요구사항 개발 및 관리-유스케이스 모델

요구사항이란?

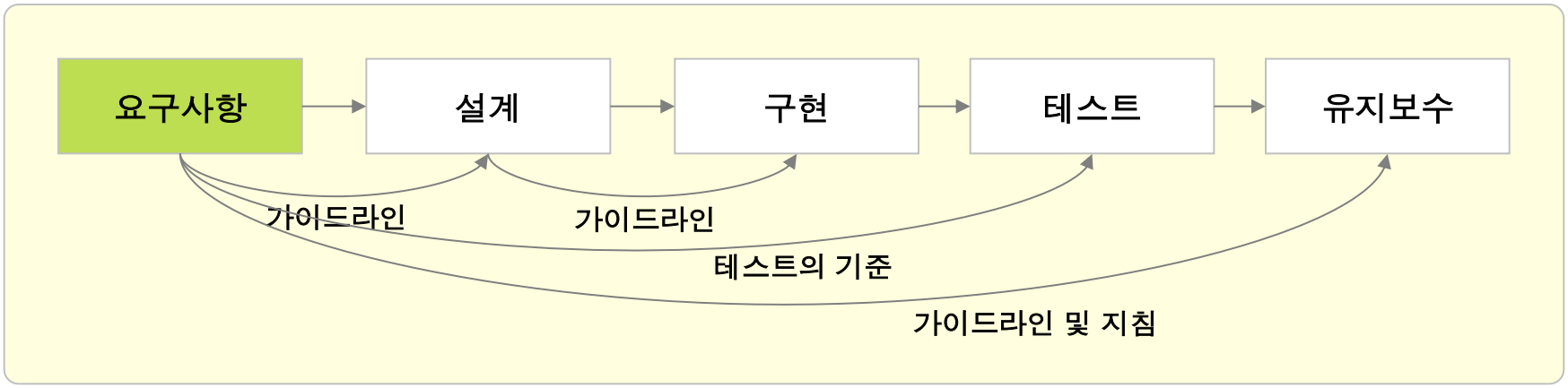
- **정의**
  - 문제의 해결 또는 목적 달성을 위하여 고객에 의해 요구되거나, 표준이나 명세 등을 만족하기 위하여 시스템이 가져야 하는 서비스 또는 제약사항
  - 고객이 요구한 사항과 요구하지 않았더라도 당연히 제공되어야 한다고 가정되는 사항들

if you get the requirements wrong, the resulting application won't solve the users' problems.
You'll be like a tourist in Boston with a broken GPS.

요구사항의 중요성

❖ 요구사항의 중요성

- 참여자들로 하여금 개발되는 소프트웨어 제품을 전체적으로 파악하도록 하여 의사 소통 시간을 절약하게 해 주는 것
- 상세한 요구사항이 있어야만 산정이 가능하고, 이를 기반으로 계획을 세울 수 있기 때문



요구사항의 분류

❖ 기능적 요구사항(Functional Requirements)

- **수행될 기능과 관련되어 입력과 출력 및 그들 사이의 처리과정**
- **목표로 하는 제품의 구현을 위해 소프트웨어가 가져야 하는 기능적 속성**
  - 예) 워드 프로세서에서 파일 저장 기능, 편집 기능, 보기 기능 등

❖ 비기능적 요구사항(Non-Functional Requirements)

- **제품의 품질 기준 등을 만족시키기 위해 소프트웨어가 가져야 하는 성능, 사용의 용이성, 안전성과 같은 행위적 특성**
- **시스템의 기능에 관련되지 않는 사항을 나타냄**
  - 예) 성능(응답 시간, 처리량), 사용의 용이성, 신뢰도, 보안성, 운용상의 제약, 안전성 등
- **아키텍처 결정에 많은 영향을 준다**
- **품질속성이라고도 한다**

예

"Allow users to reserve a hovercraft online."

"The application must support 20 users simultaneously making reservations at any hour of the day."

또 다른 형태의 요구사항

- 표준준수
- 인터페이스 요구사항
- 물리적 요구사항
  - require a minimum amount of processing power,
  - a maximum amount of electrical power,
  - Easy portability (such as a tablet or smartphone), touch screens, or environmental features (must work in boiling acid).

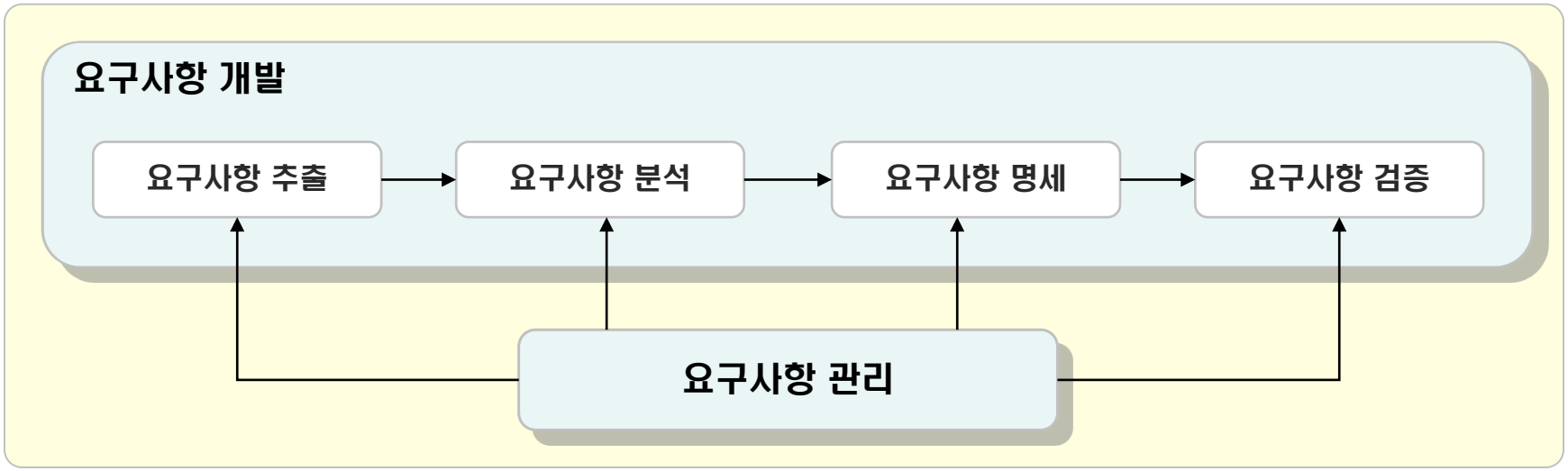
요구사항 개발 프로세스

요구사항 개발

❖ 의미

- 발주자나 고객으로부터 구현될 소프트웨어 제품의 사양을 정확히 도출하여 요구사항을 명세하고, 이를 분석한 결과를 개발자들이 이해할 수 있는 형식으로 기술하는 작업

❖ 요구사항 개발 단계



요구사항 추출 (1/2)

❖ 의미

- 고객이 원하는 요구사항을 수집
- 수집된 요구사항을 통해 개발되어야 하는 시스템에 대한 사용자 요구와 시스템 기능 및 제약사항을 식별하고 이해하는 단계

❖ 중요성

- 고객의 최초 요구사항은 추상적이기 때문에 수주자는 정확한 요구사항을 파악
- 요구사항은 계약 및 최초 산정의 기본이 됨

요구사항 추출 (2/2)

❖ 요구사항 추출 기법의 종류

– 인터뷰

- 개발될 프로젝트 참여자들과의 직접적인 대화를 통하여 정보를 추출하는 일반적인 요구사항 추출 기법
- 획득 가능한 정보
  - ➢ 개발된 제품이 사용될 조직 안에서의 작업 수행 과정에 대한 정보
  - ➢ 사용자들에 관한 정보 등
- 요구사항 분석가는 인터뷰 전략을 세우고 목표를 달성해야 함

– 시나리오

- 시스템과 사용자간에 상호 작용을 시나리오로 작성하여 시스템 요구사항을 추출
- 시나리오에 포함해야 할 필수 정보
  - ➢ 시나리오로 들어가기 이전의 시스템 상태에 대한 기술
  - ➢ 정상적인 사건의 흐름
  - ➢ 정상적인 사건의 흐름에 대한 예외 흐름
  - ➢ 동시에 수행되어야 할 다른 행위의 정보
  - ➢ 시나리오의 완료 후에 시스템 상태의 기술

요구사항 분석 (1/2)

❖ 의미

- 추출된 고객의 요구사항을 분석 기법을 이용하여 식별 가능한 문제들을 도출하고 요구사항을 이해하는 과정
- 참여자들로부터 추상적 요구사항을 명세서 작성 전에 완전하고 일관성 있는 요구사항으로 정리하는 활동

❖ 요구사항 분석의 기준

- 시스템을 계층적이고 구조적으로 표현하여야 한다.
- 외부 사용자와의 인터페이스 및 내부 시스템 구성요소 간의 인터페이스를 정확히 분석하여야 한다.
- 분석단계 이후의 설계와 구현단계에 필요한 정보를 제공하여야 한다.

