Introduction to UML

Team. 5
2014/03/14
원스타 200611494
김성원 200810047
허태경 200811466

KU KONKUK UNIVERSITY

- Index -

1. UML이란?

통합 모델링 언어(Unified Modeling Language)는 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어이다. 이 표준은 UML을 고안한 객체 관리 그룹에서 관리 하고 있다. UML은 소프트웨어 집약 시스템의 시각적 모델을 만들기 위한 도안 표기법을 포함한다.

- 배경

UML은 Rational Software와 그 협력회사에 의해 개발되었다. 업무 처리과정에서 그 업무의 범위와 규모가 커짐에 따른 시스템의 복잡성을 처리할 필요성을 느끼게 되었는데, 특히 물리적인 시스템의 분산, 동시성, 반복성, 보안, 결점 보완, 시스템들의 부하에 대한 균등화와 같은 반복해서 발생하는 구조적 문제에 대한 프로세스가 필요하게 되었다. 또한 웹의 발전에 따라 시스템을 만들기는 쉬워졌으나 이러한 구조적 문제는 더욱 악화되었기에 이러한 모든 필요성에 의해 UML이 만들어졌다.

2. UML Diagram

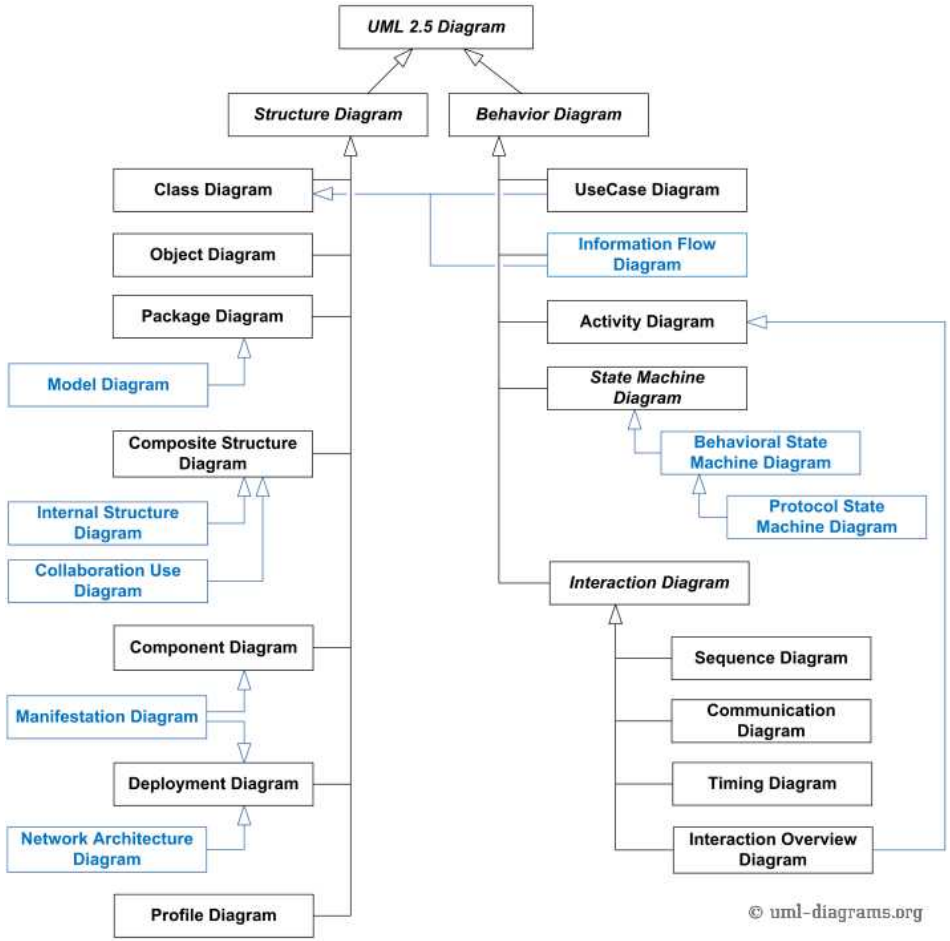
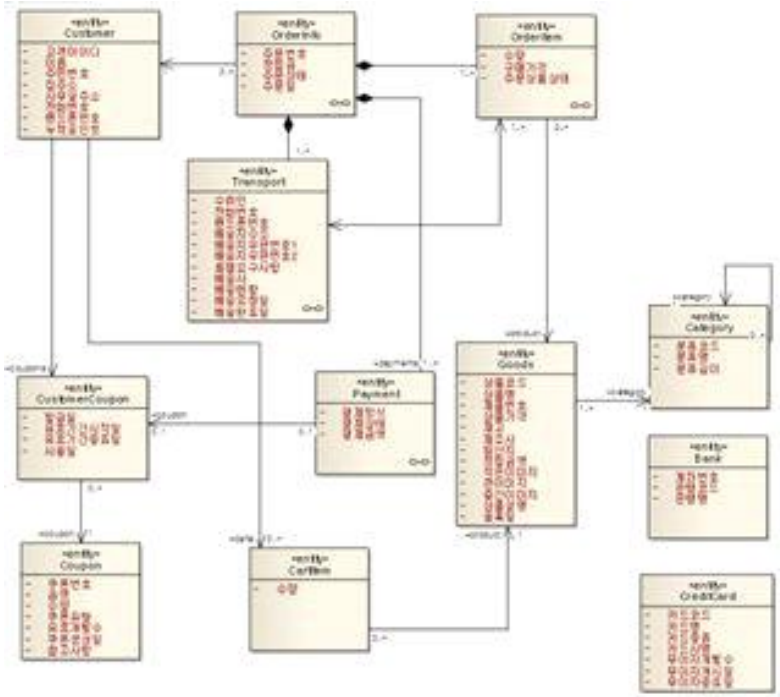


