

Korea  
Software  
Technology  
Association



## UML2.x 기초 다루기

**훈련기간: 2010.01.25 ~ 02.05**

**강사명: 손재현** -넥스트트리소프트  
-jhsohn@nexttree.co.kr

## □ 교육 목표 & 특징

- UML2.x의 이해
- 유스케이스 작성
- 객체모델링 이해
- UML2.x의 다양한 다이어그램 이해 및 활용
- 모델링 도구 사용법 습득

- 본 강의는 아래 기술에 대한 이해를 필요로 합니다.
  - 객체지향 언어(Java) 기초
  - 개발프로세스 이해

□ 교육은 매 회 4 시간씩 총 5회에 걸쳐 진행합니다.

1 일차	2 일차	3 일차	4 일차	5 일차
<ul style="list-style-type: none"> <li>- UML 개요</li> <li>UML 소개</li> <li>UML 역사</li> <li>UML 다이어그램분류</li> </ul>	<ul style="list-style-type: none"> <li>- 구조 다이어그램</li> <li>클래스</li> <li>객체</li> <li>컴포넌트</li> <li>배치</li> </ul>	<ul style="list-style-type: none"> <li>- 행위 다이어그램</li> <li>유스케이스</li> <li>액티비티</li> <li>상태기계</li> </ul>	<ul style="list-style-type: none"> <li>- 상호작용 다이어그램</li> <li>상호작용 Overview</li> <li>시퀀스</li> <li>커뮤니케이션</li> <li>타이밍</li> </ul>	<ul style="list-style-type: none"> <li>- 유스케이스 I</li> <li>유스케이스 개요</li> <li>유스케이스 내용</li> <li>유스케이스 다이어그램</li> </ul>
6 일차	7 일차	8 일차	9 일차	10 일차
<ul style="list-style-type: none"> <li>- 유스케이스 II</li> <li>유스케이스 목표수준</li> <li>유스케이스 명세</li> <li>유스케이스 패턴</li> </ul>	<ul style="list-style-type: none"> <li>- 유스케이스 III</li> <li>유스케이스 분석기법</li> <li>분석클래스</li> <li>제어클래스</li> <li>실체클래스</li> </ul>	<ul style="list-style-type: none"> <li>- 요구사항 모델실습 I</li> <li>유스케이스</li> <li>사용자 시나리오</li> <li>핵심개념 모델</li> </ul>	<ul style="list-style-type: none"> <li>- 요구사항 모델실습 II</li> <li>인터페이스 추출</li> <li>유스케이스 분석</li> <li>컴포넌트 식별</li> </ul>	<ul style="list-style-type: none"> <li>- 설계모델 실습</li> <li>컴포넌트 설계</li> <li>유스케이스 설계</li> <li>도메인 모델</li> </ul>

# 5월차 – 유스케이스

1. 유스케이스 개요
2. 유스케이스 다이어그램

ONE STEP AHEAD

# 1. 유스케이스 개요

- ☐ 소프트웨어 개발과정
- ☐ 사용자 요구사항
- ☐ 유스케이스 개요
- ☐ 유스케이스와 시나리오
- ☐ 유스케이스 내용

ONE STEP AHEAD

- **요구사항 정의(Requirements) <- Our Focus**
  - 기능적, 비 기능적 요구사항의 집합으로 시스템 비전을 기술
- **분석/설계(Analysis & Design)**
  - 구현 단계에서 시스템이 어떻게 실체화 될지를 기술
- **구현(Implementation)**
  - 실행 가능한 시스템이 될 코드를 생성
- **테스트(Test)**
  - 전체 시스템을 검증
- **배포(Deployment)**
  - 시스템을 배포하고 사용자 교육

## □ 정의

- 시스템이 갖추어야 할 능력과 조건
- 유스케이스는 그 자체로 요구사항
  - 시스템이 해야 하는 것에 대해 상세하고 정확하게 표현
  - 시스템의 기능적인 요구사항
- 유스케이스가 요구사항의 전부는 아님
  - 외부 인터페이스, 데이터포맷, 비즈니스 규칙, 그리고 복잡한 공식과 같은 것은 자세히 다루지 않음
  - 요구사항 중 기능, 즉 행위관련 부분