요구사항 분석 (2/2)

❖ 요구사항 분석 기법의 종류

- **구조적 분석(Structured Analysis)**
  - 시스템의 기능을 중심으로 구조적 분석을 실행
  - 시스템의 기능을 정의하기 위해서 프로세스들을 도출하고, 도출된 프로세스 간의 데이터 흐름을 정의

- **객체지향 분석(Object-Oriented Analysis)**
  - 요구사항을 사용자 중심의 시나리오 분석을 통해 유스케이스 모델(Usecase Model)로 구축하는 것
  - 요구사항을 추출하고, 유스케이스의 실체화(Realization)과정을 통해 추출된 요구사항을 분석

요구사항 명세 (1/4)

❖ 의미

- 분석된 요구사항을 명확하고 완전하게 기록하는 것
- 소프트웨어 시스템이 수행하여야 할 모든 기능과 시스템에 관련된 구현상의 제약 조건 및 개발자와 사용자가 합의한 성능에 관한 사항 등을 명세

❖ 최종 결과물

- 요구사항 명세서(SRS: Software Requirement Specification)

요구사항 명세 (2/4)

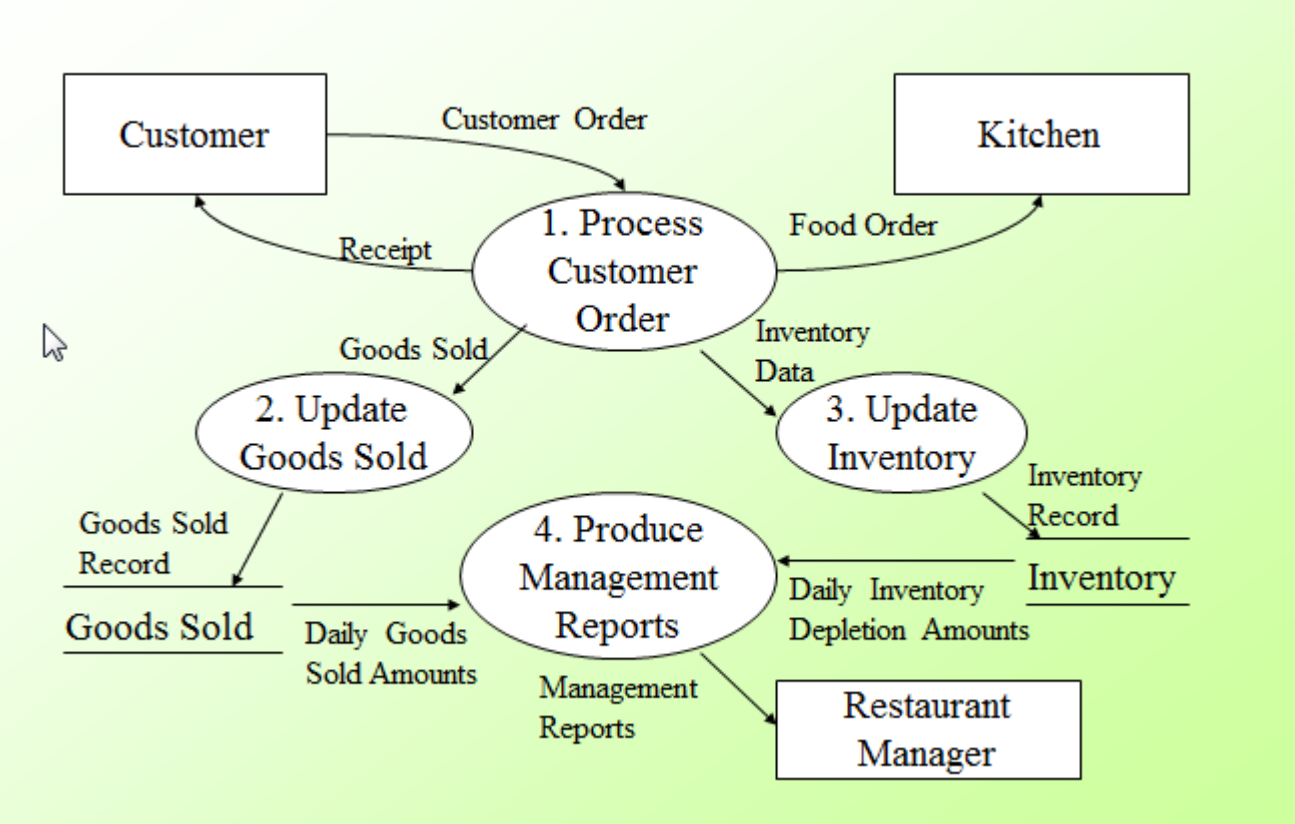
- **요구사항 명세서(SRS: Software Requirement Specification)**
  - **프로젝트 산출물 중 가장 중요한 문서**
  - **사용자, 분석가, 개발자 및 테스터 모두에게 공동의 목표를 제시**
  - **시스템이 어떻게 수행될 것인가가 아닌 무엇을 수행할 것인가에 대한 기술**
    - 시스템이 이루어야 할 목표를 기술하지만 목표를 달성하기 위한 해결 방법은 기술하지 않음

요구사항 명세 (3/4)

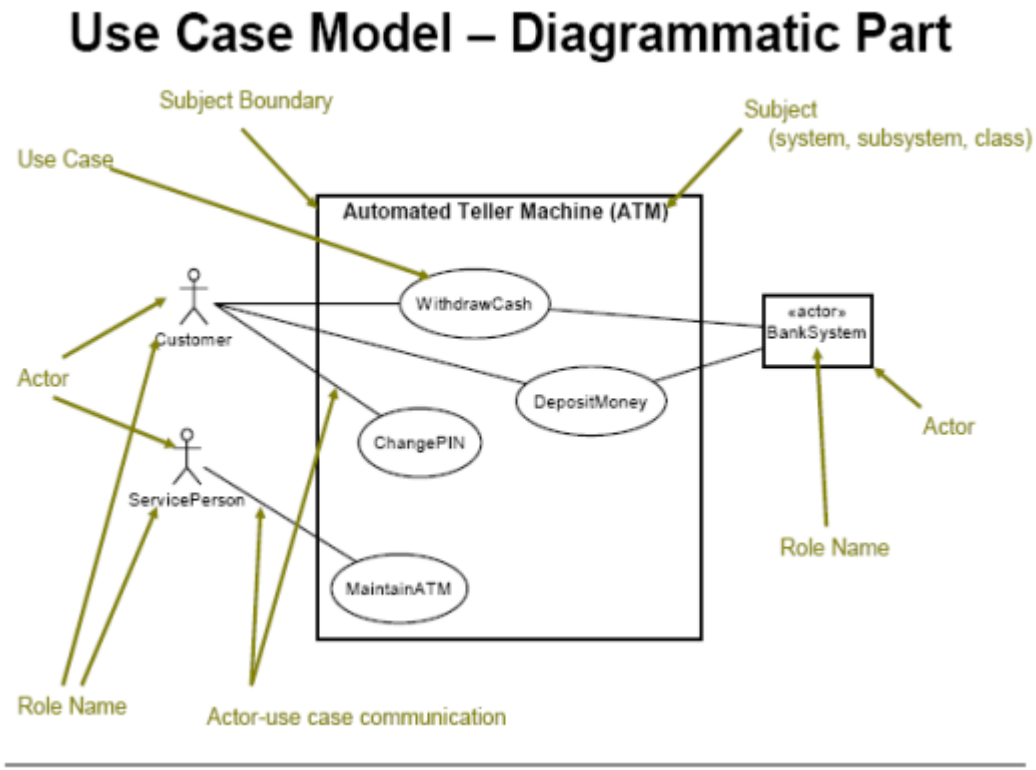
❖ 요구사항 명세서 작성 방법

- 시스템이 수행할 모든 기능과 시스템에 영향을 미치는 제약 조건을 명확하게 기술
- 명세 내용은 고객과 개발자 사이에서 모두가 이해하기 쉽고 간결하게 작성
- 기술된 모든 요구사항은 검증이 가능하기 때문에 원하는 시스템의 품질, 상대적 중요도, 품질의 측정, 검증 방법 및 기준 등을 명시
- 요구사항 명세서는 시스템의 외부 행위를 기술하는 것으로 특정한 구조나 알고리즘을 사용하여 설계하지 않도록 함
- 참여자들이 시스템의 기능을 이해하거나, 변경에 대한 영향 분석 등을 위하여 계층적으로 구성
- 요구사항을 쉽게 참조할 수 있도록 고유의 식별자를 가지고 번호화하고, 모든 요구사항이 동등한 것이 아니기 때문에 요구사항을 우선 순위화

자료흐름도


Customer
Customer Order
Kitchen
Receipt
1. Process Customer Order
Food Order
Goods Sold
Inventory Data
2. Update Goods Sold
3. Update Inventory
Inventory Record
Goods Sold Record
4. Produce Management Reports
Daily Inventory Depletion Amounts
Inventory
Goods Sold
Daily Goods Sold Amounts
Management Reports
Restaurant Manager

유스케이스 다이어그램


Use Case Model – Diagrammatic Part
Subject Boundary
Subject
(system, subsystem, class)
Use Case
Automated Teller Machine (ATM)
WithdrawCash
«actor»
BankSystem
Customer
Actor
DepositMoney
ChangePIN
Actor
ServicePerson
Role Name
MaintainATM
Role Name
Actor-use case communication

요구사항 검증 (1/2)

❖ 의미

- 사용자 요구가 요구사항 명세서에 올바르게 기술되었는가에 대해 검토하는 활동

❖ 검증 내용

- 요구사항이 사용자나 고객의 목적을 완전하게 기술하는가?
- 요구사항 명세가 문서 표준을 따르고, 설계 단계의 기초로 적합한가?
- 요구사항 명세의 내부적 일치성과 완정성이 있는가?
- 기술된 요구사항이 참여자의 기대에 일치하는가?

요구사항 검증 (2/2)

❖ 요구사항 타당성 검증 사항

검증 사항	설명
무결성(correctness) 및 완전성(completeness)	사용자의 요구를 에러 없이 완전하게 반영하고 있는가?
일관성(consistency)	요구사항이 서로간에 모순되지 않는가?
명확성(unambiguous)	요구분석의 내용이 모호함 없이 모든 참여자들에 의해 명확하게 이해될 수 있는가?
기능성(functional)	요구사항 명세서가 “어떻게” 보다 “무엇을”에 관점을 두고 기술되었는가?
검증 가능성(verifiable)	요구사항 명세서에 기술된 내용이 사용자의 요구를 만족하는가? 개발된 시스템이 요구사항 분석 내용과 일치하는지를 검증할 수 있는가?
추적 가능성(traceable)	시스템 요구사항과 시스템 설계문서를 추적할 수 있는가?

