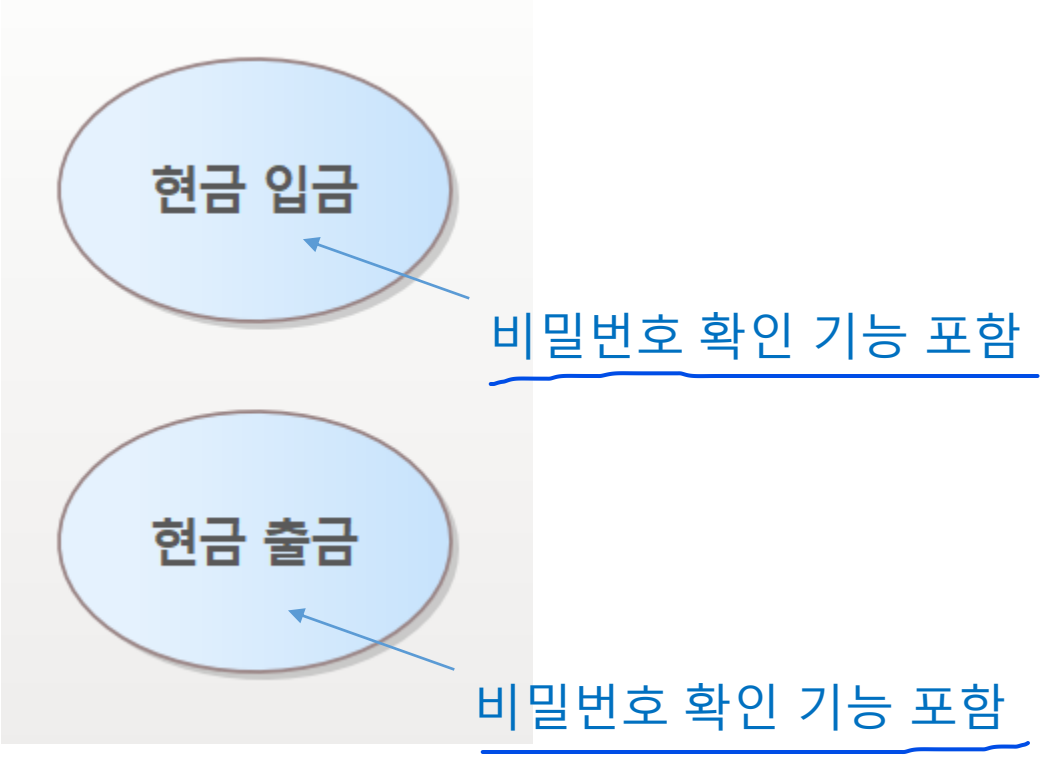
☞ 각 유스케이스에서 공통으로 사용하는 유스케이스는 별도의 유스케이스로 분할하여 공통 유스케이스로 모델링

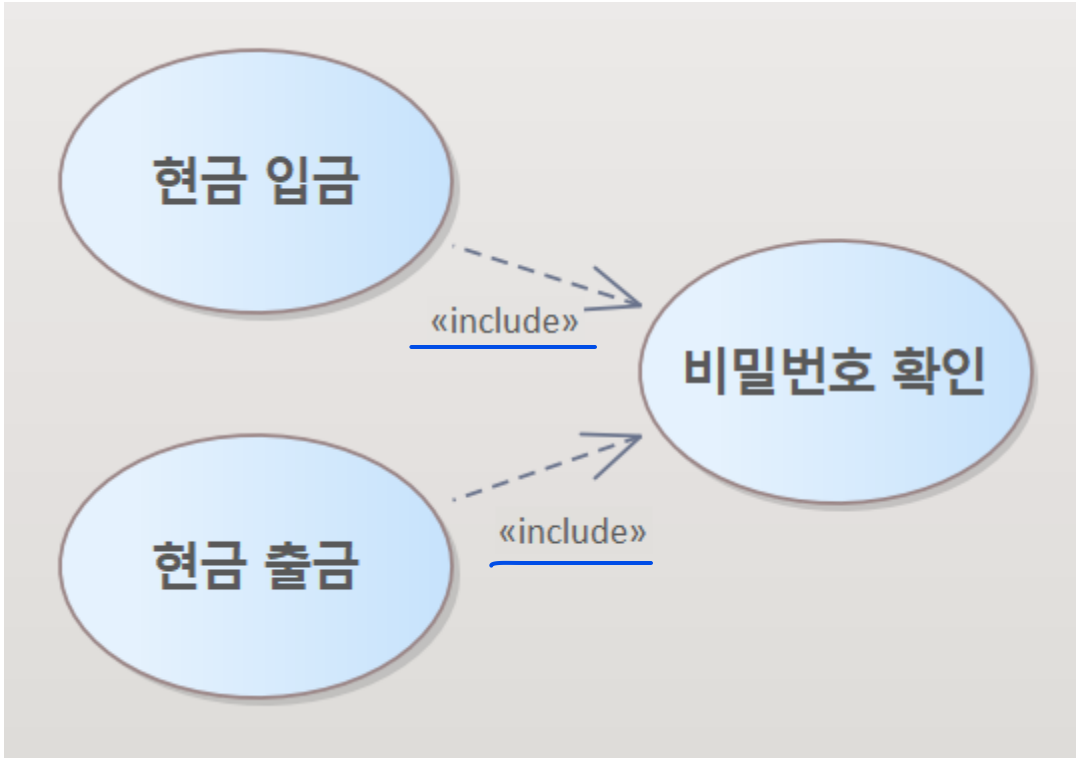


포함 관계 (Include)