## □ 요구사항 관리(Requirement Management)

- 최상의 실행 지침(Best Practice)
- 변화하는 요구사항을 찾아내고, 조직화, 문서화, 추적하기 위한 체계적인 접근 방법



## 사용자 요구사항(2/2)

### □ 요구사항 타입

- FURPS+

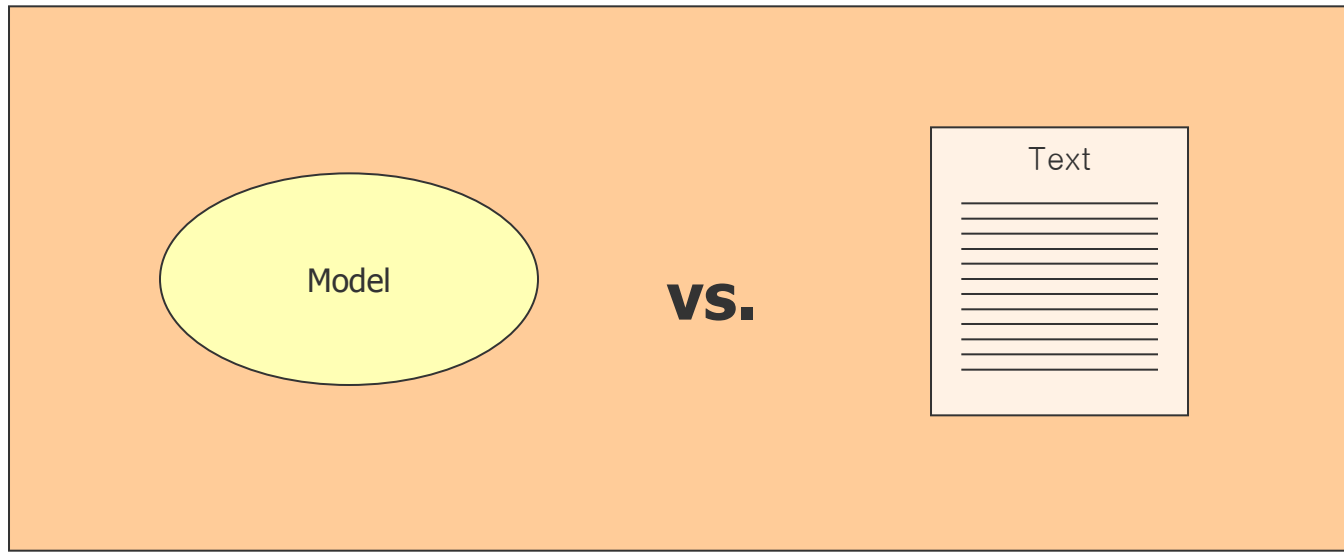
- 기능성(Functionality), 유용성(Usability), 신뢰성(Reliability), 성능(Performance), 지원성(Supportability)
- 구현(Implementation), 인터페이스(Interface), 운영(Operations), 패키징(Packaging), 법률요건(Legal)

- 기능적/비 기능적 요구사항

### □ 유스케이스

- 기능적인 요구사항 획득 및 서술

## □ 들어가기



유스케이스



## 유스케이스 개요(2/4)

### □ 역사

- 1960년대 후반 야콥슨(Ivar jacobson)에 의해 개념이 수립
- 1980년대 후반부터 요구사항 정립의 수단으로 각광을 받기 시작
- 1990년대 유스케이스가 UML에 핵심적인 개념으로 추가

### □ 소개

- 시나리오의 집합
  - 액터와 시스템간의 상호작용을 텍스트로 작성
  - 이해관계자들 중에서 일차 액터로 불리는 행위자의 요청에 대한 시스템의 응답
  - 플로우차트, 시퀀스차트, 프로그래밍 언어 등으로 작성될 수 있지만 기본적으로는 텍스트
- 대상 시스템의 범위를 정하는 것을 도움
- 고객과의 의사소통을 위한 도구



## 유스케이스 개요(3/4)

### □ 요구사항 관리

- 시스템의 기능적 요구사항(functional requirements) 획득 기법
- 사용자와 시스템간의 상호작용 서술
- 고객과의 의사소통 도구

### □ 시나리오(scenario)

- 사용자와 시스템 간의 상호작용을 서술하는 일련의 스텝들
- 발생 가능한 시퀀스 중 하나

### □ 유스케이스(use case)

- 공통적인 사용자 목표에 의해 묶인 시나리오들의 집합
- 기본적으로 텍스트
- 유스케이스의 핵심은 사용자 목표(user goal)

## 유스케이스 개요(4/4)

### □ 액터(actor)

- 사용자가 시스템에 대해서 수행하는 역할(role)
- 여러 사람이 하나의 액터로, 하나의 사람이 여러 액터로 맵핑 가능
- 액터가 반드시 사람일 필요는 없음