Clear

- **Bad example:**
  - UR2: All screens must appear on the monitor quickly.
  - **How long is quickly?**

- **Bad example:**
  - UR2: All screens must appear on the monitor quickly.
  - **How long is quickly?**

- **Good example:**
  - UR2: When the user accesses any screen, it must appear on the monitor within 2 seconds.

Unambiguous

- "Find the best route from a start location to a destination location."

complete

- **Bad example:**
  - UR3: On loss of power, the battery backup must support normal operations.
  - **For how long?**

Complete

- **Bad example:**
  - UR3: On loss of power, the battery backup must support normal operations.
  - **For how long?**

- **Good example:**
  - UR3: On loss of power, the battery backup must support normal operations for 20 minutes.

Consistent

- Do not contradict each other
- Do not provide so many constraints that the problem is unsolvable.
- Each requirement must also be self-consistent. (In other words, it must be possible to achieve.)

Reduce appointment start windows to no more than 2 hours while meeting 90 percent of the scheduled appointments.

Feasible

- **Bad example:**
  - The replacement control system shall be installed with no disruption to production.
  - **This is an unrealistic expectation.**

- **Bad example:**
  - The replacement control system shall be installed with no disruption to production.
  - **This is an unrealistic expectation.**

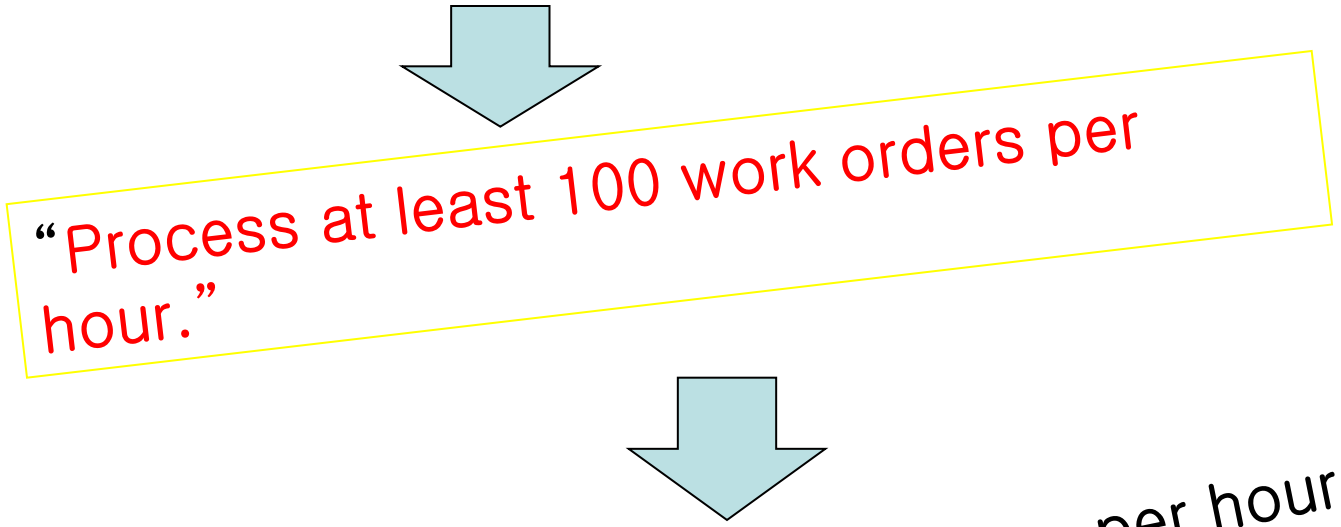
- **Good example:**
  - The replacement control system shall be installed causing no more than 2 days of production disruption.

Fast, Good, Cheap

- Build something quickly with high quality and high cost.
- Build something quickly and inexpensively but with low quality.
- Build with high quality and low cost but over a long time.

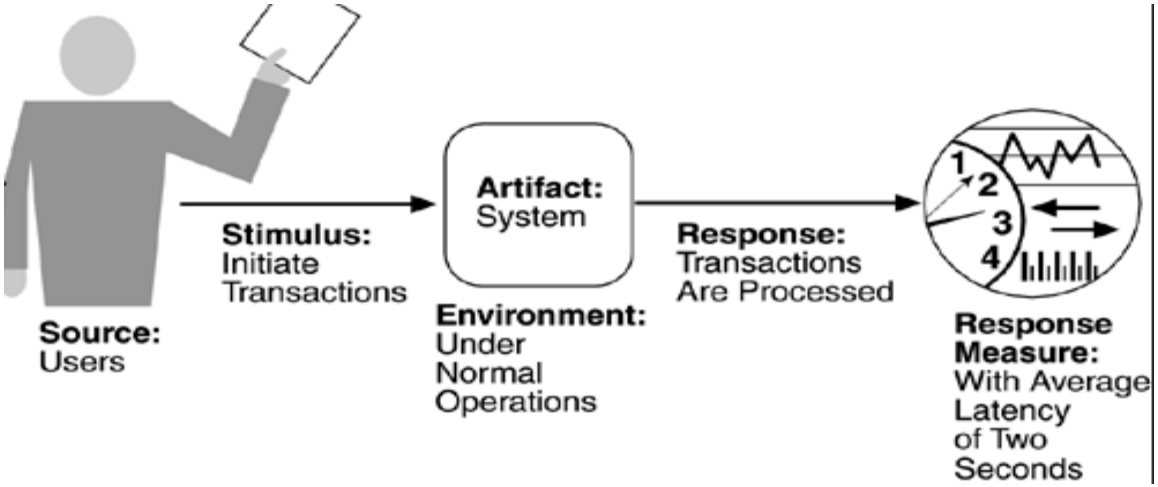
Verifiable

- "Process more work orders per hour than are currently being process



품질속성을 검증가능한 시나리오로

- **품질속성템플릿**
  - 소스(Source) : 자극을 생성하는 개체
  - 자극(stimulus): 시스템에 영향을 주는 조건
  - 대상 또는 산출물(artifact) : 자극에 의해 영향받는 시스템(부분)
  - 환경(environments): 자극이 발생된 상황
  - 반응(response): 자극으로 야기되는 시스템 행위
  - 반응척도(measure): 시스템의 반응을 평가하는 척도

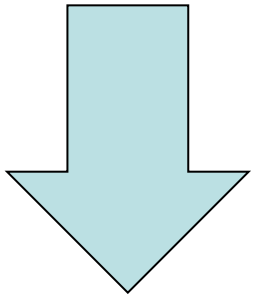


예제

- **사용자**: 등록된 회원이
- **자극**: 주소 변경을 요구했을 때
- **대상**: 회원 정보 갱신 시스템
- **환경**: 시스템이 최대 부하가 걸린 상태에서
- **반응**: 회원계정 갱신 트랜잭션이 수행되며
- **척도**: 이 트랜잭션은 0.15초 내에 완료되어야 한다

What, not on How

- The toppings form will display a list of toppings. The user can check boxes next to the toppings to add them to the bagel.



The toppings form will allow the user to select the toppings put on the bagel.

요구사항 우선 순위

- MOSCOW method
  - M— Must. These are required features that must be included. They are necessary for the project to be considered a success.
  - S— Should. These are important features that should be included if possible. If there’s a work-around and there’s no room in the release 1 schedule, these may be deferred until release 2.
  - C— Could. These are desirable features that can be omitted if they won’t fit in the schedule. They can be pushed back into release 2, but they’re not as important as the “should” features, so they may not make it into release 2, either.
  - W— Won’t. These are completely optional features that the customers have agreed will not be included in the current release. They may be included in a future release if time permits.