☞ 사용 예) 공통된 동작을 분리함으로써 유스케이스 사이의 중복을 제거함

현금입금을 하려면 반드시 비밀번호 확인, 현금 출금을 하려면 //



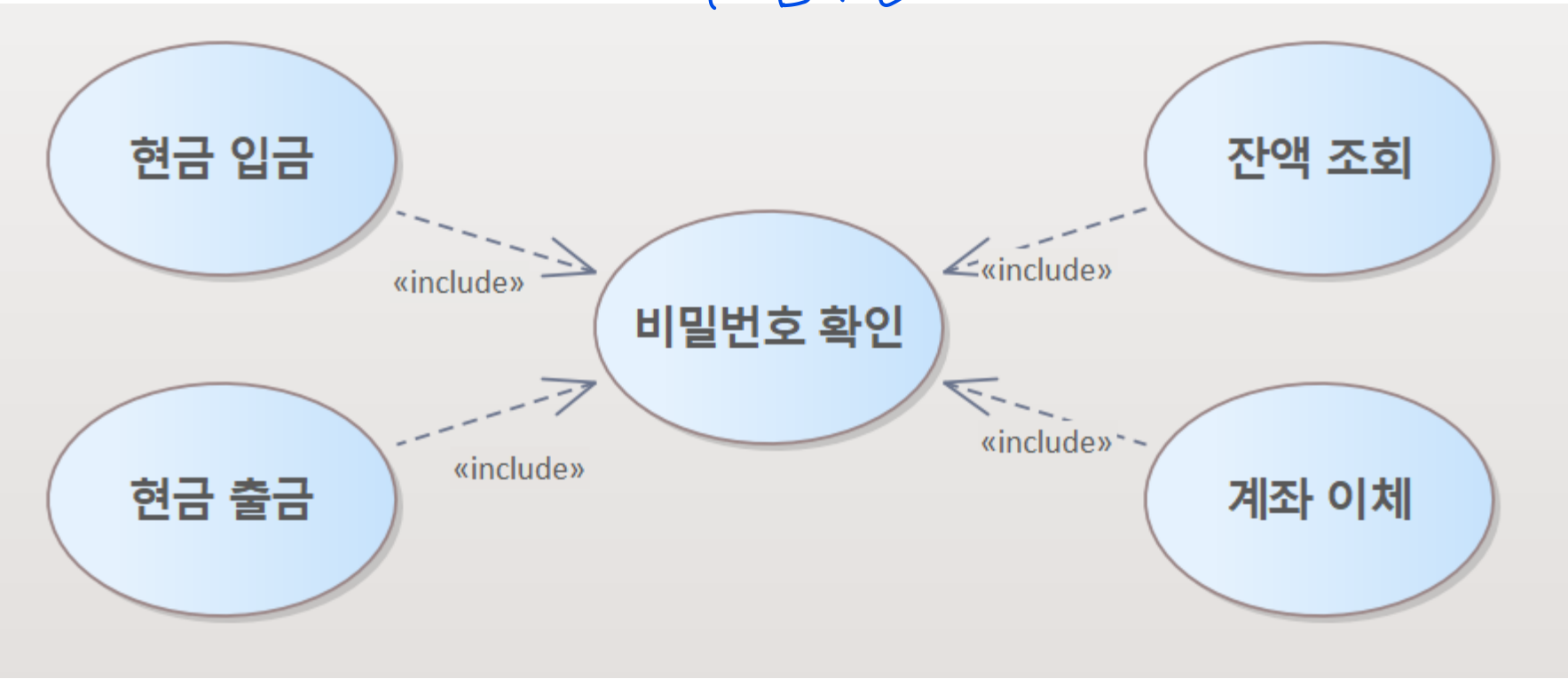


화살표를 이용해 일의 실행 순서를 기록하는 것이 아님, 기능 사이의 관계 표현(포함 · 확장, 필수 · 선택의 개념)

포함 관계 (Include)

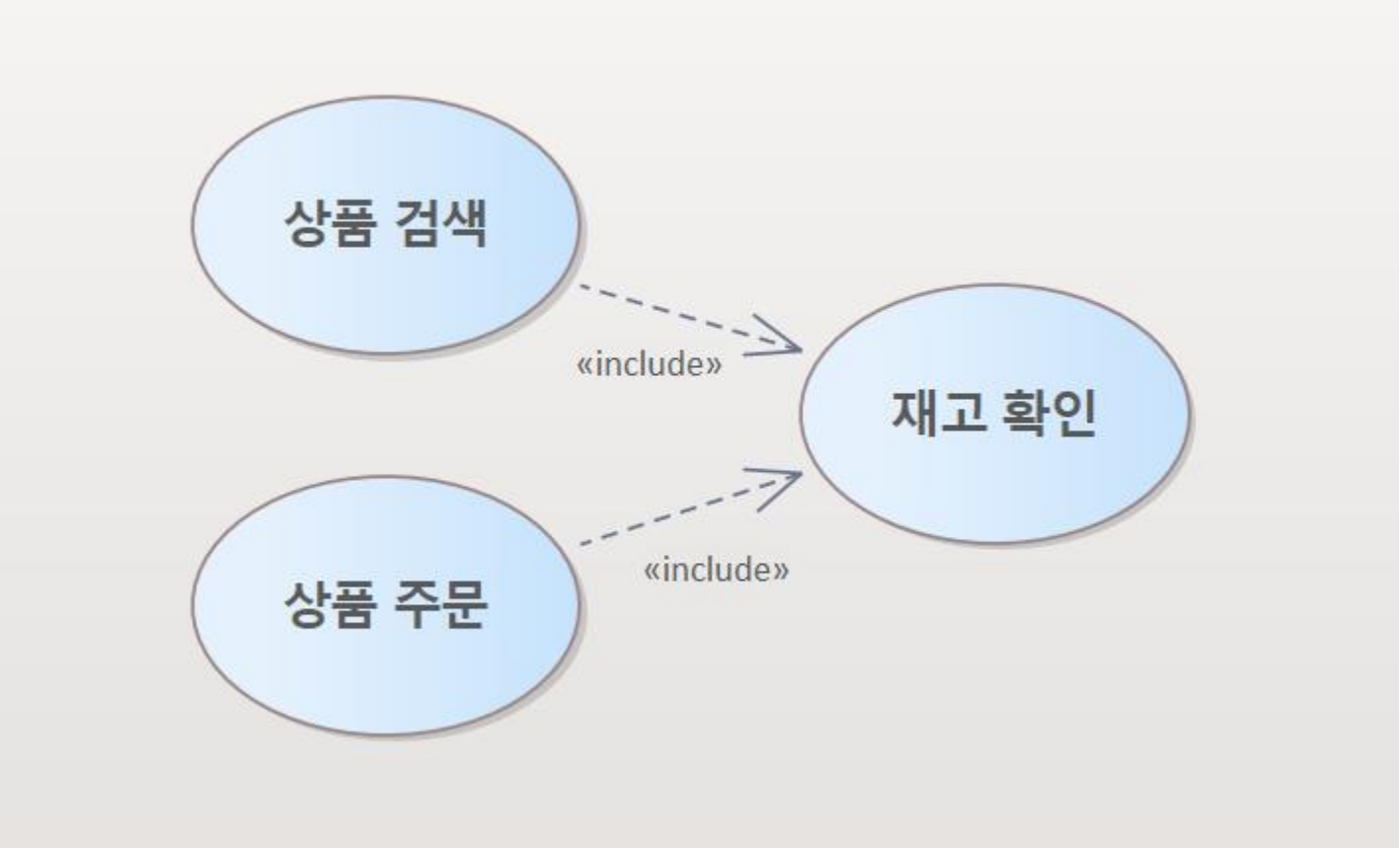
☞ 필요할 때마다 여러 다른 유스케이스에서 공유해서 사용

마치 함수같은 것



포함 관계 (Include)