### □ UML에서의 정의는 미약

- 유스케이스 다이어그램 표기법만을 정의

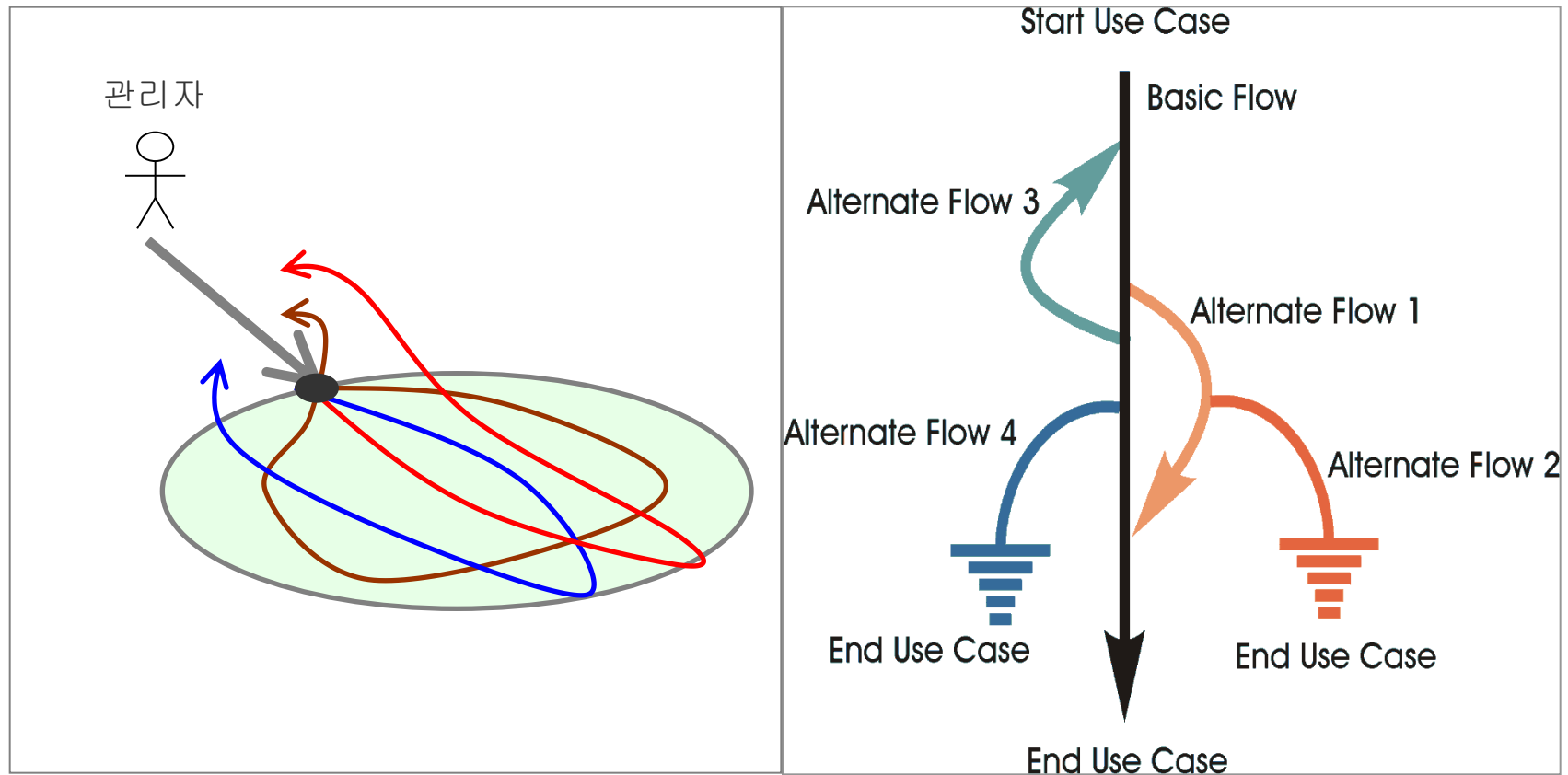
### □ 유스케이스의 가치

- 다이어그램이 아닌 내용(Content)

### □ 활용

- 요구사항을 도출하는 수단
- 기능적 요구사항 획득
- 근래의 객체지향 방법론이나 컴포넌트 기반 개발 방법론들은 유스케이스를 적극 사용

- 하나의 유스케이스는 여러 개의 가능한 시나리오를 내포함
- 유스케이스의 결과 성공일 수도 있고 실패일 수도 있음



# 유스케이스 내용 - 주 성공 시나리오

## □ 개요

- 기본 흐름은 모든 것이 제대로 작동하는 것처럼 작성: 주 성공 시나리오
- 기본 흐름은 분기나 대안 흐름이 없는 단순한 형태의 문장
- 스텝 작성 시 유의사항
  - 누가 스텝을 수행하는지 명확하게 표시
  - 사용자 인터페이스를 서술해서는 안됨
  - 액터가 수행하는 절차가 아니라 액터의 의도(intention)를 표시

### Main Success Scenarios

1. 유스케이스는 고객이 주문하기를 선택하면 시작된다.
2. 고객은 이름과 주소를 입력한다.
3. 고객이 제품 코드를 입력하는 동안
  - a) 시스템은 제품 설명과 가격을 제공한다.
  - b) 시스템은 합계에 아이템의 가격을 추가한다.

### End loop

4. 고객이 신용카드 지불정보를 입력한다.
5. 고객이 보내기를 선택한다.
6. 시스템이 그 정보를 확인하고, 주문을 저장하고, 지불 정보를 회계 시스템에 전달한다.
7. 지불이 확인되면, 주문은 확정된 것으로 표시되고, 주문 번호가 고객에게 전달되고 유스케이스는 종료된다.



# 유스케이스 내용 - 확장(Extensions)

## □ 개요

- 기본 흐름에 대한 대안을 제시
- 실패, 예외, 에러도 확장 흐름으로 문서화
- 시스템이 서로 다른 행위를 하게 하는 조건
- 확장 흐름은 언제든지 일어날 수 있는 일을 보여줌
  - 정상적인 이벤트의 흐름을 방해하는 것
- 주 성공 시나리오의 변동(variants)
- 확장은 성공할 수도 있고 실패할 수도 있음

### Extensions.

1\*. 보내기를 선택하기 전 언제든지, 고객은 취소를 선택할 수 있다. 주문은 저장되지 않고 유스케이스는 종료된다.

6a. 정보가 올바르지 않을 경우:

6a1. 시스템은 고객에게 정보를 수정하도록 한다.

7a. 지불이 확인되지 않을 경우:

7a1. 시스템은 고객에게 지불 정보를 수정하거나 취소하도록 한다.

.1 고객이 정보를 수정하면, 기본 흐름의 단계 4로 돌아간다.

.2 고객이 취소를 선택하면, 유스케이스는 종료된다.

## 유스케이스 내용 - 액터(Actor)

### □ 액터란 무엇인가?

- 이벤트를 완결하기 위해 시스템과 상호 작용하는 개체
- 액터가 사람일 경우, 시스템과 상호 작용하는 사용자에게 의해 수행되는 역할(Role)을 나타냄
- 시스템과 상호작용하고, 역할을 수행을 위하여 실재하는 외부 개체

### □ 액터를 정의해야 하는 이유?

- 액터가 수행하는 역할은 유스케이스가 필요한 이유와 결과에 대한 관점을 제공
- 액터에 초점을 맞추으로써, 시스템이 어떻게 구현될 지가 아니라 시스템이 어떻게 사용될 지에 집중
- 유스케이스 모델링에 포함될 필요가 있는 잠재적인 사용자 식별을 함

## 유스케이스 내용 - 일차액터

- 시스템의 서비스 중에 하나를 요청하는 이해관계자
- 시스템의 관점에서 볼 때, 시스템의 운용을 통해서 달성할 수 있는 하나의 목표를 가진 자
- 일차 액터가 유스케이스를 시작하지 않는 두 가지 경우
  - 궁극적 일차액터에 의한 시작
    - 예 : 서비스요청을 하는 고객의 요청을 실행하는 전화 교환원
    - 일차 액터 : 고객
  - 시간에 의한 시작
    - 예 : 매일밤 자정에 실행해야 하는 유스케이스
    - 일차 액터 : 해당 유스케이스에 관심을 갖는 이해관계자