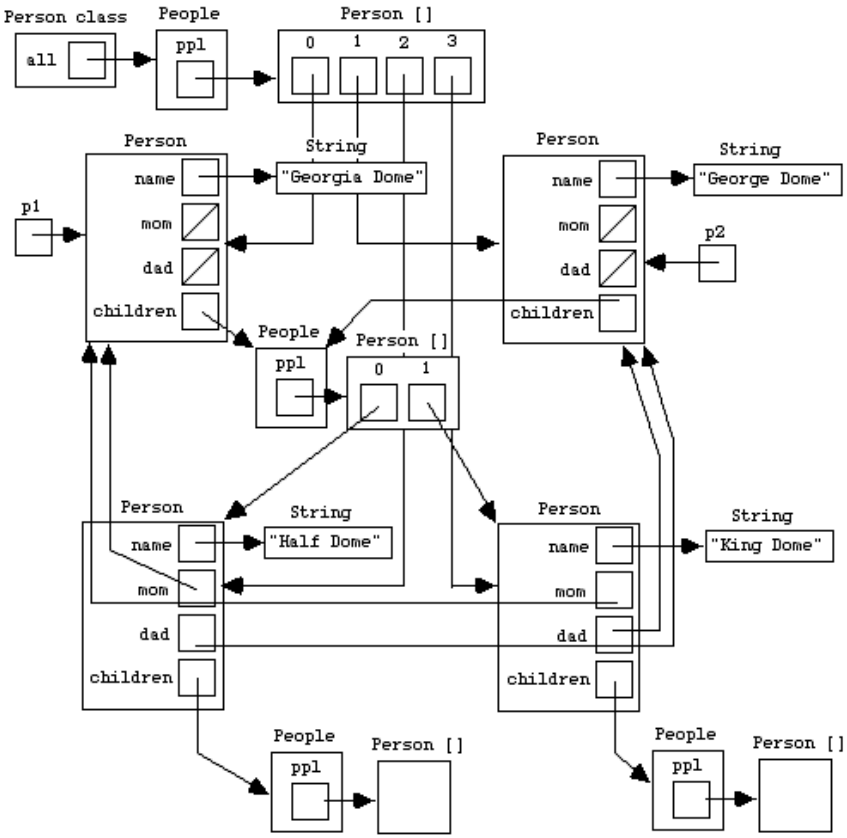
	Diagram		Note
구조성	클래스	Class	비슷한 속성, 공통적인 행동 수단을 지닌 것들의 범주 혹은 그룹
	객체	Object	클래스의 인스턴스 즉, 값이 매겨진 속성과 행동을 가지고 있는 개별적인 개체
	컴포넌트	Component	컴퓨터 시스템을 명확하게 나타내기 위한 목적
	배치	Deployment	컴퓨터를 기반으로 하는 시스템의 물리적 구조를 나타냄
	배포	Package	내부에 모델 요소를 포함할 수 있는 패키지 구성을 나타냄
행동성	유스케이스	Use case	사용자의 입장에서 본 시스템의 행동
	상태	Status	객체의 상태 변화를 보여줌
	활동	Activity	객체의 액션 흐름을 플로우 차트처럼 나타냄
작용성	시퀀스	Sequence	객체끼리 주고받는 메시지의 순서를 시간의 흐름에 따라 보여줌
	통신/협력	Communication	시퀀스 다이어그램 처럼 메시지 흐름을 파악할 수 있지만 시간의 흐름이 아닌 객체에 주목, 객체간의 관계도 표현한다.

1) Class Diagram



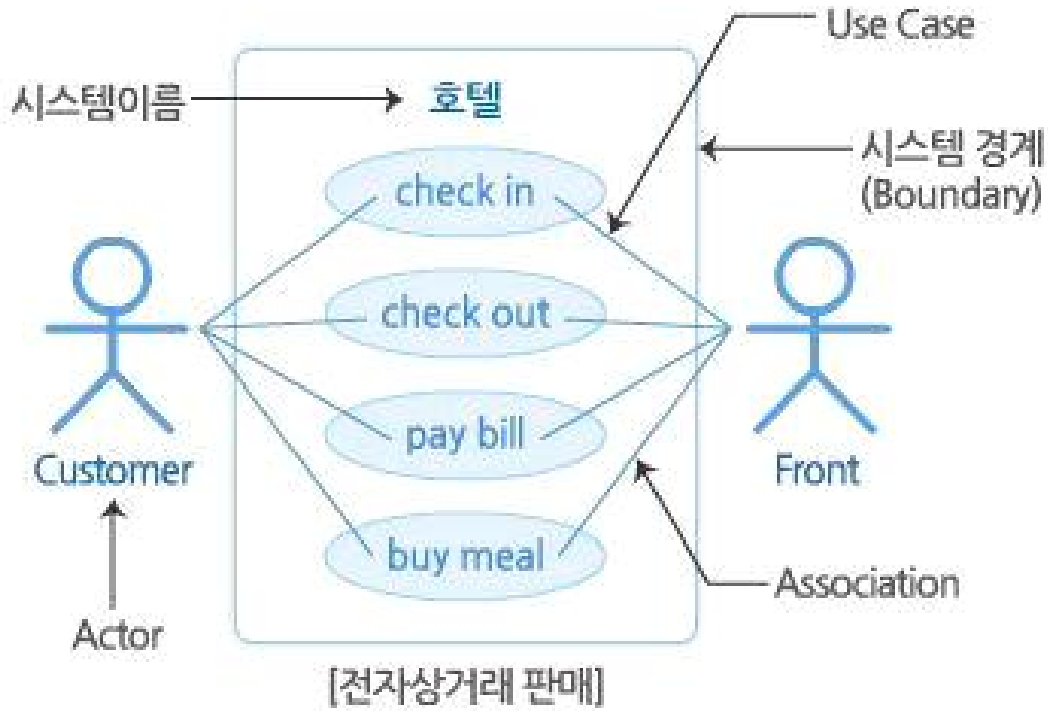
클래스 다이어그램(Class Diagram)은 클래스관련 요소들의 여러 가지 정적인 관계를 시각적으로 표현한 것이다. 클래스 다이어그램은 클래스(Class) 뿐만 아니라 인터페이스(Interface), 열거형(Enumeration), 패키지(Package) 및 여러 가지 관계들뿐만 아니라 인스턴스(Instance)와 그것들의 연결(Link) 등도 포함할 수 있다.

2) Object Diagram



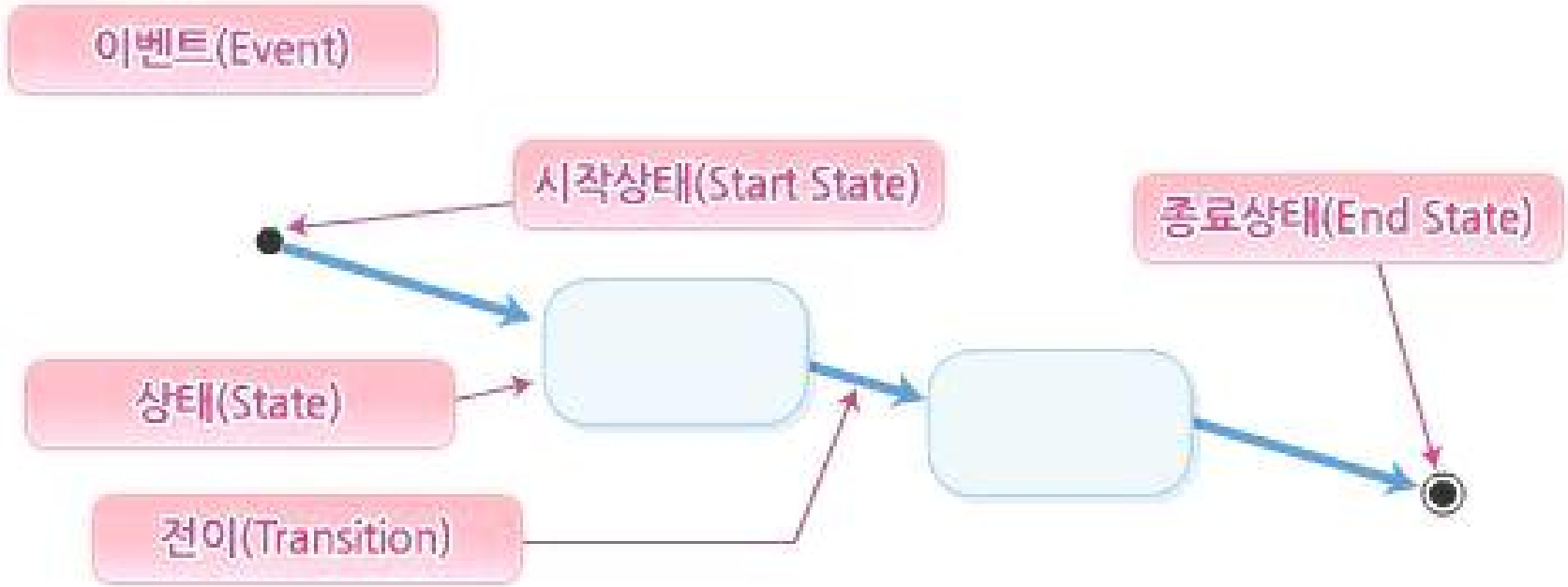
객체(Object)란 클래스의 Instance 즉, 값이 매겨진 속성과 행동을 가지고 있는 개별적인 객체를 일컫는다.