예) 쇼핑몰 시스템 유스케이스 다이어그램 일부분

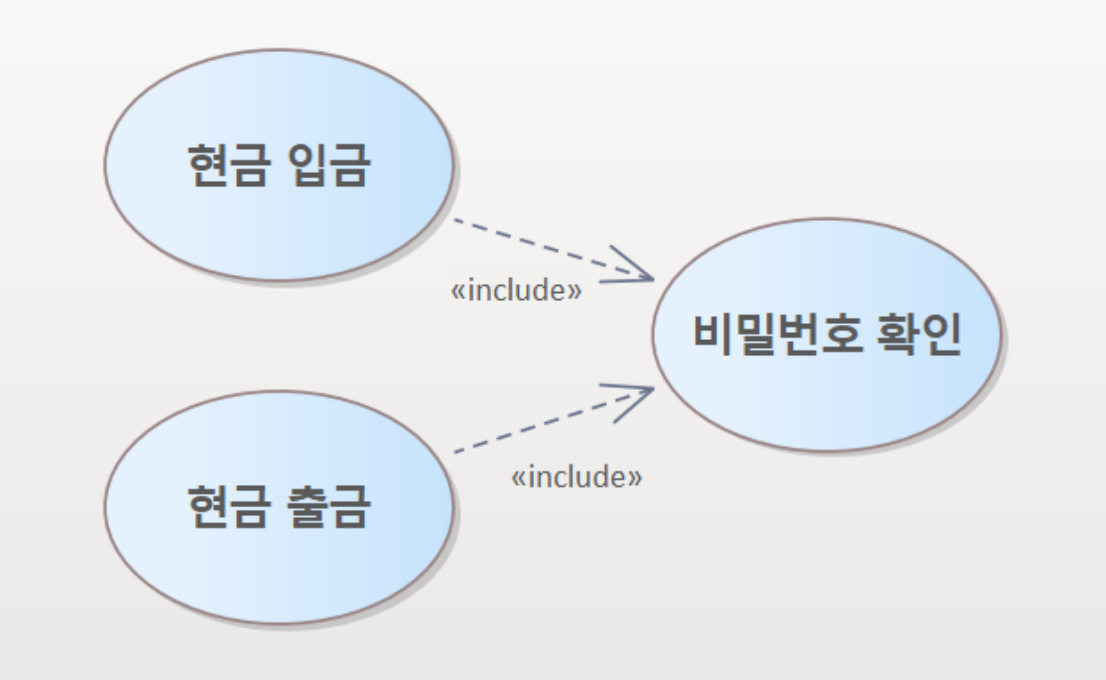


소프트웨어공학

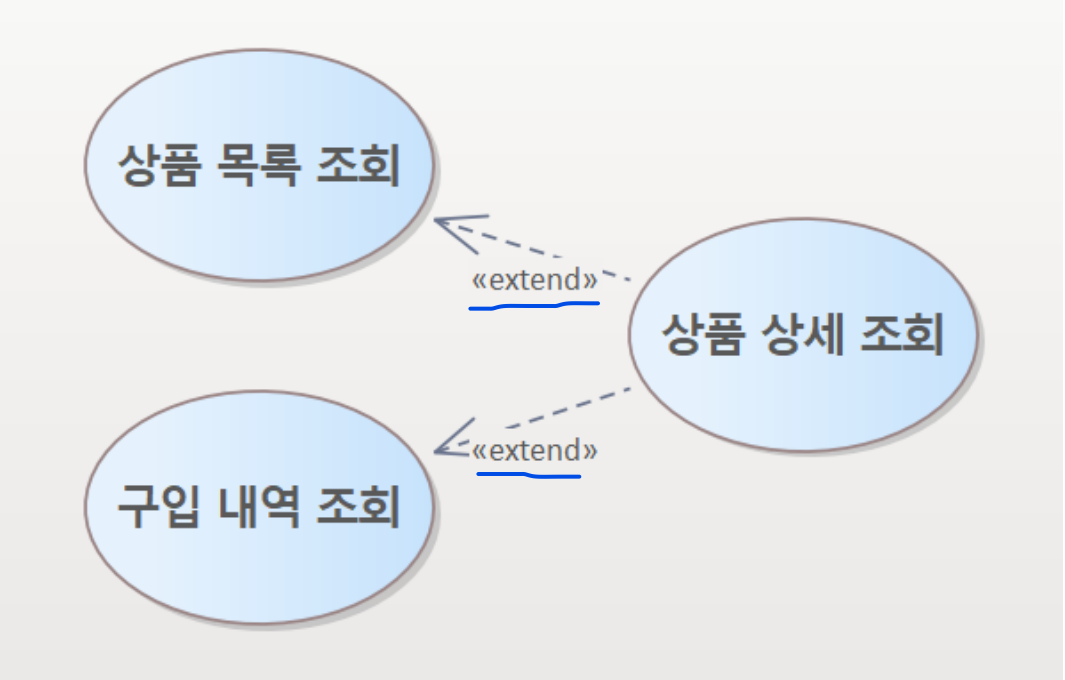
UML 및 UseCase 다이어그램 PART 3

확장 관계 (Extend)