## 유스케이스 내용 – 지원액터(Secondary Actor)

- 유스케이스를 수행하는 동안 시스템과 통신하는 다른 액터
- 시스템에 서비스를 제공하는 외부 액터
  - 예 : 고속프린터, 웹서비스, 사람
- 시스템이 사용할 외부 시스템과 인터페이스 간의 프로토콜 식별에 도움
- 다른 유스케이스의 일차 액터가, 또 다른 유스케이스에서는 지원액터가 될 수 있음
  
- 액터 실습 - 다음 중 액터는?
  - 현금 인출기(ATM), 고객, 현금 인출카드, 현금 인출기 화면, 은행 소유주, 서비스 담당직원, 프린터, 은행의 주 컴퓨터 시스템, 은행강도

## 유스케이스 내용 – 액터 찾기(1/2)

### □ 액터 찾기

- 시스템과 상호작용을 수행하는 외부 개체를 찾음
- 시스템과 상호 작용하는 Stakeholder 분석
- 요구사항 워크숍 초안
- 현재 프로세스와 시스템에 대한 사용자 매뉴얼과 훈련 지침서
- 가이드와 매뉴얼은 종종 잠재적인 액터의 역할을 직접적으로 나타냄
- 사용자와의 미팅 시 작성된 메모는 잠재적인 액터일 수 있는 사용자 식별을 도움

## 유스케이스 내용 - 액터 찾기(2/2)

### □ 액터를 찾기 위한 질문

- 특정 요구사항에 흥미를 가지는 사람은 누구인가?
- 조직 내부의 어디에서 시스템이 사용되는가?
- 시스템을 사용함으로 이득을 얻는 사람은 누구인가?
- 누가 시스템에 정보를 제공하고 정보를 사용하고 제거하는가?
- 누가(또는 무엇이) 시스템과의 이벤트를 시작하는가?
- 누가(또는 무엇이) 시스템이 이벤트에 응답하는 것을 돕기 위해 시스템과 상호 작용하는가?
- 누가 시스템을 유지보수 하는가?
- 시스템이 기존 시스템(legacy system)과 상호 작용하는가?
- 시스템은 외부 자원을 사용하는가?
- 시스템에 대해 이미 액터가 정의되어 있는가?
- 분석 동안 모델링 되어야 하는 시스템과 상호 작용하는 하드웨어 또는 소프트웨어 디바이스가 존재하는가?
- 만약 시스템 내에서 이벤트가 발생한다면, 외부 개체가 이 이벤트를 통지 받을 필요가 있는가? 시스템이 태스크 수행을 돕기 위해 외부 개체에 이를 요청할 필요가 있는가?

## 유스케이스 내용 - 추가 항목

### □ 추가 항목

- 사전조건(pre-condition)
  - 유스케이스 시작 전에 시스템이 보장해야 하는 사항
- 보증(guarantee)
  - 유스케이스 종료 시에 시스템이 보장하는 사항
  - 성공 보증(success guarantee)은 성공 시나리오 종료 시 보장하는 사항 표시
  - 최소 보증(minimal guarantee)은 임의의 시나리오 종료 시 보장하는 사항 표시
- 트리거(trigger)
  - 유스케이스가 시작되도록 한 이벤트
- 유스케이스는 간략하게 유지하는 것이 중요
  - 긴 유스케이스는 가독성 저하




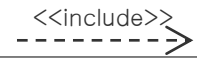
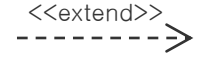


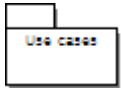

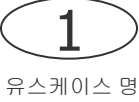
## 2. 유스케이스 다이어그램

- ❑ 유스케이스 다이어그램
- ❑ 유스케이스 수준
- ❑ 유스케이스 목표 수준
- ❑ 유스케이스 크기
- ❑ 유스케이스 분리

ONE STEP AHEAD



## □ 표기법(Notation)

시스템 경계		시스템의 경계와 시스템을 위한 유스케이스를 포함. 액터들은 시스템 경계 외부에 위치한다.
연관관계		직접적으로 상호작용하는 액터와 유스케이스 간의 관계. 액터는 하나 이상의 유스케이스와 연관될 수 있으며, 유스케이스 또한 하나 이상의 액터와 연관될 수 있음.
단방향 연관관계		통신의 개시자(Initiator)를 나타내는 연관
포함 관계		하나의 유스케이스가 다른 유스케이스를 사용함을 나타내는 두 유스케이스 간의 관계
확장 관계		하나의 유스케이스가 다른 유스케이스의 행동을 추가함을 나타내는 두 유스케이스 간의 관계
일반화 관계		액터 간의 계승 관계를 나타내는 유스케이스
액터		시스템과 상호작용하는 사용자, 시스템, 다른 요소를 나타내는 외부 개체
패키지		유스케이스(그리고 UML의 다른 요소들)의 컨테이너. 다른 속성을 가진 임의의 수의 유스케이스 모델을 구성하기 위해 사용
유스케이스		시스템에 의하여 수행되는 행위들의 집합을 설명
우선순위 유스케이스		좀 더 긴급하거나 중요한 유스케이스를 구별하기 위해 우선순위가 부여된 유스케이스