3) UseCase Diagram



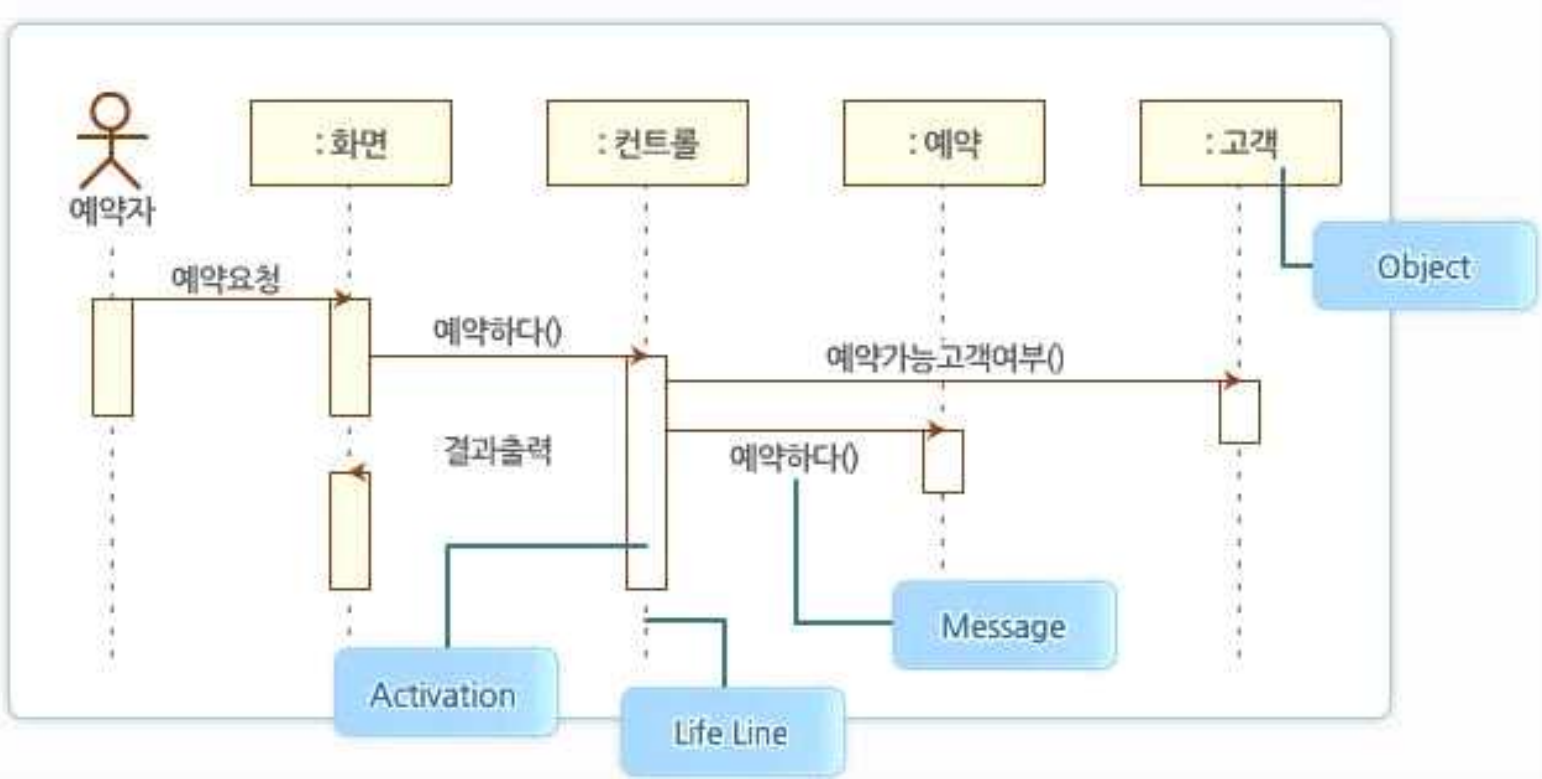
유스케이스 다이어그램(Use Case Diagram)은 특정 시스템 혹은 개체내의 유스케이스(Use Case)들과 그 외부의 액터(Actor)들 간의 관계를 표현한 것이다. 유스케이스는 해당 시스템의 기능을 표현하며 그것들이 어떤 외부 액터들과 상호작용하는지를 나타낸다.