유스케이스 기반의 요구사항 분석

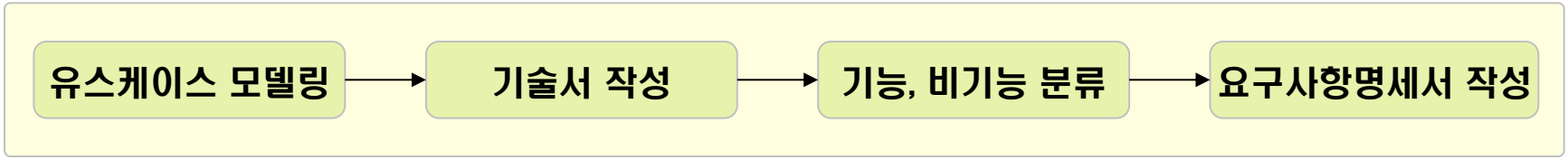
요구사항 분석

❖ 의미

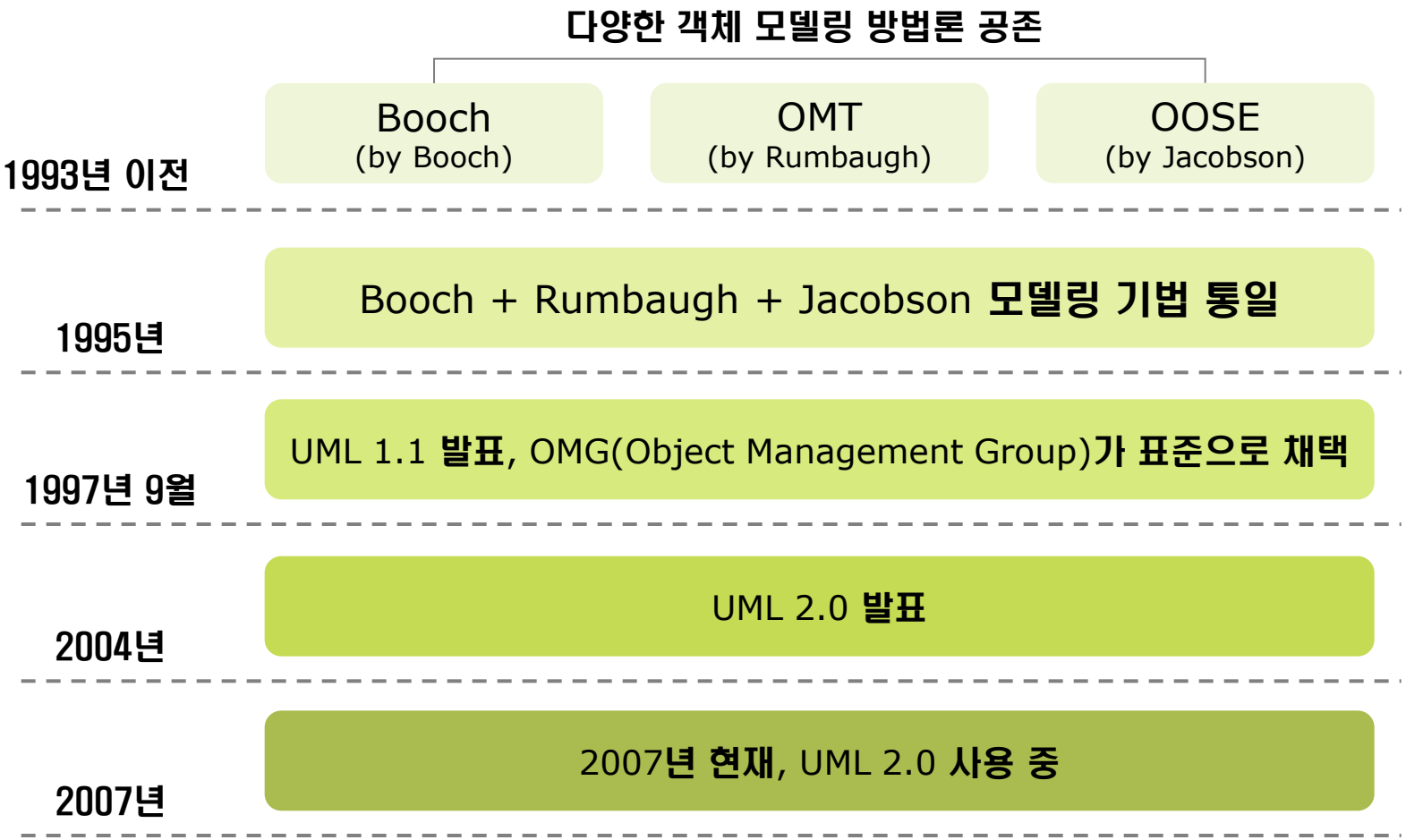
- 요구사항 명세서 작성의 기반을 다지는 작업

❖ 요구사항 분석 방법

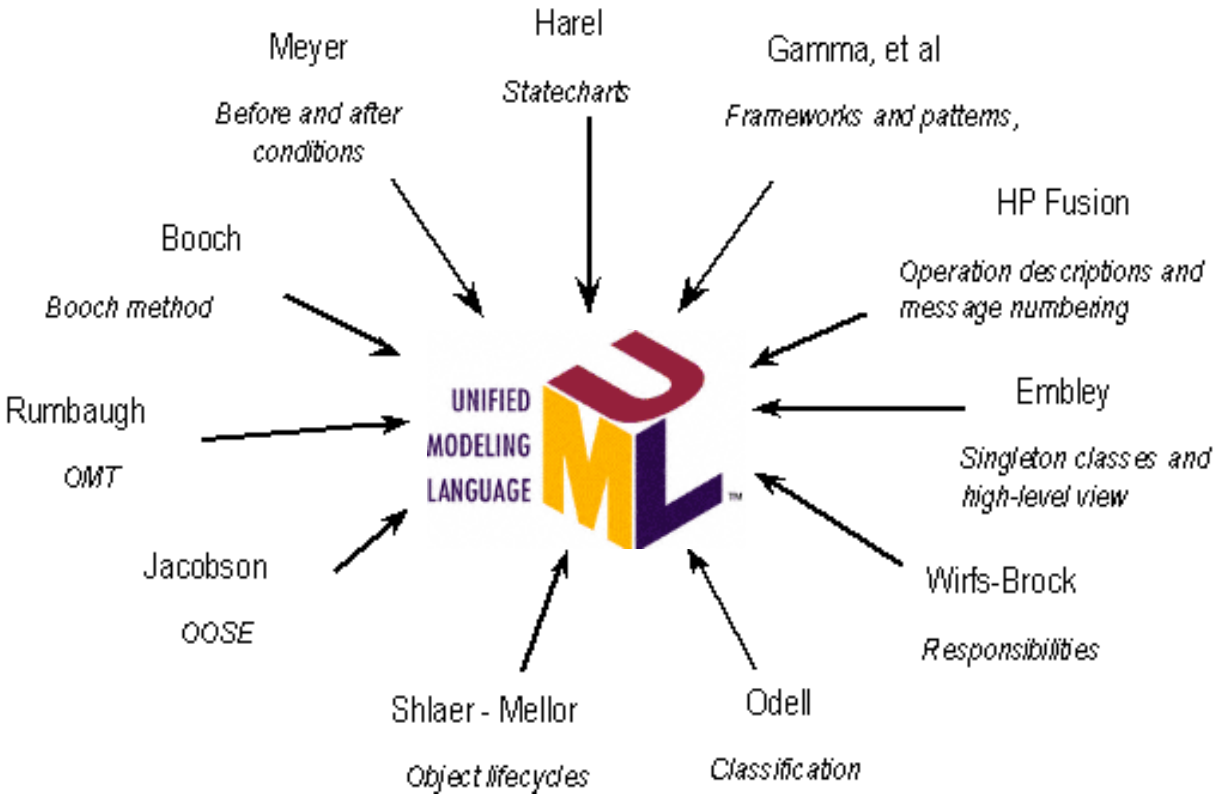
- 객체지향 방법인 유스케이스 기반 분석



UML의 역사

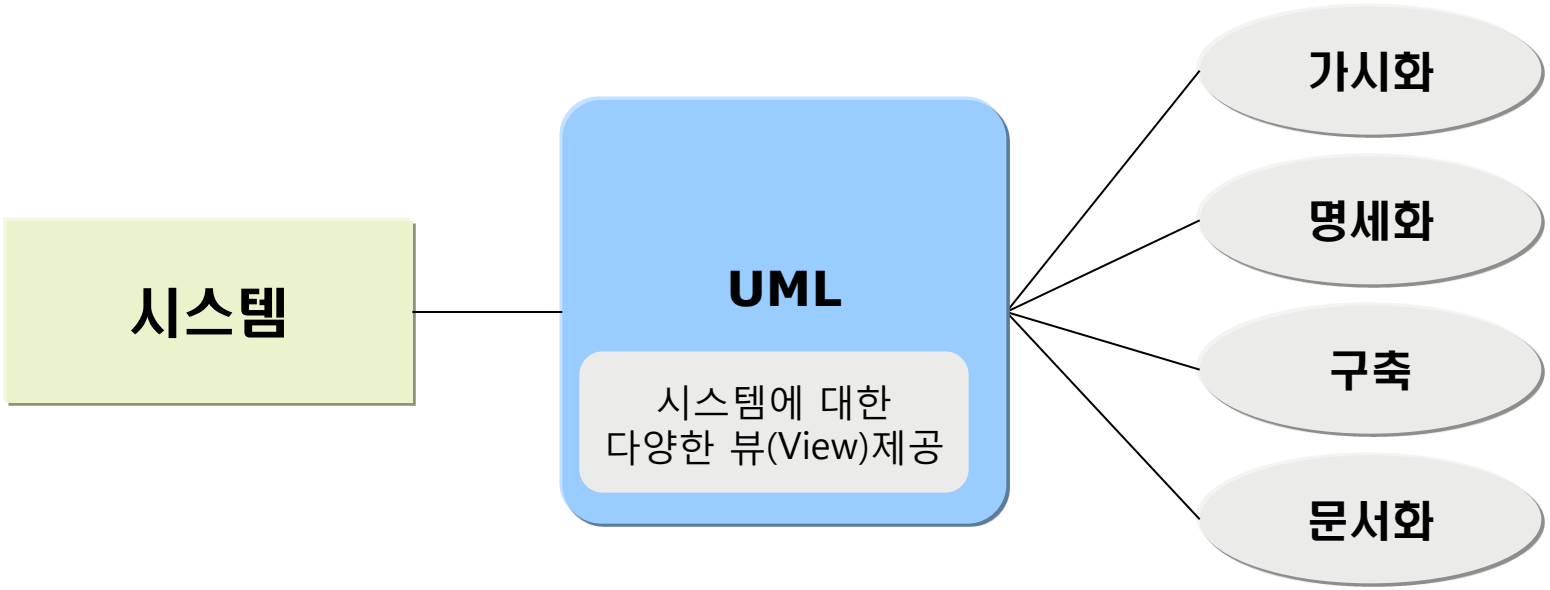

다양한 객체 모델링 방법론 공존
Booch
(by Booch)
OMT
(by Rumbaugh)
OOSE
(by Jacobson)
1993년 이전
Booch + Rumbaugh + Jacobson 모델링 기법 통일
1995년
UML 1.1 발표, OMG(Object Management Group)가 표준으로 채택
1997년 9월
UML 2.0 발표
2004년
2007년 현재, UML 2.0 사용 중
2007년

통합된 표준 모델링 언어, UML



UML의 표기법(notation)만 알고 있다면
프로젝트 이해 관계자간의 의사소통의 불일치를 지적할 수 있다!