- 포함 관계와 확장 관계의 비교
  - 모두 기본 유스케이스로부터 일부 행위를 추출하여 추가 유스케이스를 만드는 것



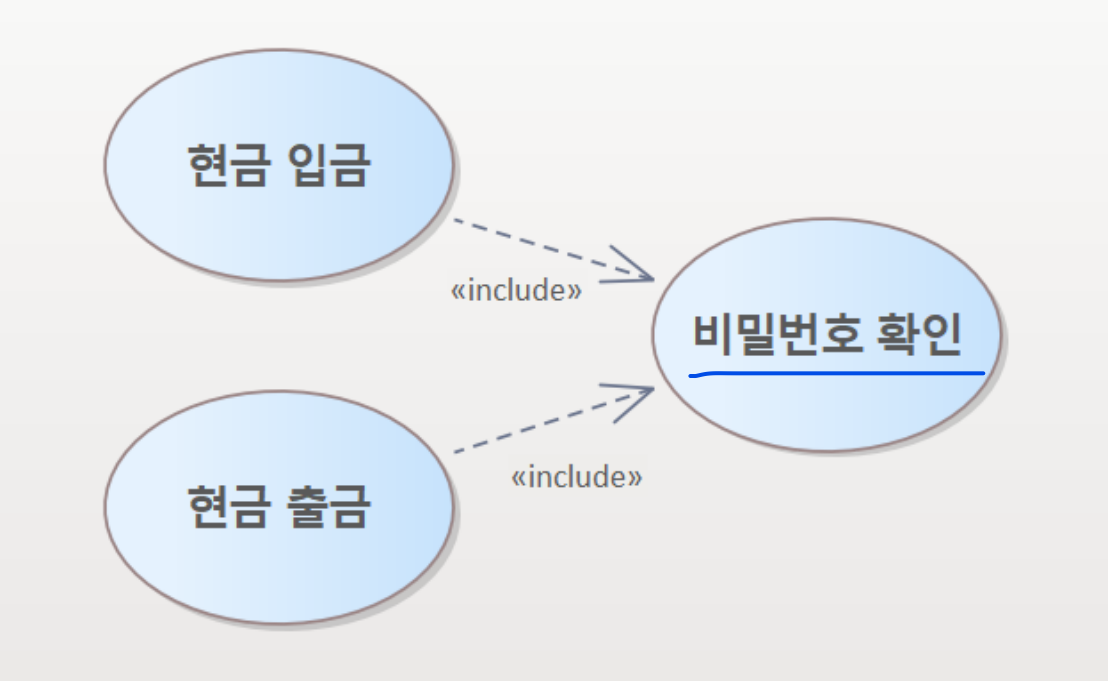
포함 관계(include)



확장 관계(extend)

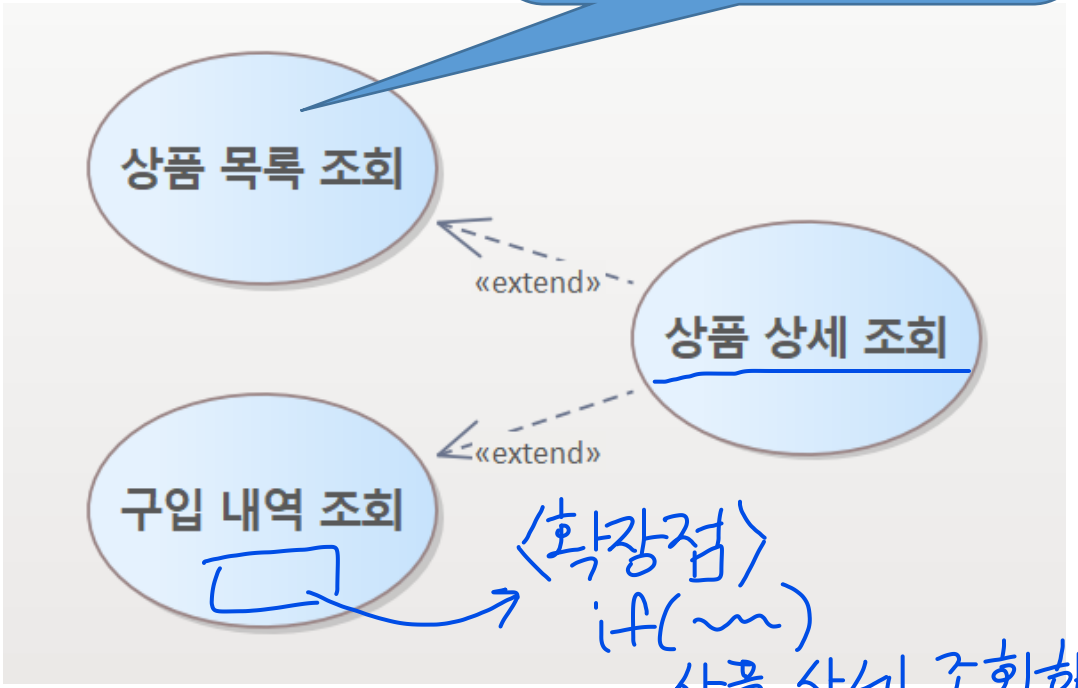
확장 관계 (Extend)

- 포함 관계와 확장 관계의 비교
  - 모두 기본 유스케이스로부터 일부 행위를 추출하여 추가 유스케이스를 만



포함 관계(include)

반드시 실행되어야 하는 세부 기능을 분리
(필수적 관계)