4) State Diagram



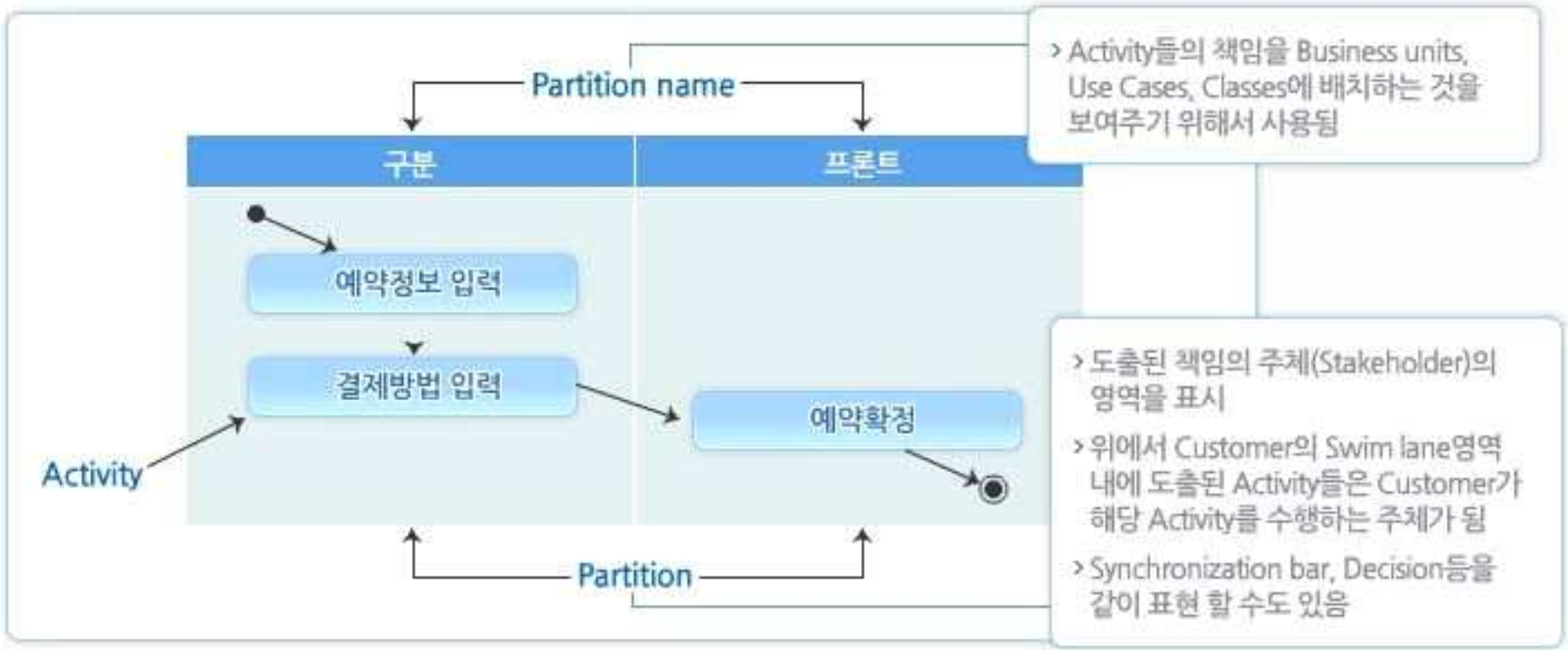
상태 다이어그램(Statechart Diagram)은 특정 개체의 동적인 행위를 상태(State)와 그것들 간의 전이(Transition)를 통해 묘사한다. 일반적으로 클래스의 인스턴스에 대한 행위를 묘사하는데 사용되지만 그 밖의 요소들에 대해서도 얼마든지 사용될 수 있다.

5) Sequence Diagram



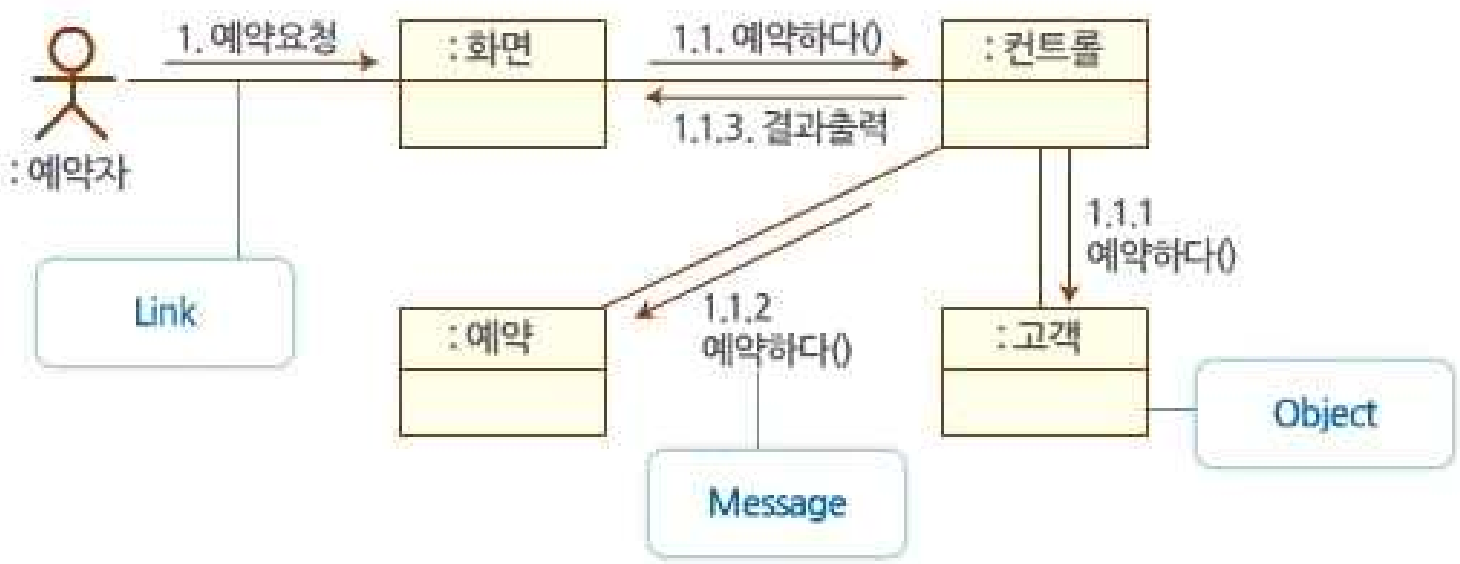
클래스 다이어그램과 객체 다이어그램이 정적인 정보를 나타낸다면, 시퀀스 다이어그램은 동적인 정보를 나타낸다고 할 수 있다. 즉, 여러 객체들이 서로 메시지를 주고 받으며 작업을 진행 하는데 Sequence Diagram은 객체들의 주고 받는 메시지 순서를 시간의 흐름에 따라 보여주는 그림이다.

6) Activity Diagram



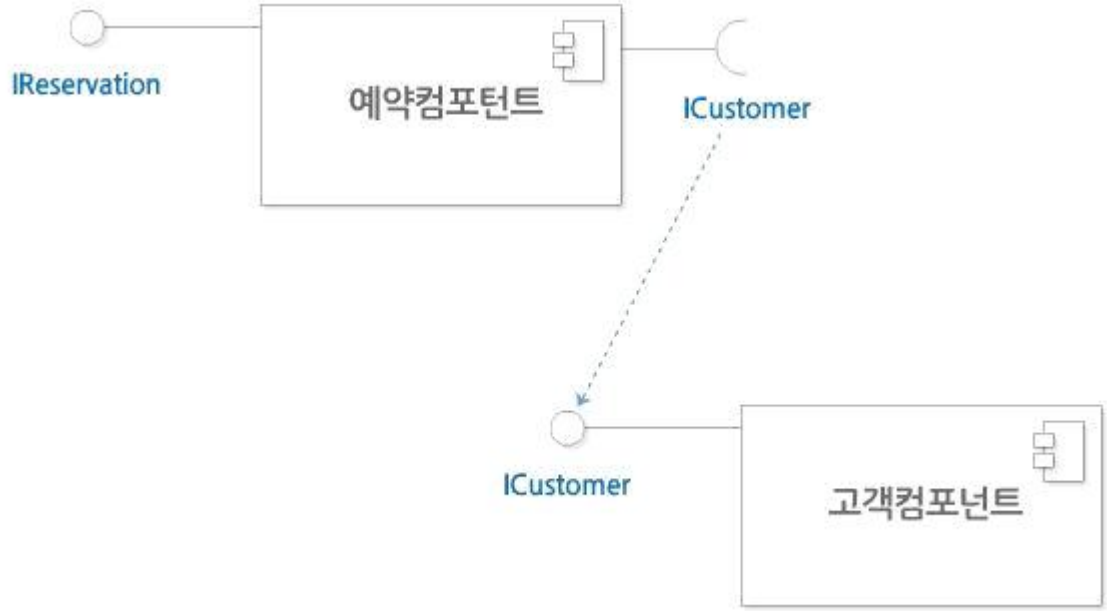
Activity Diagram은 Activity를 처리하는 동안 두 개 이상의 클래스 객체들 간 제어 흐름을 보여준다. Activity 다이어그램은 비즈니스 단위 레벨에서 상위 레벨의 비즈니스 프로세스를 모델링 하거나 저 수준 내부 클래스 액션을 모델링 하는데 사용된다.