시스템 구축 시 UML의 역할

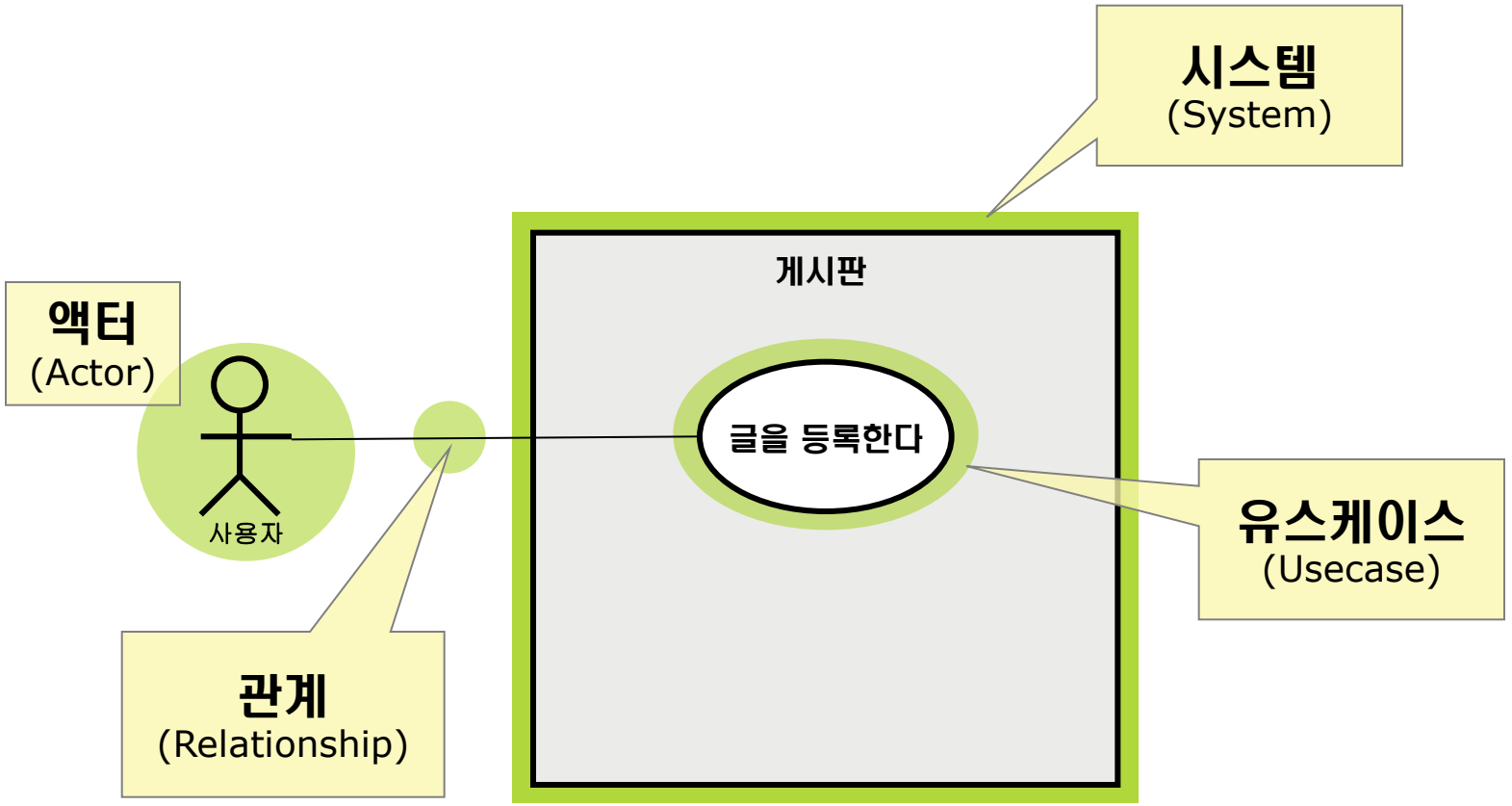


유스케이스 다이어그램(1/2)

❖ 개요

- 사용자의 관점에서 시스템의 서비스 혹은 기능 및 그와 관련한 외부 요소를 보여주는 다이어그램
- 고객과 개발자가 함께 보며 요구사항에 대한 의견을 조율할 수 있음

유스케이스 다이어그램의 구성요소



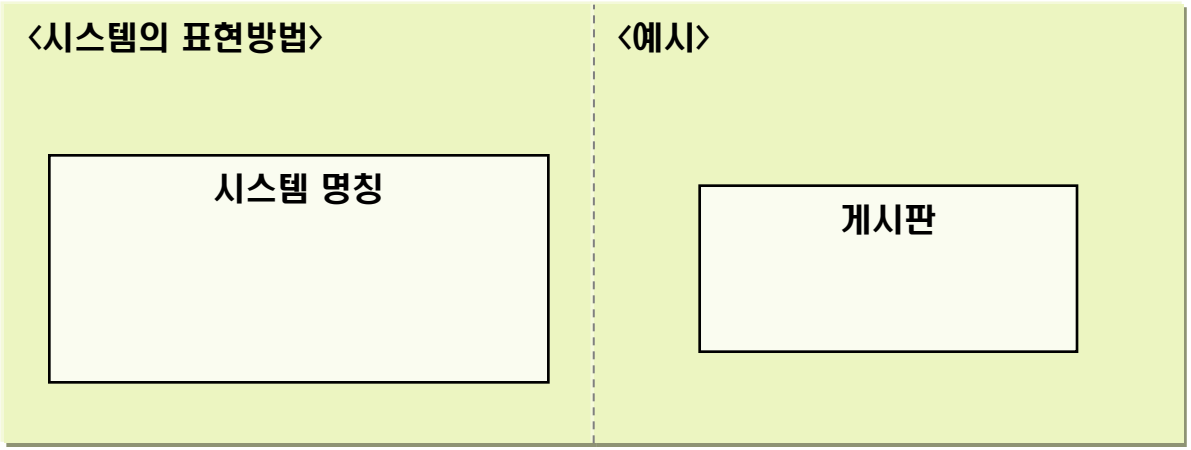
시스템(System)

❖ 의미

- 만들고자 하는 어플리케이션

❖ 표기법

- 유스케이스를 둘러싼 사각형의 틀을 그리고, 시스템 명칭을 사각형 안쪽 상단에 기술



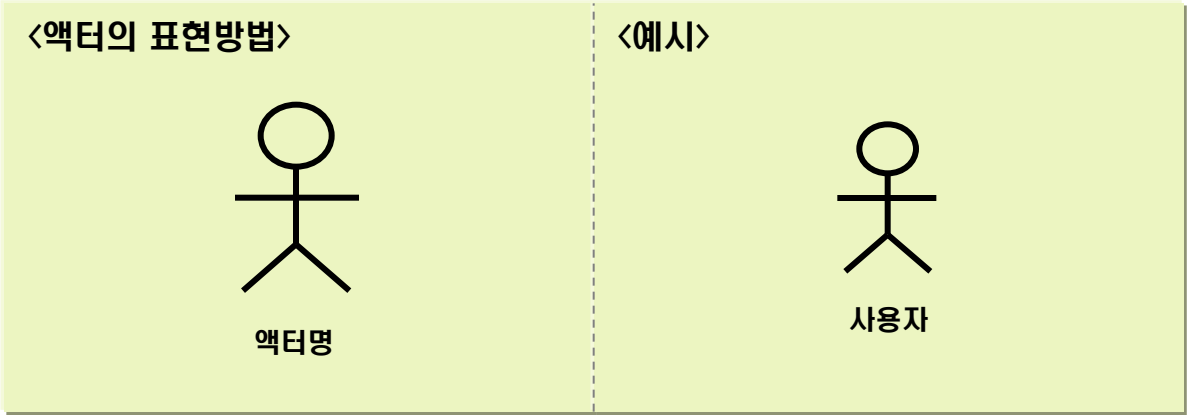
액터(Actor)

❖ 의미

- 시스템의 외부에 있으면서 시스템과 상호 작용을 하는 사람 또는 다른 시스템

❖ 표기법

- 원과 선을 조합하여 사람 모양으로 표현
- 그 위 또는 아래에 액터명 표시
- 액터명은 액터의 역할로 정함



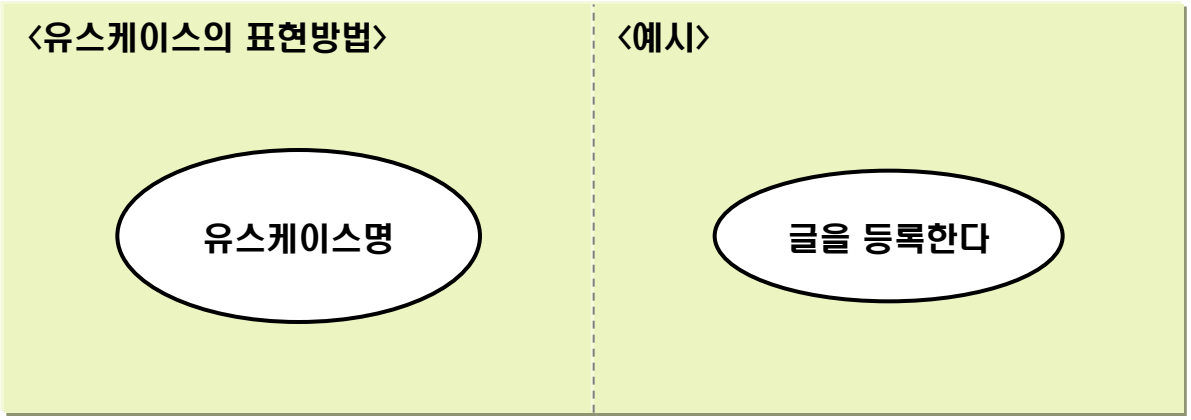
유스케이스(Usecase)

❖ 의미

- 시스템이 액터에게 제공해야 하는 기능의 집합
- 시스템의 요구사항을 보여줌

❖ 표기법

- 타원으로 표시하고 그 안쪽이나 아래쪽에 유스케이스명을 기술
- 유스케이스의 이름은 “~한다” 와 같이 동사로 표현
- 각 유스케이스가 개발될 기능 하나와 연결될 수 있도록 함