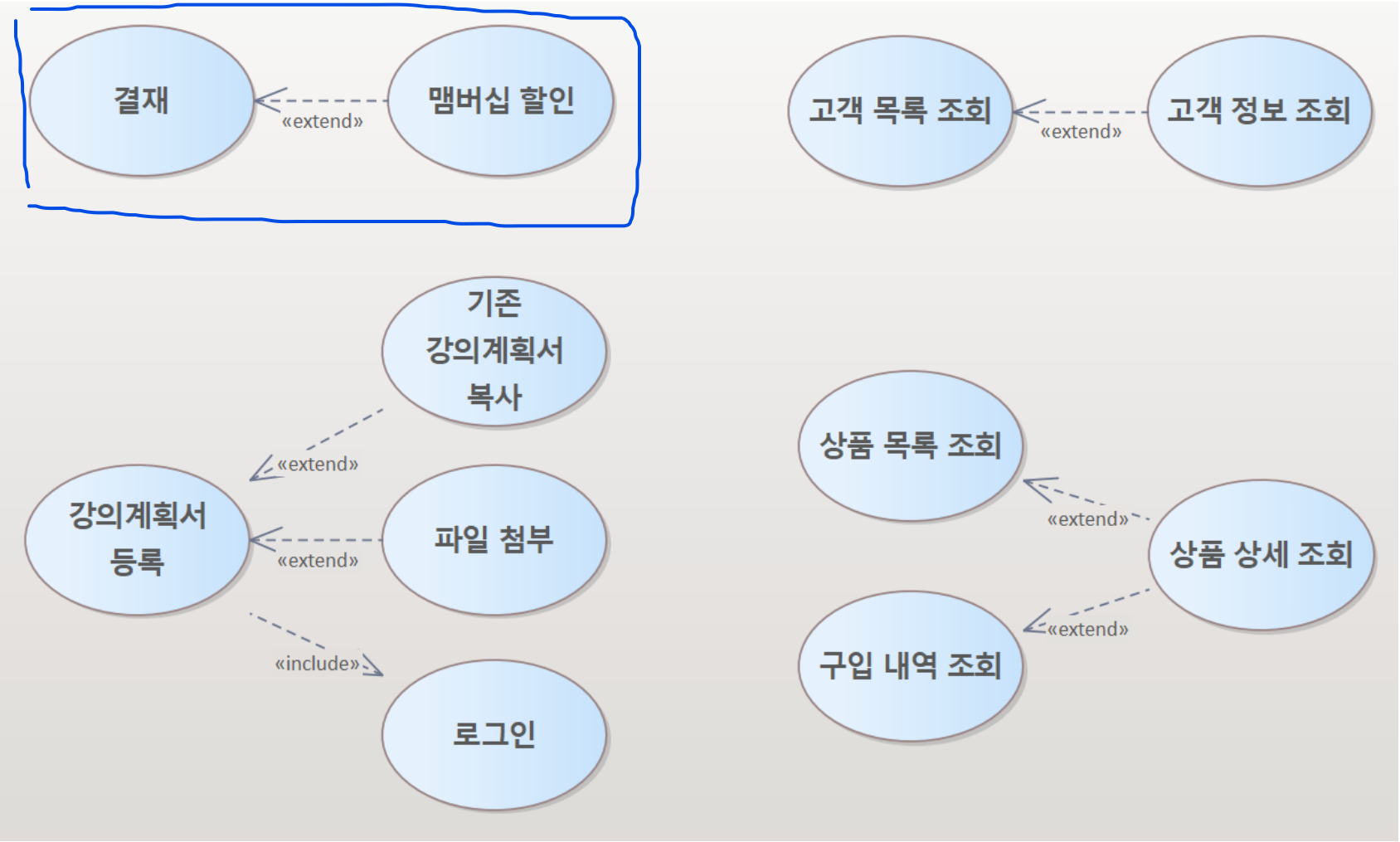


확장 관계(extend)

조건에 맞을 때만 선택적으로 실행되어지는 세부 기능을 분리
(선택적 관계)

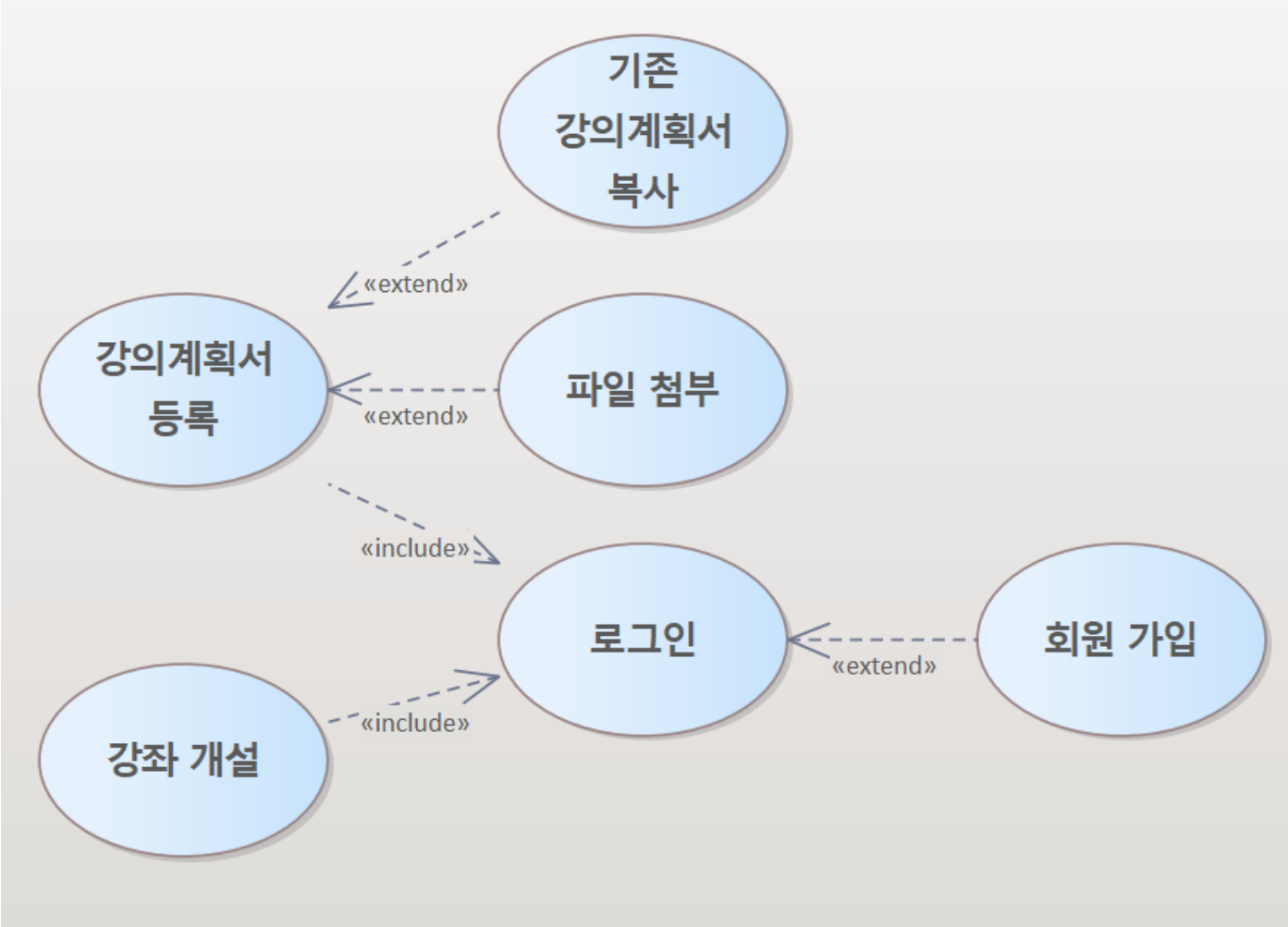
확장 관계 (Extend)