7) Comunication Diagram



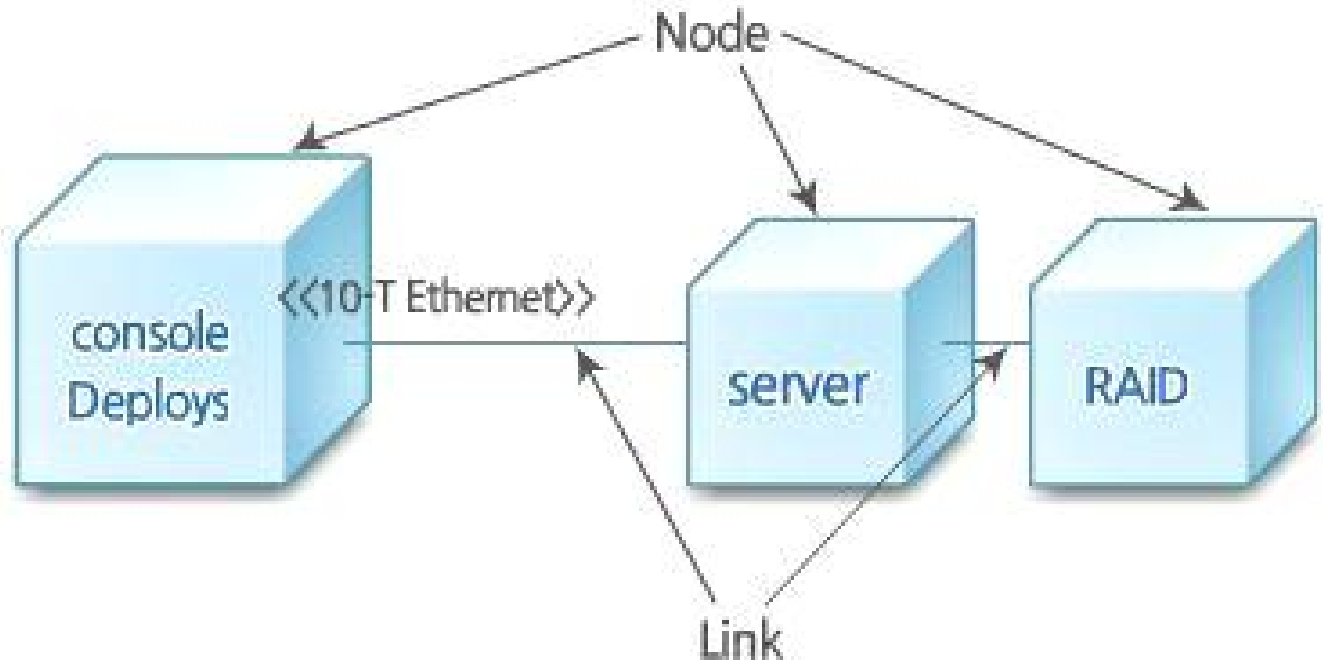
통신 다이어그램은 클래스 배열에서 얻은 정보의 조합을 나타내며 유스 케이스 다이어그램 정적 구조와 시스템의 동적 거동을 모두 기술한다.

8) Component Diagram



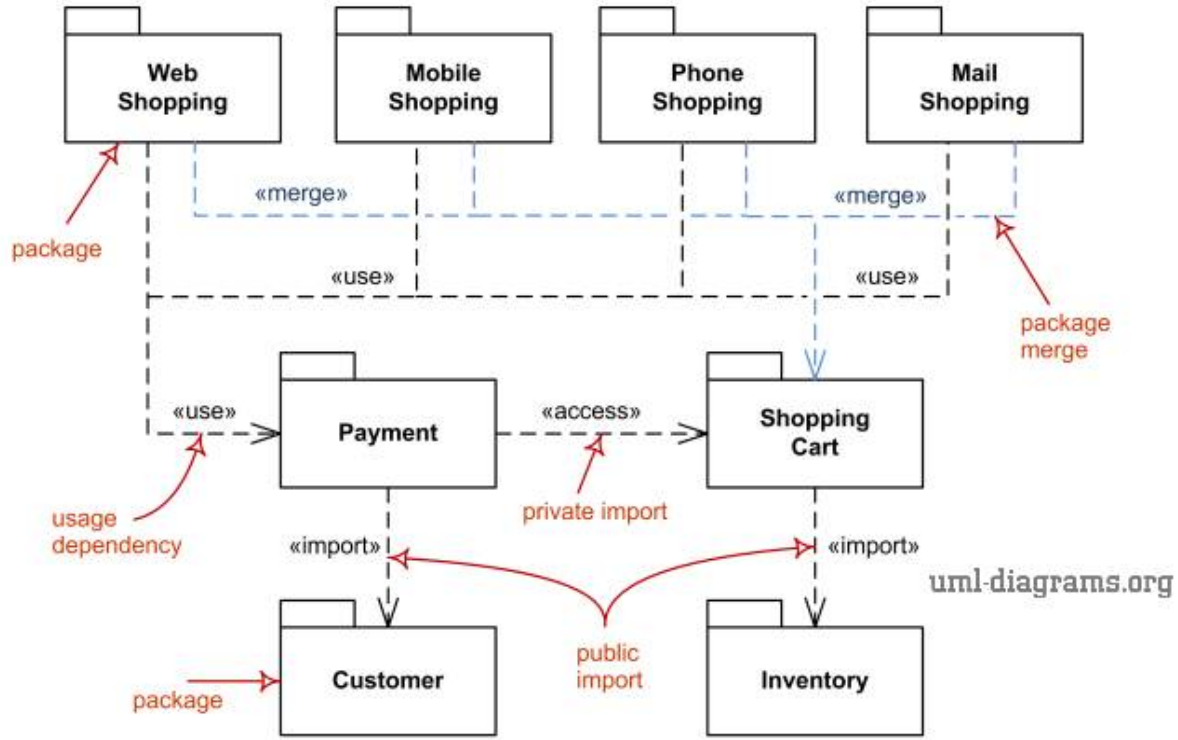
Component Diagram은 시스템을 물리적으로 볼 수 있도록 합니다. 이것의 목적은 소프트웨어가 시스템의 다른 소프트웨어 Component들(예를 들어, 소프트웨어 라이브러리)에 대해 소프트웨어가 갖고 있는 종속 관계를 보여주는 것 입니다.