관계(Relationship) (1/3)

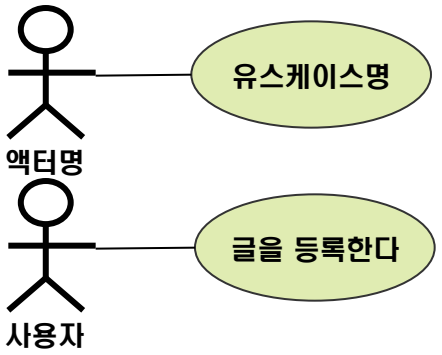
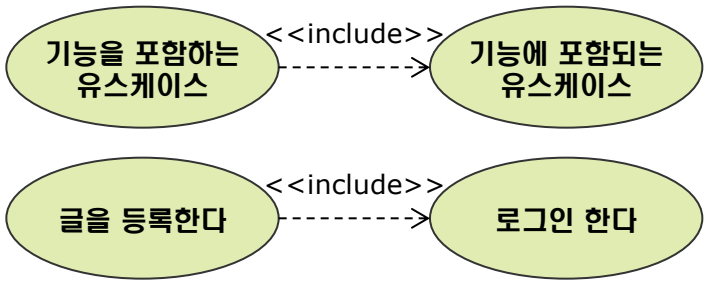
❖ 의미

- 액터와 유스케이스 사이의 의미 있는 관계

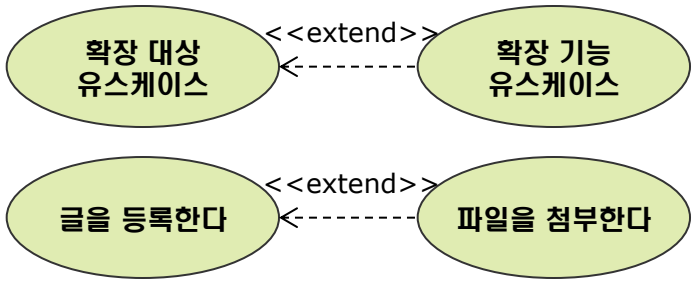
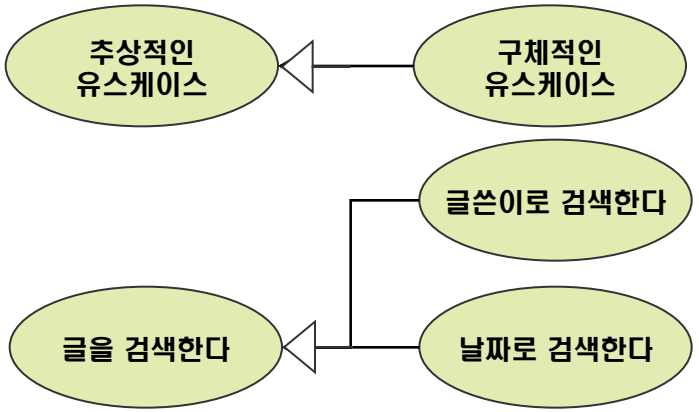
❖ 종류

- 연관 관계(association)
- 의존 관계(dependency)
  - 포함 관계(include)
  - 확장 관계(extend)
- 일반화 관계(generalization)

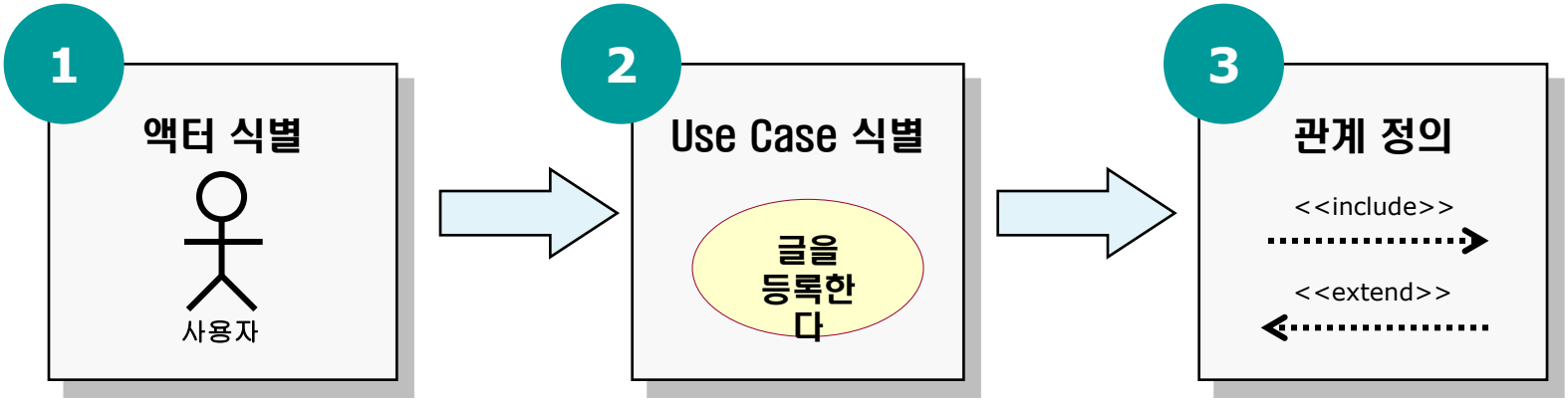
관계(2/3)

관계 종류	설명	표기법
연관 관계 (association)	• 유스케이스와 액터간의 상호작용이 있음을 표현 • 유스케이스와 액터를 실선으로 연결함	
포함 관계 (include)	• 하나의 유스케이스가 다른 유스케이스의 실행을 전제로 할 때 형성되는 관계 • 포함되는 유스케이스는 포함하는 유스케이스를 실행하기 위해 반드시 실행되어야 하는 경우에 적용 • '포함하는 유스케이스'에서 '포함되는 유스케이스' 방향으로 화살표를 점선으로 연결하여 표현하고 <<include>>라고 표기	

관계(3/3)

관계 종류	설명	표기법
확장 관계 **(extend)**	• 확장 기능(Extending) 유스케이스와 확장 대상(Extended) 유스케이스 사이에 형성되는 관계 • 확장 대상 유스케이스를 수행 할 때에 특정 조건에 따라 확장 기능 유스케이스를 수행하기도 하는 경우에 적용 • '확장 기능 유스케이스'에서 '확장 대상 유스케이스' 방향으로 화살표를 점선으로 연결하여 표현하고 <<extend>>라고 표기	
일반화 관계 **(generalization)**	• 유사한 유스케이스들 또는 액터들을 모아 그들을 추상화한 유스케이스 또는 액터와 연결시켜 그룹핑(Grouping)함으로써 이해도를 높이기 위한 관계 • '구체적인 유스케이스'에서 '추상적인 유스케이스' 방향으로 끝부분이 삼각형의 테두리로 표현된 화살표를 실선으로 연결하여 표현	

유스케이스 다이어그램 작성 순서



- 사용자의 역할을 식별
- 외부 시스템 식별

- 액터가 요구하는 서비스 식별
- 액터가 요구하는 정보 식별
- 액터가 시스템과 상호작용하는 행위 식별

- 액터간 유스케이스간의 일반화 정의
- 액터간 유스케이스간의 연관관계 정의
- 유스케이스간 포함, 확장관계 정의

액터 식별

- 액터를 찾기 위한 질문들
  - 누가 정보를 제공하고, 사용하고, 삭제하는가?
  - 누가 또는 어떤 조직에서 개발될 시스템을 사용할 것인가?
  - 누가 요구사항에 대해 관심을 가지고, 시스템이 만들어낸 결과에 관심이 있는가?
  - 누가 시스템이 잘 운영될 수 있도록 유지보수 및 관리를 하는가?
  - 개발될 시스템과 상호작용하는 하드웨어나 소프트웨어 시스템은 무엇인가?

유스케이스 식별

- **유스케이스를 찾기 위한 질문들**
  - 액터가 원하는 시스템 제공 기능은 무엇인가?
  - 액터는 시스템에 어떤 정보를 생성, 수정, 조회, 삭제하고 싶어 하는가?
  - 액터는 시스템의 갑작스러운 외부 변화에 대해 어떤 정보를 필요로 하는가?
  - 시스템이 어떤 기능을 제공하면 액터의 일상 작업이 효율적이고 편리해지는가?
  - 모든 기능 요구사항들을 만족할 수 있도록 유스케이스가 모두 식별되었는가?

관계를 식별하기 위한 질문

- 연관 관계(Association)
  - 액터와 유스케이스 간에 상호 작용이 존재하는가?
- 포함 관계(Include)
  - 이 유스케이스를 실행하기 위하여 반드시 실행되어야 하는 유스케이스가 존재하는가?
- 확장관계(Extend)
  - 이 유스케이스를 실행함으로써 선택적으로 실행되는 유스케이스가 있는가?
- 일반화 관계(Generalization)
  - 액터 또는 유스케이스가 구체화 된 다른 여러 액터나 유스케이스를 가지고 있는가?

[예] 게시판

❖ 예제 요구사항

- SE사는 A고객으로부터 다음의 요구사항을 전달받았다.