- 확장 관계의 다양한 예



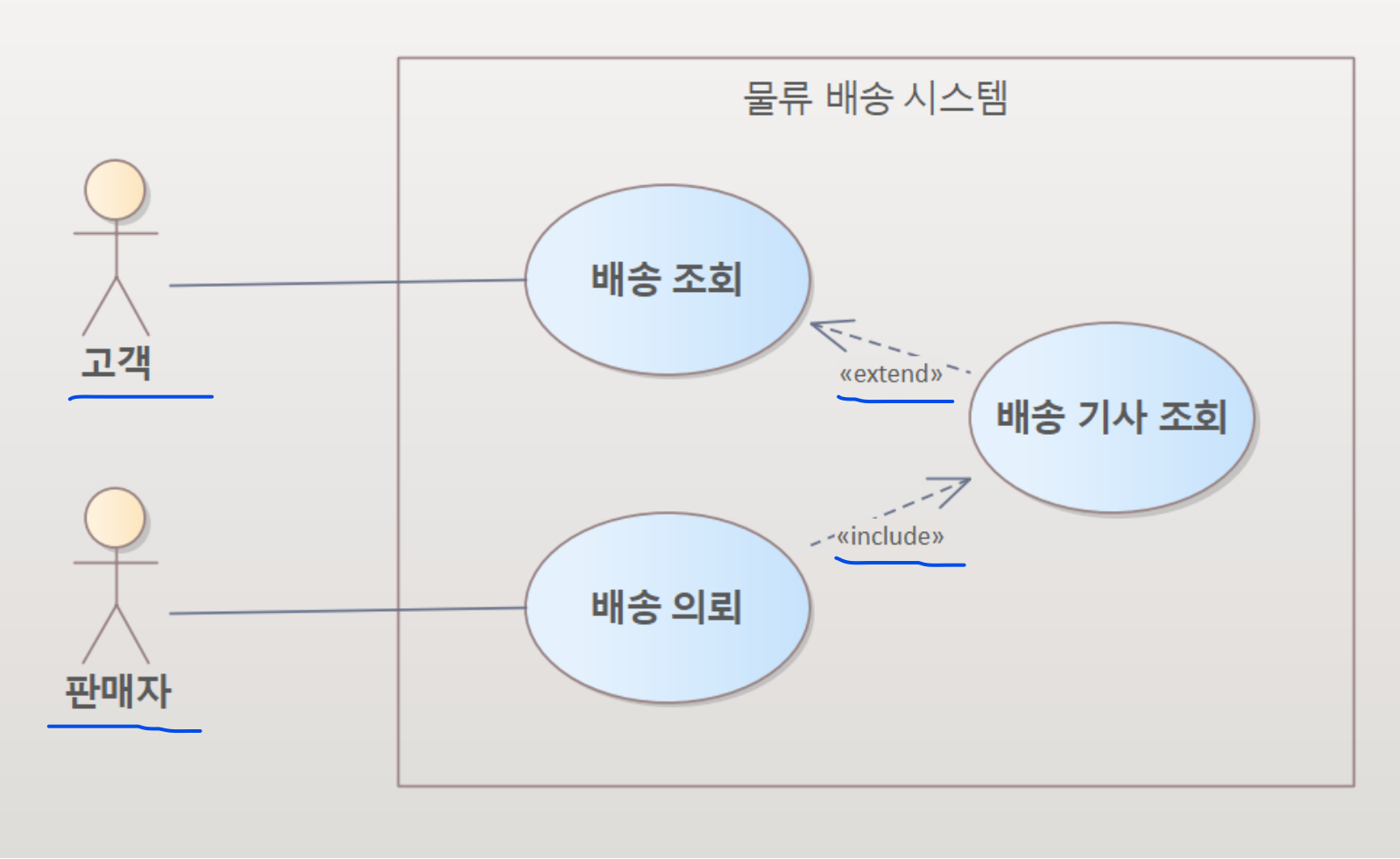
확장 관계 (Extend)

- 포함 관계와 확장 관계의 다양한 조합



확장 관계 (Extend)

- 포함 관계와 확장 관계의 다양한 조합



정리

- **포함 관계** : 기본 유스케이스를 실행하면서 반드시 실행되어야 하는 유스케이스(필수적 관계, 포함 관계)
  - 예) (함수라면) 반드시 호출되어야 하는 함수
  - 공통된 부분을 분리

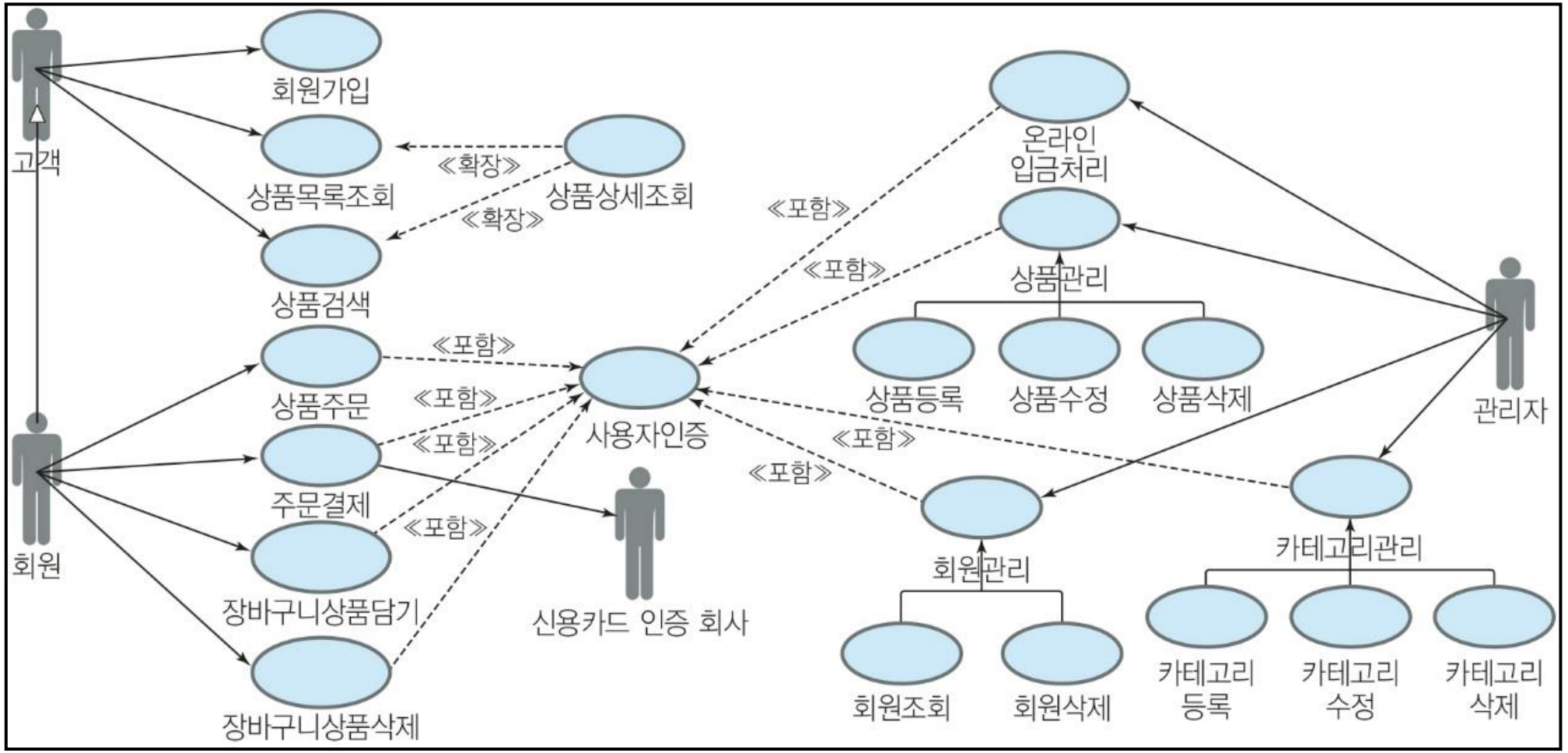
- **확장 관계** : 기본 유스케이스를 실행하면서 선택적으로 실행되는 유스케이스(선택적 관계, 확장 관계)
  - 예) 선택적으로 호출될 수 있는 함수
  - (확장할 때의 조건을 확장점이라고 하고 기본 유스케이스에 표시)
  - 선택적으로 실행되는 부분들을 분리