## 유스케이스 다이어그램(2/3)

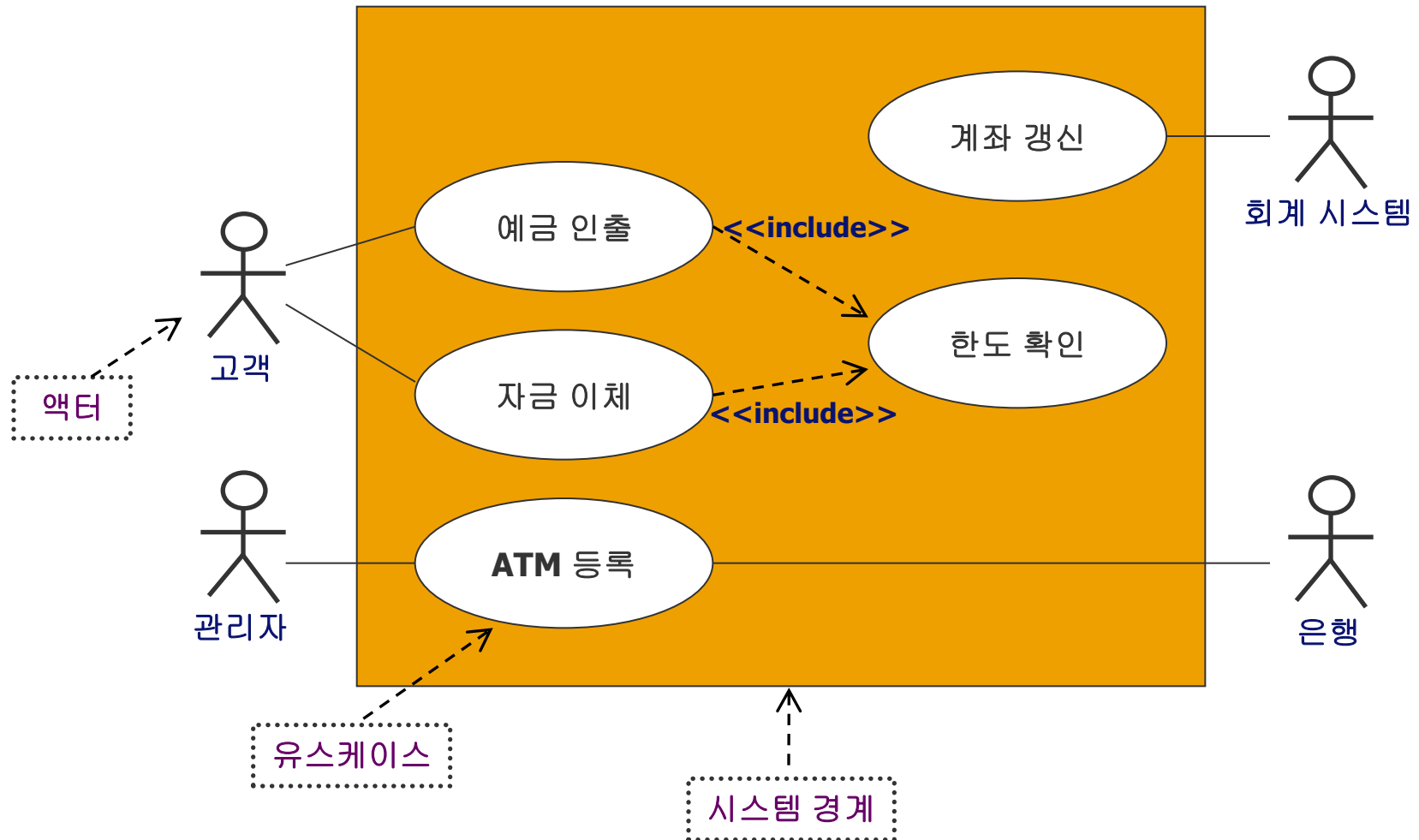
### □ UML과 유스케이스

- 유스케이스 내용에 대한 언급은 없음
- 유스케이스 다이어그램 형식만을 제공
- 중요한 것은 유스케이스 내용
  - 유스케이스 다이어그램에 너무 많은 시간을 투자하지 말 것

### □ 유스케이스 다이어그램

- 유스케이스들의 목차의 그래픽적인 테이블 표현
- 액터, 유스케이스, 액터와 유스케이스 간의 관계로 구성
- 관계의 종류
  - 액터가 유스케이스를 실행
  - 유스케이스가 다른 유스케이스를 포함
- 포함관계 이외의 유스케이스간 관계는 무시할 것
  - 대신 유스케이스의 텍스트 서술에 집중

## □ 유스케이스 다이어그램의 예



## □ 유스케이스 수준 개요

- 요약 목표
- 사용자 목표 <- FOCUS!!!
- 하위 기능

## □ 요소 비즈니스 프로세스(EBP :Elementary Business Process)

- 사용자 목표 수준 유스케이스의 크기와 유사
- 어떤 비즈니스 이벤트에 대처하기 위해서 한 사람이 한 장소에서 특정 시점에서 수행할 수 있는 작업의 단위
- 비즈니스 가치를 생성할 수 있는 수준
  - 예) 주문하기: 이는 '문서 출력하기'나 '항목 삭제하기'와 같은 수준이 아니라 주요 흐름이 5~10 스텝 정도 존재하는 크기

## 유스케이스 수준(2/3)

### □ 요소 비즈니스 프로세스 적용

- 유스케이스 작업을 끝난 후에 일을 했다는 만족도를 느끼는 정도
- 한 사람이 한자리에서 수행하는 어떤 테스트를 수행하는 정도(2~20분)
- 상급자에게 자신의 수행한 작업을 보고할 수 있는 수준 ( Boss 테스트 )
  - 예) ‘오늘 로그인 10번 수행하였습니다.’ vs. ‘오늘 대출을 10건 수행하였습니다.’
- 유스케이스 작업을 수행하면 연말 실적으로 반영되어 좋은 연봉 협상을 할 수 있는 근거가 되는 수준

### □ 사용자 목표 수준으로 볼 수 없는 예제

- ‘온라인 경매를 통한 구매완료’: 온라인 경매는 일반적으로 수일 이상이 걸리는 작업(요약 수준)
- ‘로그인’: 연속 10회 로그인 한다는 작업은 시스템을 사용하는 사람의 업무 책임이나 목적을 만족시키지 못함(하위 기능 수준)

## 유스케이스 수준(3/3)

### □ EBP 가이드라인 적용 예제

다음은 분석가와 사용자(출납원)간의 요구사항 파악을 위하여 오갈 수 있는 대화이다. EBP 가이드라인을 적용하여 사용자 목표 수준의 유스케이스를 파악하는 방법을 알아보자.