9) Deployment Diagram



배치 다이어그램은 시스템을 구성하는 하드웨어간의 연결관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치상태를 표현한 다이어그램이다.

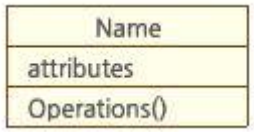
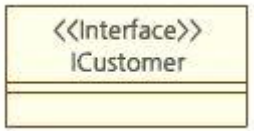
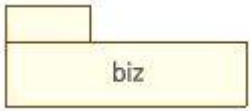
10) Package Diagram



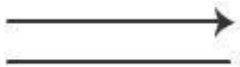
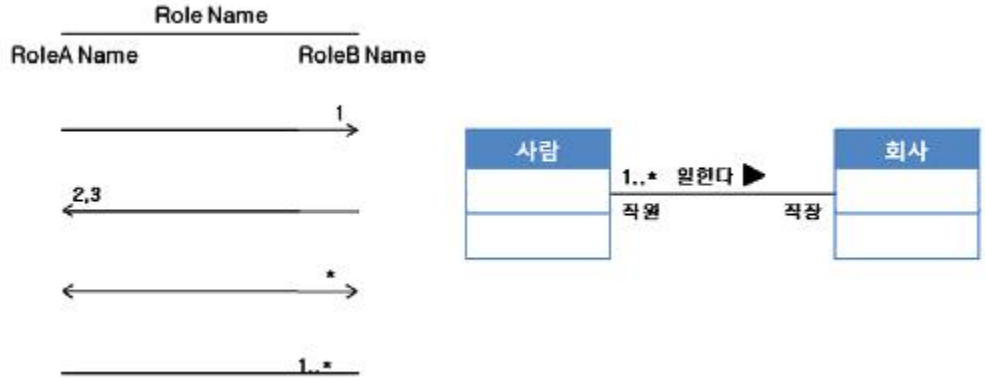
시스템을 이해하기 위한 목적으로 추상적인 개념들을 모은 하나의 그룹을 패키지라고 한다.
패키지는 요소들을 그룹으로 조직하기 위한 범용 메커니즘으로 모델의 요소들을 조직하고 이해할 수 있도록 해준다. 패키지에 담기는 것은 비단 클래스에만 국한되는 것은 아니며, 유스케이스, 활동다이어그램등과 같은 것들도 담을 수 있고, 다른 패키지들도 담을 수 있다.

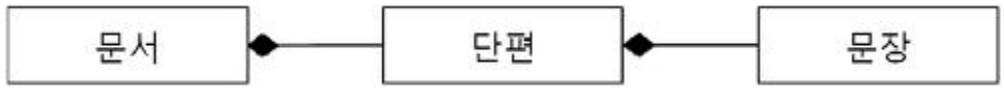
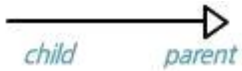
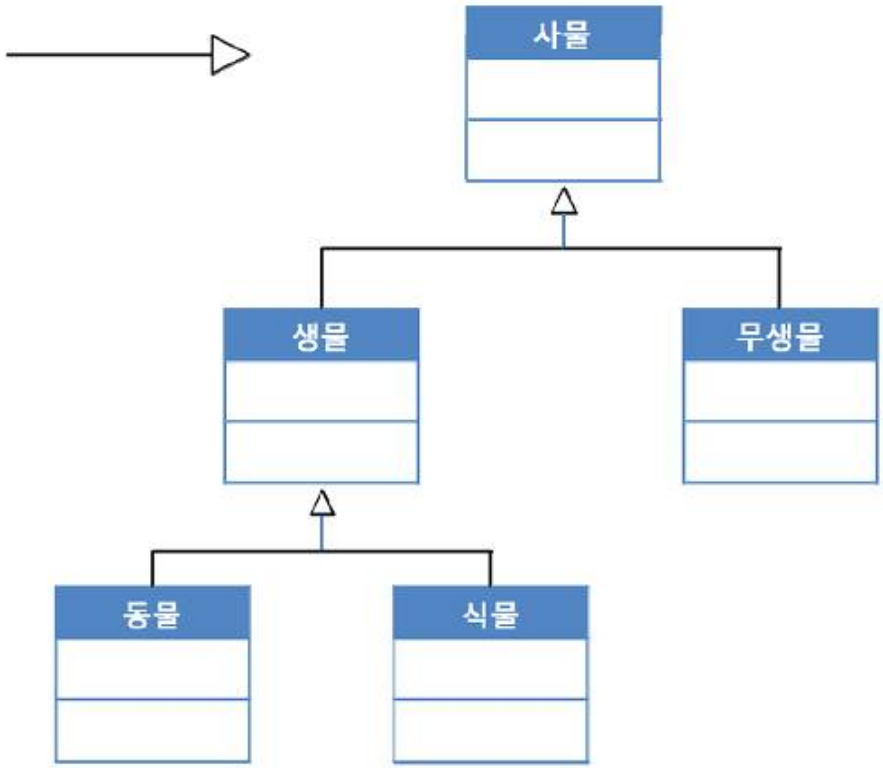
3. UML 표기법

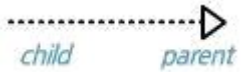
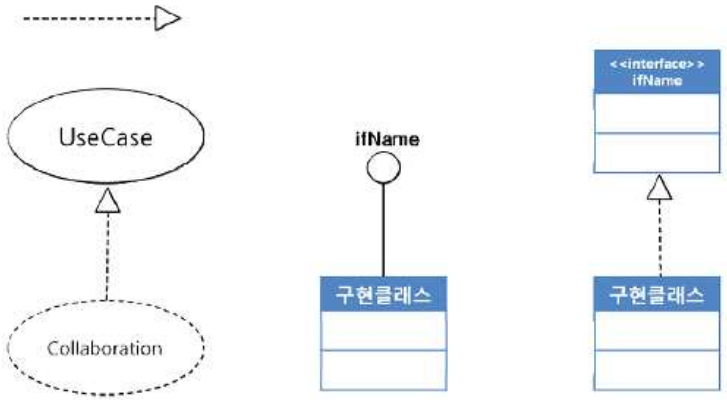
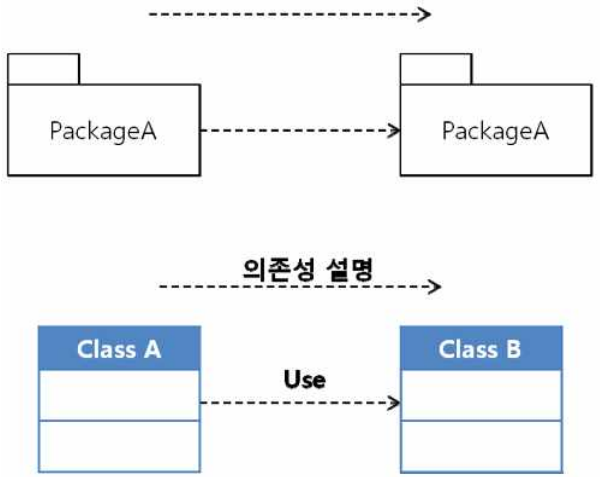
1) Class Diagram