· 사용자 요구사항

: 글을 등록, 수정, 삭제할 수 있는 게시판을 개발한다.
(단, 관리자 모드는 개발하지 않는다)

- **글을 등록할 때에는 파일을 첨부할 수 있다.**
- **글을 조회하여 읽을 수 있다.**
- **등록된 글은 글쓴이 혹은 날짜 별로 검색할 수 있다.**
- **게시판의 모든 기능은 사용자 로그인 후에 사용할 수 있다.**

1) 시스템 식별

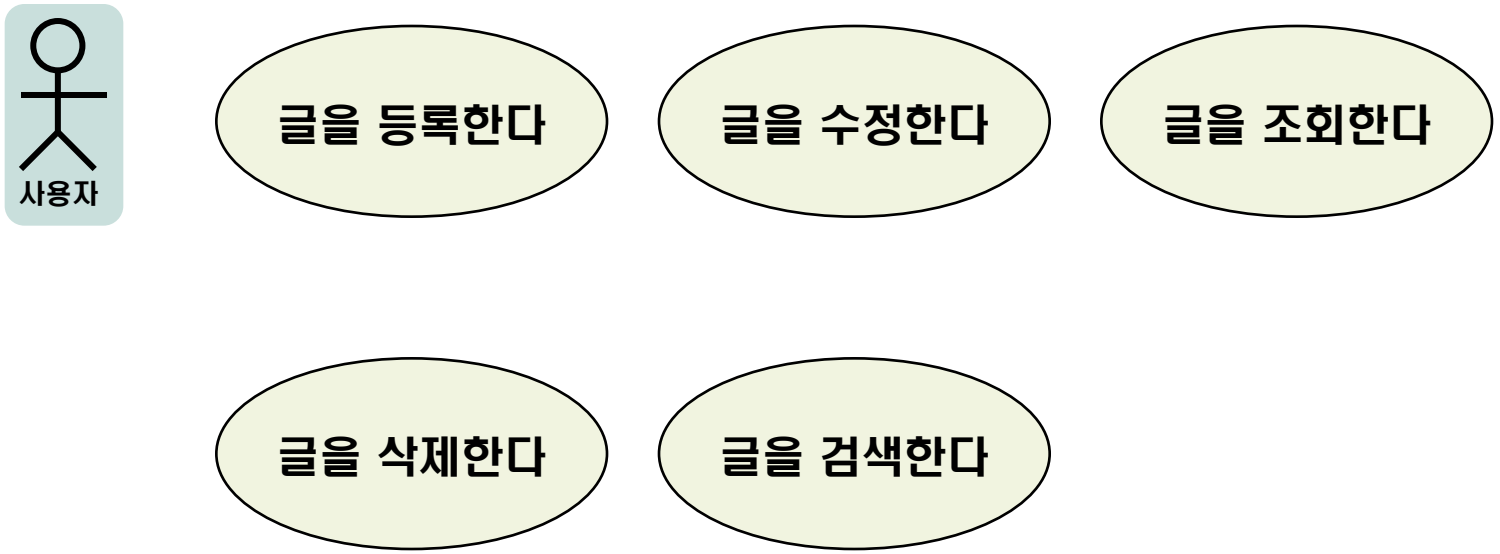
❖ 요구사항을 통해 만들고자 하는 시스템은 ‘게시판’ 임

2) 액터 식별

- 개발할 '게시판' 외부에서 상호작용하는 액터로 글을 등록하고 삭제하는 등의 역할을 하는 '사용자'가 식별됨

3) 유스케이스 식별

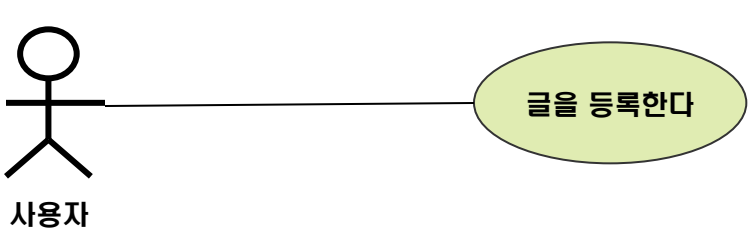
- ‘사용자’ 는 게시판을 통해 글을 등록, 수정, 조회하는 등의 작업을 함



4) 관계 정의

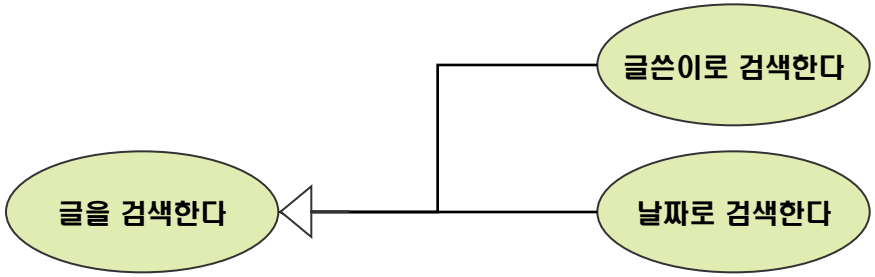
- **연관관계**

(액터-유스케이스)



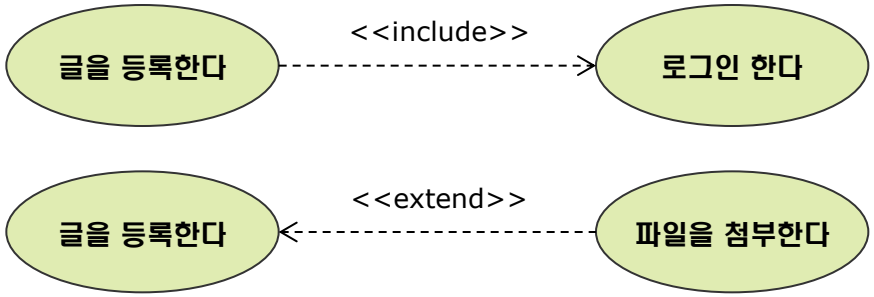
- **일반화관계**

(유스케이스(액터)-유스케이스)

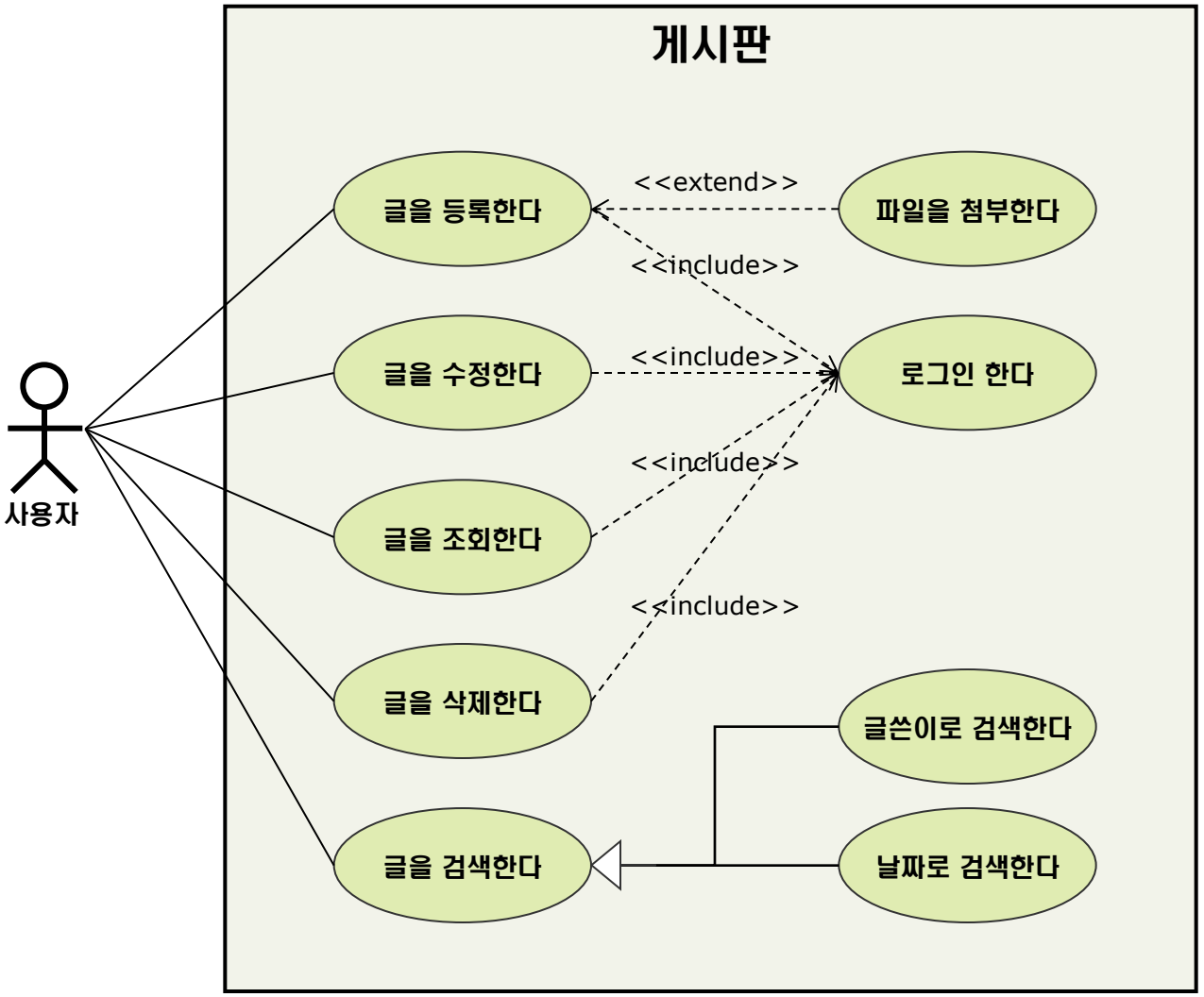


- **포함관계, 확장관계**

(유스케이스-유스케이스)



완성된 게시판 유스케이스 다이어그램



유스케이스 기술서 작성 (1/3)

❖ 개요

- **유스케이스 다이어그램을 보완하기 위한 산출물**
- **유스케이스 다이어그램과의 차이**
  - 유스케이스 다이어그램: 유스케이스는 시스템의 기능을 표현하는 것
  - 유스케이스 기술서: 각각의 유스케이스에 대해서 해당 유스케이스가 어떻게 수행되는지를 표현하는 수단

유스케이스 기술서 작성 (2/3)