**시스템 분석자:** 시스템 사용을 할 때 당신의 목표는 무엇입니까?

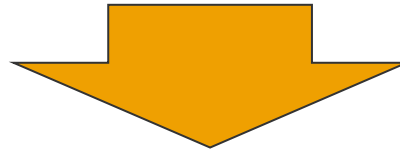
**출납원:** 신속하게 로그인하는 것과, 판매를 파악하는 것입니다.

**시스템 분석자:** 더 높은 수준의 목표를 생각해 봅시다. 무엇 때문에 로그인이 필요하다고 생각하죠?

**출납원:** 시스템에 저의 신분을 확인시킵니다. 그러면, 판매 파악과 기타업무를 위해 제가 시스템을 이용하는 것이 허용되는가를 시스템이 검증할 수 있습니다.

**시스템 분석자:** 왜 그러한 일을 한다고 생각하시지요?

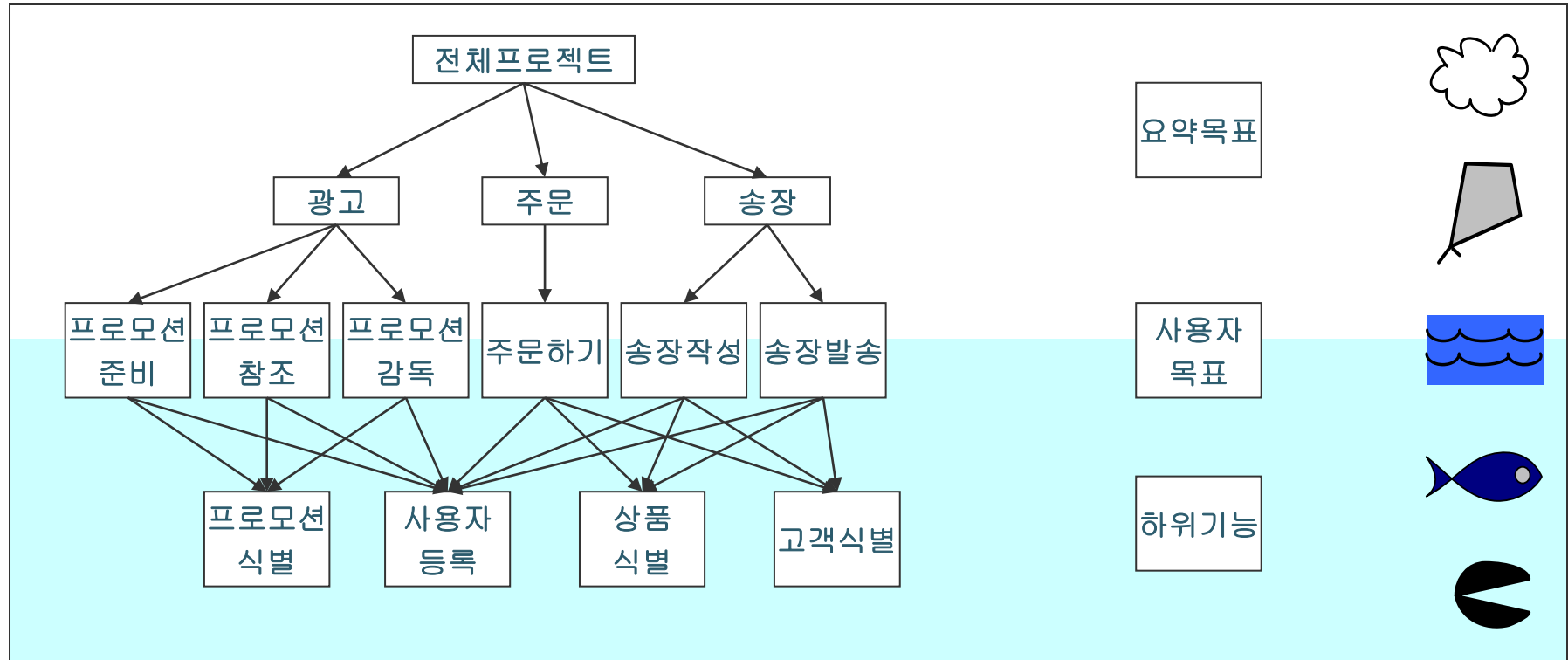
**출납원:** 도난을 방지하고, 데이터 변조를 막고, 회사의 정보 유출을 막기 위해서입니다.



“도난을 방지하고~” 부분은 사용자 목표보다 높은 수준이다. 대화 중 “신분을 확인 시키고 검증 받는다”와 같은 목표가 사용자 목표 수준에 가깝다. 하지만, 이는 비즈니스 가치를 부가하지는 않을 것이다. 즉, “판매를 파악한다”와 같은 내용이 EBP가이드라인을 따르는 사용자 목표 수준의 유스케이스로 적합하다고 보면 된다.