Name	표기법	설 명	적 용
Class		해당 도메인의 관점에서 객체들을 추상화하여 이름과 특징들 및 행위를 부여한 것	Class Diagram
Interface		지켜야 할 약속이나 규약을 제시하는 불완전한 Operation들을 갖는 표현	Class Diagram
Package		Class들 혹은 Interface등이 모여서 이루어 지는 집합적 표현 Package를 이용하여 Component를 표현하기도 함	Class Diagram

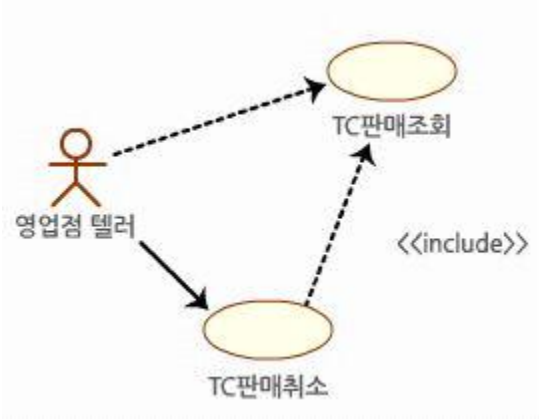
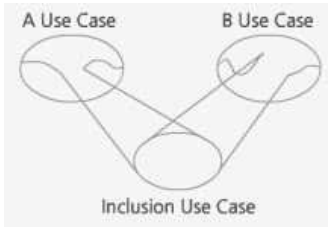
– Class Diagram을 위한 Relation Ships 표기법

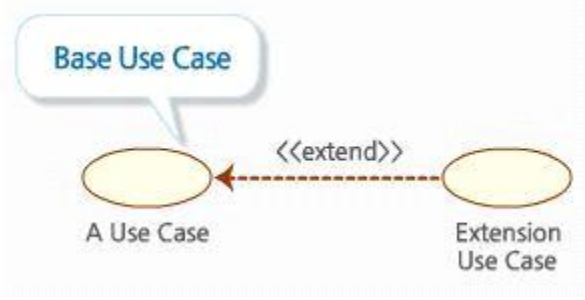
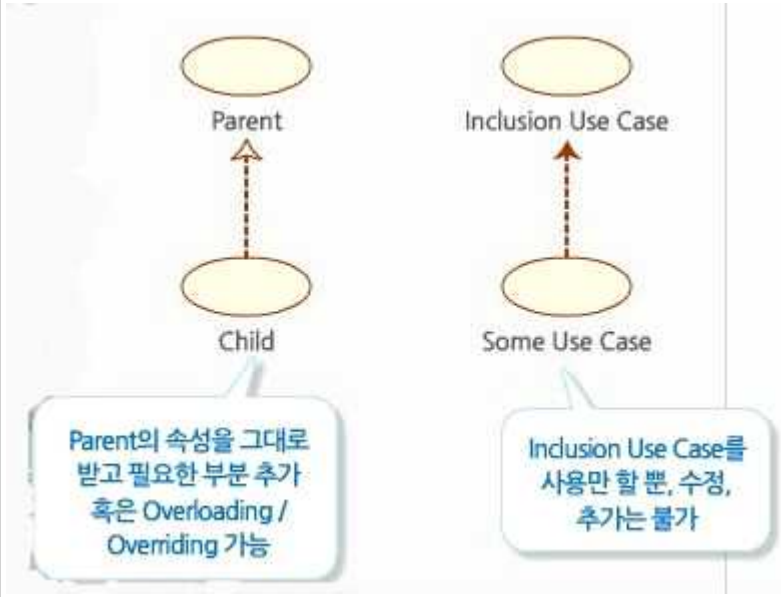
Name	표기법	설 명
Association		방향성에 따라 한 쪽 Class에서 다른 쪽의 Class의 Type에 해당하는 Attribute를 가짐 
Aggregation		Association의 특수한(전체와 부분) 관계로서 부분이 전체와 생사를 같이 하지 않음 

Composition	whole part	
Generalization		부모의 Attributes나 Operations를 자식이 물려 받음(상속)을 표현 

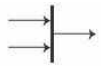
Realization		자식(Class)이 부모(Interface)의 불완전한 Operations들을 반드시 구체적으로 구현함을 표현 
Dependency		클래스 혹은 Package들 간에 참조(Local Variable 형태)를 표현 

2) Use Case Diagram

Name	표기법	설 명	적 용
Actor		Use Case를 이용하는 사람이나 시스템 혹은 장치	Use Case Diagram
Use Case		시작과 끝이 존재하며, Actor에게 유의미한 결과를 주는 프로세스	Use Case Diagram
Association		Actor와 Use Case간의 연관을 표현 (Actor가 Use Case를 사용)	Use Case Diagram
Note		추가적인 설명을 위해 사용	공통
Include		두 개 이상의 다른 Use Case에서 공통적으로 나타나는 Events Flow를 별도의 Use Case로 도출 	

Extend		하나의 완성된 Use Case(Base Use Case)에서 Optional하게 발생할 수 있는 Events Flow를 별도의 Use Case로 도출 (애프터 서비스 개념)	
Generalize		부모 Use Case가 가진 모든 속성을 자식 Use Case가 그대로 물려받고 부가적인 속성을 추가로 가짐	

3) State/Activity Diagram

Name	표기법	설명	적용
Start		프로세스의 시작을 표현	State Diagram Activity Diagram
End		프로세스의 끝을 표현	State Diagram Activity Diagram
State		한 객체의 생명주기 중에서 특정 Attribute의 변화에 따라 특정 Operation 만을 수행할 수 있는 시간적인 존재상태를 표시	State Diagram Activity Diagram
Activity		행위 혹은 행위들의 모임을 표현	Activity Diagram
Transition		Transition은 State 또는 Activity 사이의 변화를 나타냄 일반적으로 이벤트의 변화를 표현	State Diagram Activity Diagram
Decision Activity		조건의 분기를 표현	Activity Diagram
Synchronization bar		프로세스의 병합	Activity Diagram

4. GRAPPLE에 따른 UML 작성 과정

GRAPPLE(Guidelines for Rapisd APPLication Engineering)의 진행 영역

- 요구사항 수집(Requirement Gathering)
- 분석(Analysis)
- 설계(Design)
- 개발(Developement)
- 배포(Deployment)

1) 요구사항 수집

가장 중요한 영역이라고 할 수 있다. 의뢰인의 도메인을 이해해야 제대로된 개발을 할 수 있다.