정리

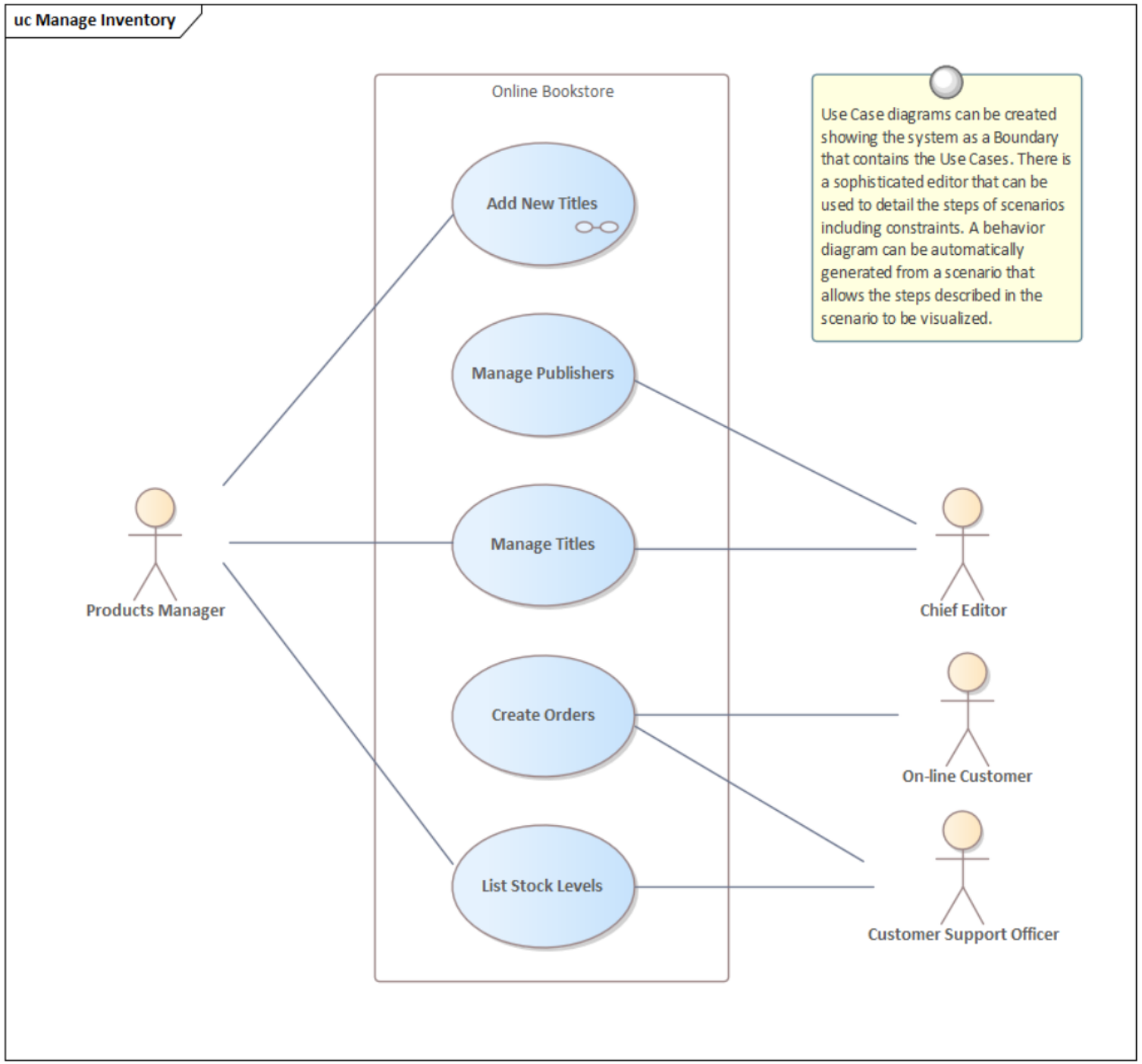
- 기존 유스케이스로부터 일부 행위를 추출하여 추가 유스케이스를 만드는 것
- (시스템의 행위나 기능에 영향을 미치지 않고도 중복되는 사례를 감소시켜)