# 유스케이스 목표 수준(1/5)

- 하위 그리고 또 그 하위-하위 목표를 가지는 목표들
- 어떤 수준의 목표를 서술해야 하는지에 대한 유스케이스 작성시의 혼란
- 유용한 도구 : 목표수준에 이름 붙이기



## 유스케이스 목표 수준(2/5)

### □ 사용자 목표 찾기

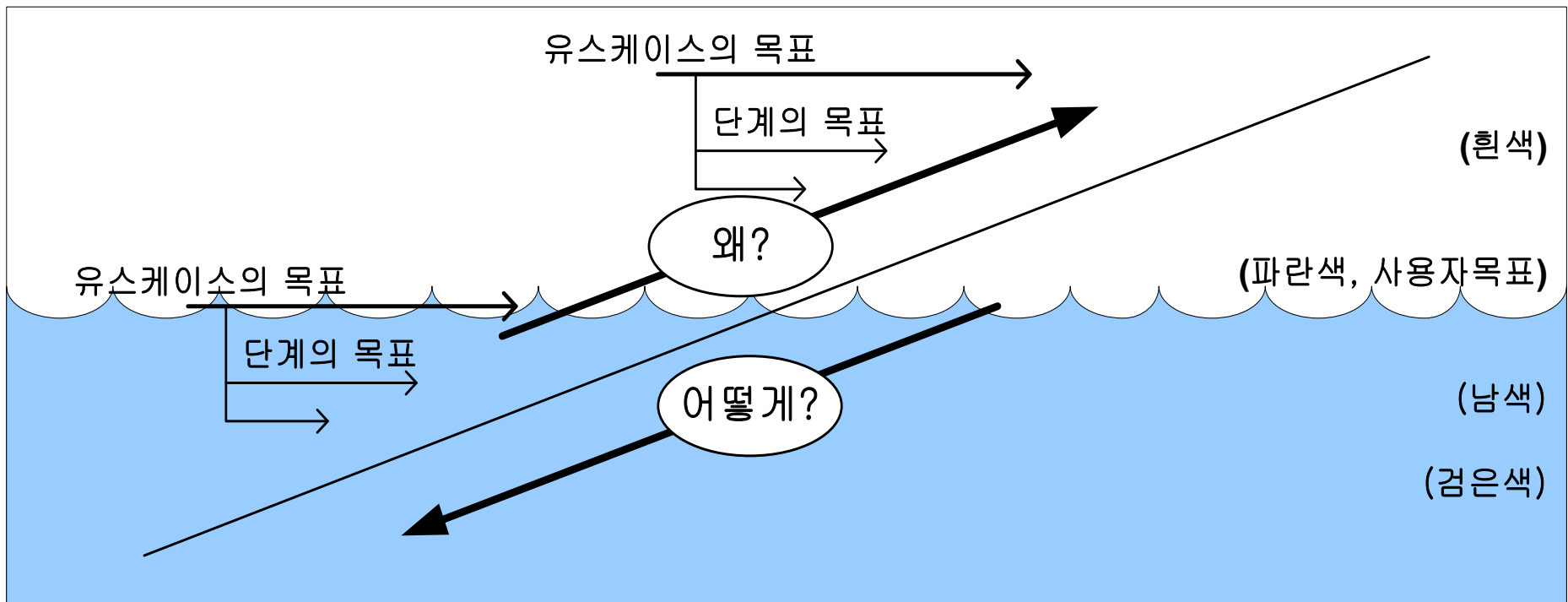
- 어떤 사람이 시스템에서 무엇인가를 원한다. 그것을 얻은 뒤 그(녀)는 계속 하거나 다른 일을 한다. 지금 그녀가 여러분의 시스템에서 원하는 것은 무엇인가?
- 이 수준은 비즈니스 프로세스 모델링의 기초 비즈니스 프로세스, 유스케이스의 사용자 목표
- 사용자 목표를 찾기 위한 두 가지 질문
  - 일차액터가 정말로 원하는 것은 무엇인가?
  - 이 액터가 왜 이 일을 하는가?



## 유스케이스 목표 수준(3/5)

### □ 목표 수준 높이기와 낮추기

- 액터는 왜 이 일을 하는가? 어떻게 하는가?



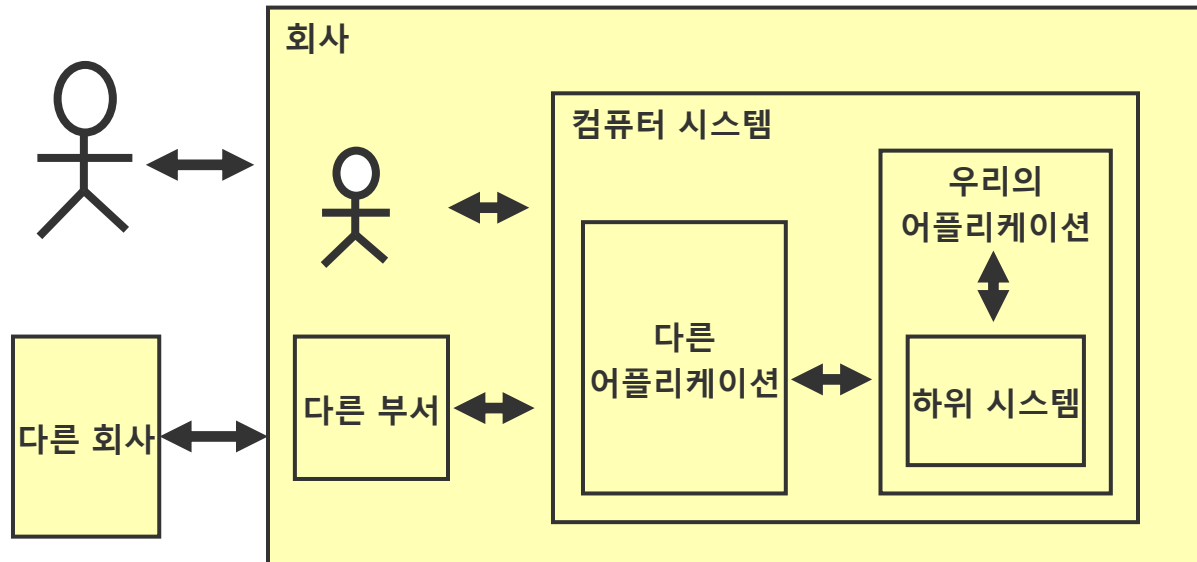
## 유스케이스 목표 수준(4/5)