❖ 유스케이스 기술서 항목

- 유스케이스 명
- 액터 명
- 유스케이스 개요 및 설명
- 사전 및 사후 조건
- 작업 흐름
  - 정상흐름(Normal Flow): 해당 유스케이스가 정상적으로 수행되는 흐름을 표현하는 절차
  - 대안 흐름(Alternative Flow): 유스케이스 내의 작업 흐름이 수행되는 중에 특정 시점에서 여러 가지 선택적인 흐름으로 나뉘어질 경우에 발생하는 흐름
  - 예외 흐름(Exceptional Flow): 유스케이스 내의 작업 흐름이 수행되는 중에 발생할 수 있는 예외 상황이나 오류를 표현하는 흐름
- 시나리오: 각 시나리오는 유스케이스의 특정한 예를 나타냄

유스케이스 기술서 작성 (3/3)

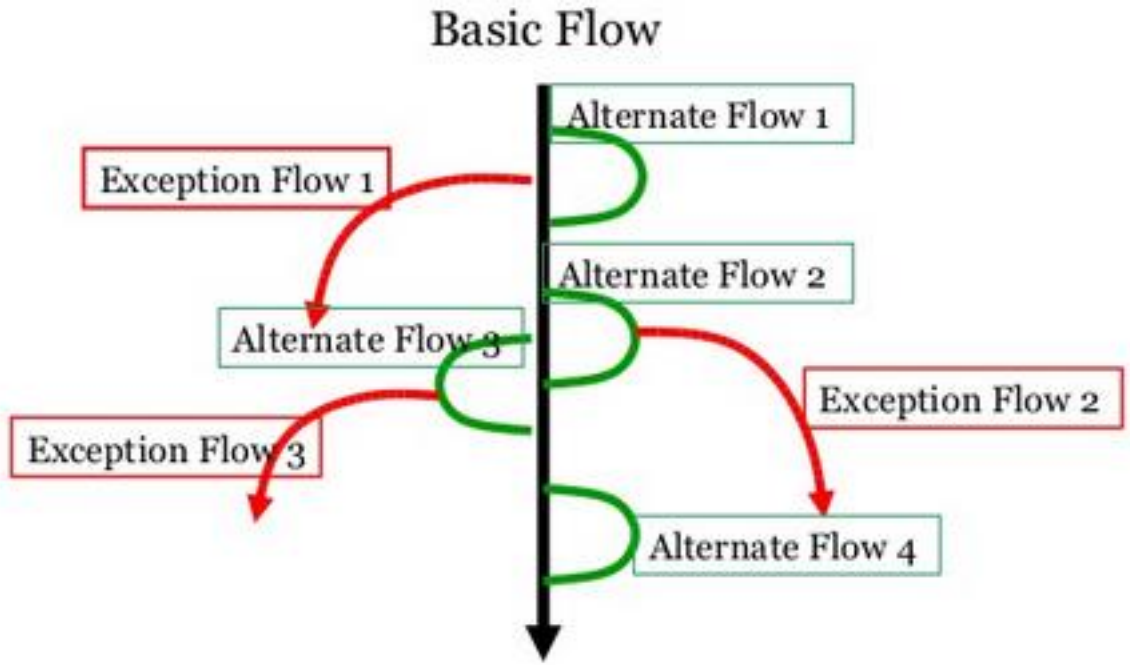
❖ 유스케이스 기술서 예제

- **유스케이스명: 글을 등록한다**
- **액터명: 사용자**
- **유스케이스 개요 및 설명**
  - **사용자는 원하는 글을 게시판에 등록한다.**
- **사전 조건: 로그인 한다**
- **작업 흐름**
  - **정상흐름**
    1. 사용자는 글쓰기 버튼을 선택한다.
    2. 시스템은 글쓰기 박스를 보여한다.
    3. 사용자는 글쓰기 박스에 원하는 글을 작성한다.
    4. 사용자는 등록 버튼을 선택한다.
    5. 시스템은 글을 데이터베이스에 저장한다.
  - **대안 흐름**
    1. 정상 흐름 4에서 등록 취소를 수행하는 경우, 게시판 목록 조회 화면을 표시한다.
  - **예외 흐름**

    1. 정상 흐름 3에서 글쓰기 박스에 글을 쓰지 않고 등록 원하는경우, “내용을 넣으세요” 라는 메시지를 표시한다.

대안 흐름과 예외흐름

- 대안 흐름은 유스케이스를 바로 종결지시키지 않고 기본 흐름과 다른 경로를 실행한다
- 예외 흐름은 유스케이스를 비정상적인 상태로 종결한다



Numbering Scheme

- Flow of events can be nested. Their numbers are of form
  - **CurrentNumber "." Digit** --> sequential flow

    3. Customer selects product. ← **Label for sequence of nested steps**

    3.1. The Customer selects a product entering the number of items required.

    3.2. The System reserves the items in the inventory and adds them to the Customer's shopping cart.
  - **CurrentNumber Character** --> alternative flow

    3.1a. Customer cancels selection dialog. ← **Event occurs**

    3.1a.1. Use case continues at step 4.

Example Authenticate

Use Case: AuthenticateCustomer

Brief Description: The Customer wants to identify him/herself to the System. A project constraint states that identification is made with a card and a personal identification number (PIN). The System uses the service of a Bank Authentication Server (BAS) to verify PIN.

Level: User-goal

Primary Actor: Customer

Main Success Scenario:

1. Customer inserts card; System reads card details*.
2. System validates card type.
3. Customer provides PIN to System.
4. System requests BAS to verify identification information*.
5. BAS informs System that identification information is valid, and System informs Customer.
6. Use case ends.

* -- technical terminology is explained in the glossary

예외 흐름

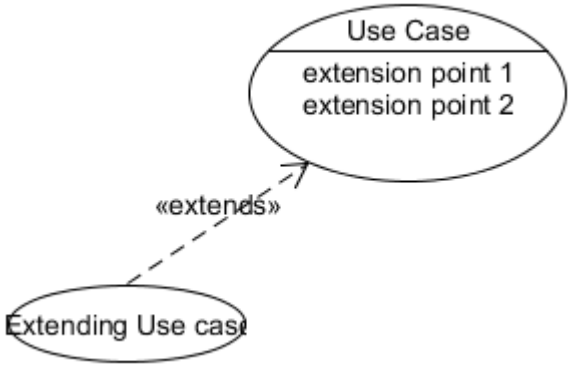
(1-5)a. (at any time) Customer cancels the authentication process:
(1-5)a.1. System ejects the card; use case ends in failure.

1a. System is out of service:
1a.1. System informs the Customer that it is currently out of service; use case ends in failure.

2a. System ascertains that card type is unknown:
2a.1. The System informs the Customer about unknown card type and ejects the card; use case ends in failure.

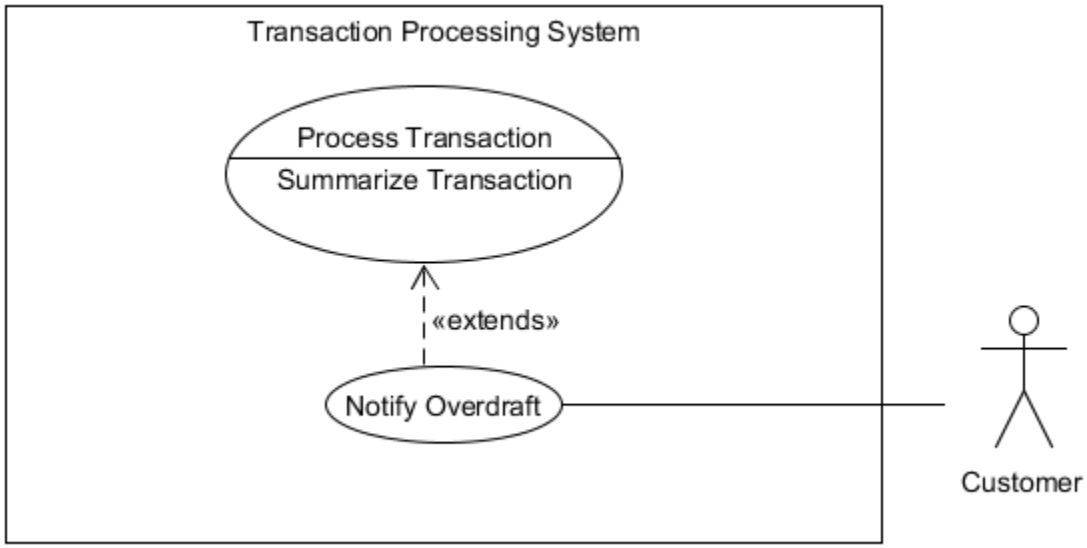
3a. System times out on waiting for Customer to provide PIN:
3a.1. System ejects the card; use case ends in failure.

<<extend>> 관계



- 확장점 1, 2에서 extending use case가 Use Case의 행위를 확장한다
- 확장 관계는 행위를 추가하고자 할 때 사용한다

예



Example <<extend>>

Use Case: ProcessTransactions

Brief Description: Process bank transactions collected for accounts.

Main Success Scenario:

1. At the end of the day (23:58), the use case starts.

Steps 2.-5. are done for each unprocessed transaction:

2. The System determines the account to which the transaction is applied.
3. The System applies the transaction to the account. The balance of the account is increased/decreased by the amount of the transaction.
4. The System records a log for the transaction information.
5. The System marks the transaction is processed.
6. {Summarize Transaction}

   A transaction summary is created for each account.

7. Use case ends.

Label for Extension Point

Example <<extend>>

Use Case: NotifyOverdraft

Brief Description: Notifies the Customer that his account has become overdrawn.

Pre-Condition: This service is only available if the Customer has purchased the overdraft notification service.

Main Success Scenario:

1. The use case starts as extension of ProcessTransactions at {Summarize Transactions} if the Customer has purchased the overdraft notification service and the set of completed transactions has caused the account to become overdrawn.
2. The System determines the Customer's preferred notification mechanism.
3. The System composes the overdraft notification.
4. The System transmits the overdraft notification to the Customer using the Customer's preferred notification mechanism.
5. The use case ends.