  유스케이스들에 대한 이해 및 관리(유지보수, 기능 추가 및 변경 등) 용이
- 전체적인 프로젝트 구조를 단순하고 효율적으로 구성

예시: 인터넷쇼핑몰 유스케이스 다이어그램



예시: Online Bookstore



예시

Medical Appointment System

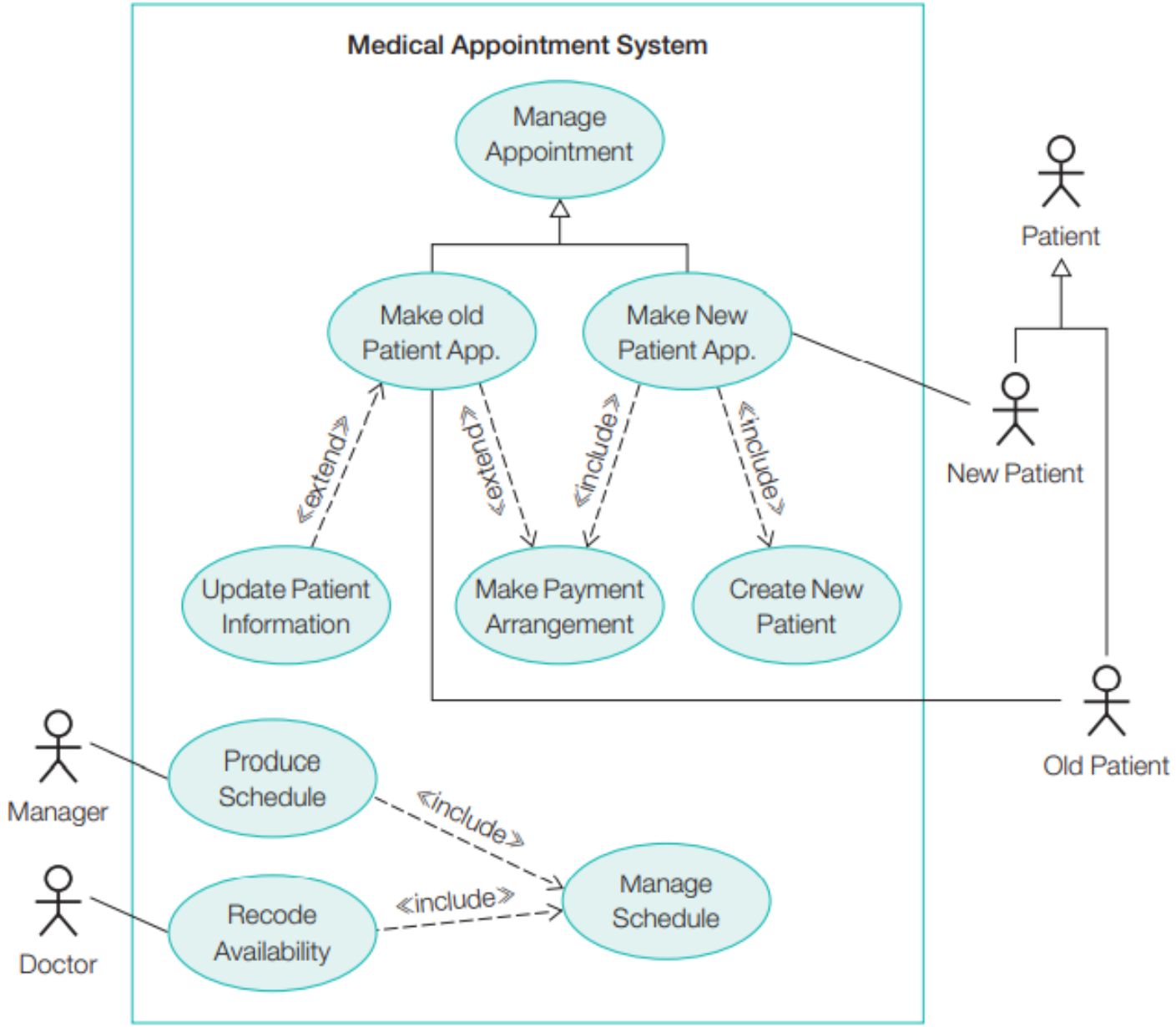
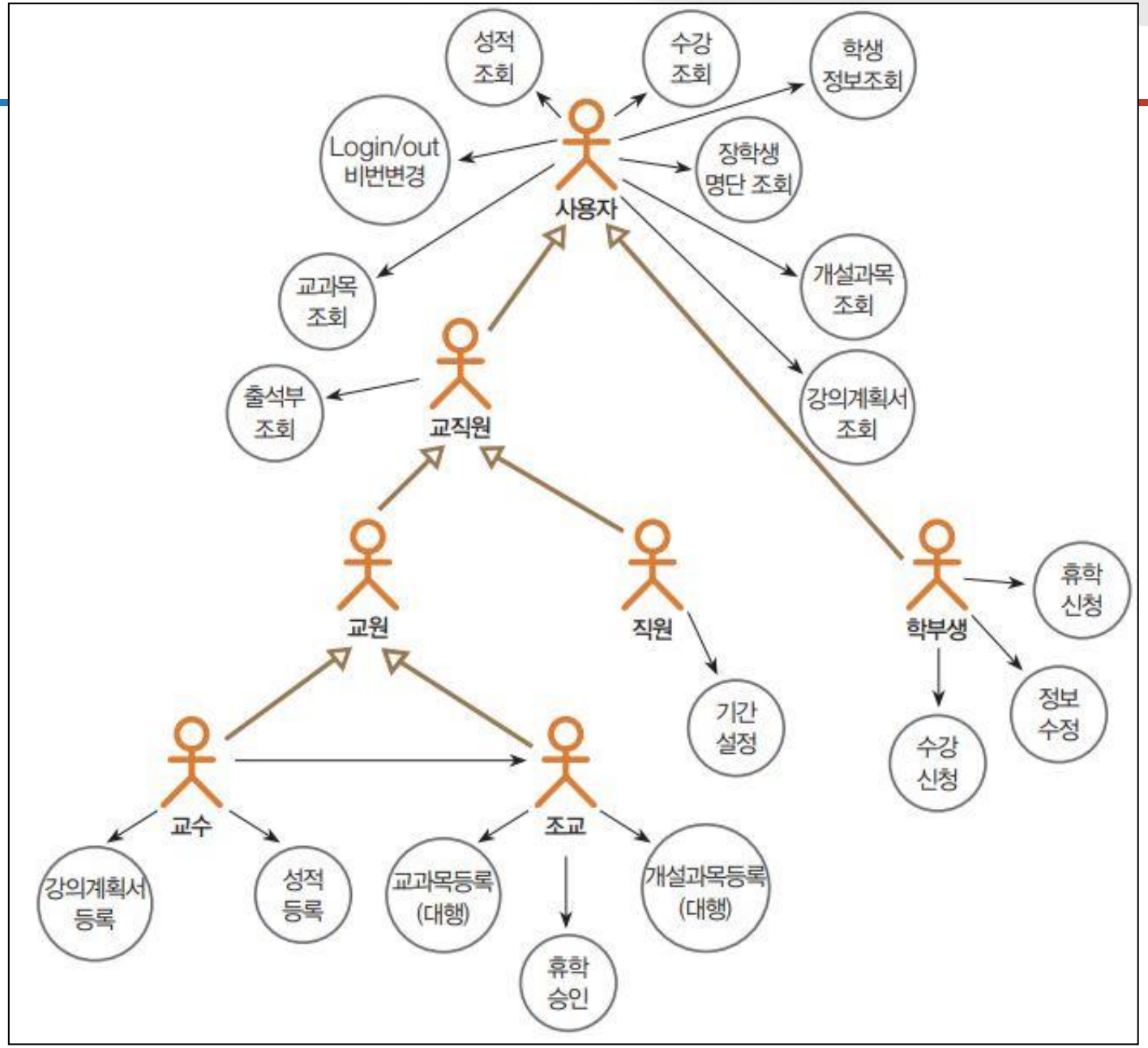


그림 8-3 진료 예약 시스템의 유스 케이스 다이어그램 예

예시

학사 관리 시스템의 간단한 기능 정리



T h a n k y o u

TECHNOLOGY

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Velit ex
ullicabo ipsum, labore sed tempora ratione asperiores des
cipserat bore sed tempora rati jgert one bore sed tem!