### □ 유스케이스의 길이

- 잘 작성된 유스케이스 대부분은 3~8단계. 최고 11단계
- 10단계 이상이라면 유저인터페이스 세부사항을 포함했다거나, 너무 낮은 수준의 행동단계를 기술했을 가능성이 크다. 이러한 경우:
  - 유저인터페이스의 세부사항을 제거한다. 액터의 움직임이 아닌 의도를 보여 줌
  - 한 단계 위의 목표 수준을 찾기 위해 "왜"라는 질문을 함으로써 목표수준을 높임
  - 단계를 통합
- 대부분의 유스케이스 길이는 두 페이지 정도가 적절

## □ 설계 범위

- 시스템의 영역
- 설계나 논의에 대한 책임을 지는 하드웨어와 소프트웨어로 구성된 시스템의 집합 경계
- 모든 유스케이스의 관심 주제임



## □ 가장 바깥쪽 유스케이스

- 모든 유스케이스에 대해 가장 바깥쪽의 설계 범위
- 작성시 장점
  - 적은 시간 안에 작성 가능함
  - 시스템의 컨텍스트를 명확히 제공함
  - 시스템의 가장 바깥쪽 사용자(주로 비즈니스 액터)에게 제공하는 궁극적 이익을 가시화함
  - 시스템의 행위를 전체적으로 훑어볼수 있도록 가시화함

## 유스케이스 크기(2/3)

### □ 사용자 목표 수준 유스케이스 - 개략적인 가이드라인

#### ▪ Ivar jacobson의 제안

- '하나의 특정 비즈니스 프로세스를 구현하고 있는 대형 시스템에 있어서 20개 이상의 유스케이스를 도출하지 않는 것이 가장 적절하다' -> 이는 포함, 확장, 일반화 관계 등을 제외한 숫자
- 하지만, 유스케이스 숫자가 20개가 넘거나 적을 경우에 유스케이스 분해나 통합을 통하여 20개의 숫자로 조절하는 작업은 무의미 함
  - 5개의 유스케이스로 훌륭하게 표현된 상업용 시스템이 분명 존재하고, 40개정도의 유스케이스가 적절하게 도출되어 있는 시스템도 존재함
  - 하지만, 700개의 유스케이스가 존재한다면, 이는 분명히 잘못된 모델임

## 유스케이스 크기(3/3)

### □ 사용자 목표 수준 유스케이스 - 개략적인 가이드라인

- 유스케이스 포인트[UCP]
  - 1UCP = 1 Use Case Point
  - 1UCP를 구현하는데 드는 시간은 약 20 Man/Hour 정도
  - 간단한 유스케이스는 대략 5 UCP 정도로 추정
    - 이를 구현하기 위해서는 100M/H(2.5 M/W)가 소요
    - 즉 1명이 2.5 주 정도 걸려 구현할 수 있는 정도의 수준
    - 이는 절대적인 시간은 아니고, 상황에 따라서 많이 달라질 수 있음
    - 개발자의 수준이 매우 높다고 여겨지면, 간단한 유스케이스는 약 1주 이내에도 구현 가능함
- IBM 컨설턴트의 제안
  - 6명이 9개월 정도 수행하는 프로젝트는 보통 대략적으로 10~30개 정도의 유스케이스가 적절함

## □ 개요

- 다른 산출물과 마찬가지로 유스케이스를 작성하는 목적 중의 하나는 의사소통
- 작성된 유스케이스는 읽기에 불편함이 없어야 함

## □ 상황 및 분리 방법

- 대안흐름에 대한 브레인스토밍을 하다 보면 자연적으로 복잡한 확장 흐름이 발생
  - 이 경우 확장 흐름으로부터 하위 유스케이스를 분리하여 포함 관계로 유지
- 하위 유스케이스에 대한 일차 액터의 목표를 정의하고, 이 목표를 하위 유스케이스의 이름으로 명명
- 보통 사용자 목표수준의 유스케이스로부터 분리된 하위 유스케이스는 하위 기능 수준의 유스케이스가 됨

## 유스케이스 분리(2/3)

### □ 분리해야 할 경우

- 특정 흐름이 여러 유스케이스에서 사용
  - 특정 흐름을 하위 유스케이스로 만들어 이를 사용하는 유스케이스가 참조할 수 있게 작성 -> 이것이 하위 기능 수준의 유스케이스를 만드는 가장 유일한 이유
- 확장 흐름이 너무 복잡하여 유스케이스가 읽기 매우 어려움
  - 2페이지 이상의 유스케이스 시나리오가 작성된다던가 확장 흐름의 들여쓰기가 세 번 이상 반복된다면 하위 유스케이스로 분리



## 유스케이스 분리(3/3)

### □ 고려사항

- 유스케이스를 분리하면 프로젝트 관리상의 비용이 증가
  - 단지 이전에 언급한 상황에만 비추어 하위 유스케이스를 기계적으로 분리하는 것은 좋은 방법이 아님
  - 새로운 유스케이스에 대하여 추적하고, 이를 기반으로 프로젝트 일정을 잡고, 이를 검토하고, 유지 보수해야 하는 비용
- 가독성
  - 유스케이스 분리 시 하위 유스케이스의 표시는 밑줄을 그어 표기하는 것이 가독성을 높임
  - 예제

#### 유스케이스: 주문하기

1. 사원이 고객을 식별하고, 시스템은 고객의 기록을 찾아낸다.
2. 사원이 주문 정보를 입력한다.
3. 시스템이 지불금액 계산을 한다.
- ...

#### 유스케이스: 지불금액 계산

1. 시스템은 주문에 대한 기본 지불금액을 계산한다.
2. 시스템은 고객에 대한 할인금액을 계산한다.
- ...