업무과정 파악

- 의뢰인의 업무가 어떻게 진행되는지 파악.
- 의뢰인의 도메인(기본지식, 사고방식)에 맞는 용어 집합 획득. 차후에 이런 용어를 사용해 준다.
- 결과물 : 해당 업무 과정에서의 단계와 결정 위치(decision points)를 정리하여 나타낸 활동 다이어그램

도메인 분석

- 의뢰인의 도메인을 가능한 더욱 탄탄하게 파악하기 위한 목적. 의뢰인의 모든 것을 파악해야 한다.
- 결과물1 : 추상적인 클래스 다이어그램(명사 - 클래스, 속성; 동사 - 오퍼레이션)
- 결과물2 : 대화과정을 정리한 노트 (필요하다면 녹음테이프)

연동시스템 확인

- 새로운 시스템이 의존할 시스템과 이 시스템에 잘 맞물리는 시스템을 검색.
- 결과물 : 배포 다이어그램(노드, 노드간 접속, 상주 컴포넌트, 컴포넌트간 의존관계)

시스템 요구 사항 파악

- 개발팀이 공동 애플리케이션 개발(Joint Application Developement, JAD) 모임 소집.
- '의사 결정권자와 사용자들의 요구사항'에 대한 개발자들의 의견을 이끌어냄.
- 결과물1 : 패키지 다이어그램(추상적인 시스템 기능)
- 결과물2 : 패키지 다이어그램 내의 유스 케이스
- 결과물3 : 대화과정을 정리한 노트
- 결과물4 : 더욱 정리된 클래스 다이어그램

요구사항 수집 결과를 가지고 의뢰인과 의사 교환
결과물은 다음과 같이 다양할 수 있다. 업체마다 다름.
- 의뢰인의 승인 요구
- 요구사항 수집 결과에 대한 비용 산정

2) 분석
"요구사항 수집" 영역의 결과를 가지고 문제의 이해도를 높인다.
이 영역은 "요구사항 수집" 중에 시작되는데 JAD 모임에서 클래스 다이어그램을 손질하기 시작하는 시점이 바로 이 진행 영역에 해당된다.

시스템 사용법의 이해
- 추상 수준의 유스케이스 분석 영역.
- JAD 모임에서 유스케이스의 시작 행위자(사람 혹은 시스템)와 결과받는 행위자를 알아낸다.
- 결과물 : 행위자와 유스케이스 간에 스테레오 타입을 가진 의존 관계(<<extend>>, <<include>>)를 나타내는 유스케이스 다이어그램.

유스케이스에 살 붙이기
- 개발팀이 사용자와 작업을 계속해 유스케이스를 구성하는 각 단계의 순서 분석.
- JAD 모임 다시 열어 앞서의 모임을 발전시킴. (사용자에게 usecase를 설명해야 함)
- 결과물 : 유스케이스 내의 진행 단계를 적어 둔 텍스트.

클래스 다이어그램 손질
- JAD 모임 진행되는 동안 클래스 다이어그램 수정.
- 연관, 추상 클래스, 다중성, 일반화, 집합 연관 등의 내용 적어야 함.
- 결과물 : 완성된 클래스 다이어그램.

객체의 상태 변화 분석
- 필요할 때마다 상태 변화를 나타냄으로써 모델을 자세히 만든다.
- 결과물 : 상태 다이어그램.

객체간 교류 정의
- 결과물 : 시퀀스 다이어그램과 협력 다이어그램.

연동 시스템과의 통합 분석
- 통신 접속 방법, 네트워크 구조, 데이터 베이스 구조 등을 결정.
- 결과물 : 상세한 배포 다이어그램과 데이터 모델.

3) 설계
솔루션을 설계한다. "분석"과 "설계" 영역은 설계 완료시까지 자유롭게 오갈 수 있다.

객체 다이어그램의 개발과 손질
- 클래스 다이어그램을 가지고 객체 다이어그램을 그린다.
- 활동 다이어그램은 "개발" 진행 영역에서 대부분의 코딩 작업 기반을 제공.
- 결과물1 : 객체 다이어그램
- 결과물2 : 활동 다이어그램

컴포넌트 다이어그램의 개발
- 컴포넌트를 시각화하고 의존 관계를 표시.
- 결과물 : 컴포넌트 다이어그램.

배포 계획

- 시스템 배포와 연동 시스템과의 통합을 계획.
- 배포 다이어그램에 컴포넌트를 적절히 배포해 그려 넣는다.
- 결과물 : 배포 다이어그램의 일부가 될 부분 다이어그램.

사용자 인터페이스의 설계와 원형 정의

- 사용자와의 JAD 모임을 연다.
- 모든 유스케이스를 고려하여 사용자 인터페이스를 만든다.
- 유스케이스에 맞는 화면 프로토타입을 종이에 그려 낸다.
- 화면 컴포넌트(푸시 버튼, 체크 박스 등)을 나타내는 포스트잇 노드를 원하는 위치에 붙인다.
- 위의 과정을 통해 사용자 인터페이스의 원형을 잡는다.
- 결과물 : 화면 원형을 잡은 스크린샷.

시험 설계

- 유스 케이스 다이어그램을 사용하여 시험용 스크립트나 자동화된 시험 도구를 개발한다.
- 결과물 : 시험 스크립트.

문서화 시작
- 최종 사용자나 시스템 관리자를 위한 시스템 설명서를 작성한다.
- 스토리 보드를 작성하며 각 문서의 골격을 만든다.
- 결과물 : 문서의 골격

4) 개발
코드 작성
- 클래스 다이어그램, 객체 다이어그램, 활동 다이어그램, 컴포넌트 다이어그램을 가지고 코딩 시작.
- 결과물 : 코드.

코드 시험
- 시험 전문가가 코드가 제대로 작동하는지 여부를 시험 스크립트를 사용해 평가.
- 시험을 통과할 때까지 "코드 작성" 단계에 이것을 반영(피드백).
- 결과물 : 시험 작업물.

사용자 인터페이스의 구축과 코드의 연결 및 시험
- 사용자 인터페이스를 코드에 연결.
- 이것이 제대로 동작하는 지 검사.
- 결과물 : 사용자 인터페이스까지 붙은 소프트웨어.

문서화 완료
- 소프트웨어의 개발과 동시에 완료.
- 결과물 : 시스템에 관련된 설명서 혹은 문서.

5) 배포
적절한 하드웨어에 설치해서 다른 시스템과 연동하는 일이 남았다. "배포" 진행 영역은 "개발" 진행 영역이 시작되기 전에 이루어진다.

백업과 복구에 대한 계획
- 시스템 충돌시 취할 조치 계획.
- 백업/복구 계획에는 시스템을 백업하고, 충돌 사고치 복구 조치가 명시되어 있다.
- 결과물 : 백업/복구 계획 문서.

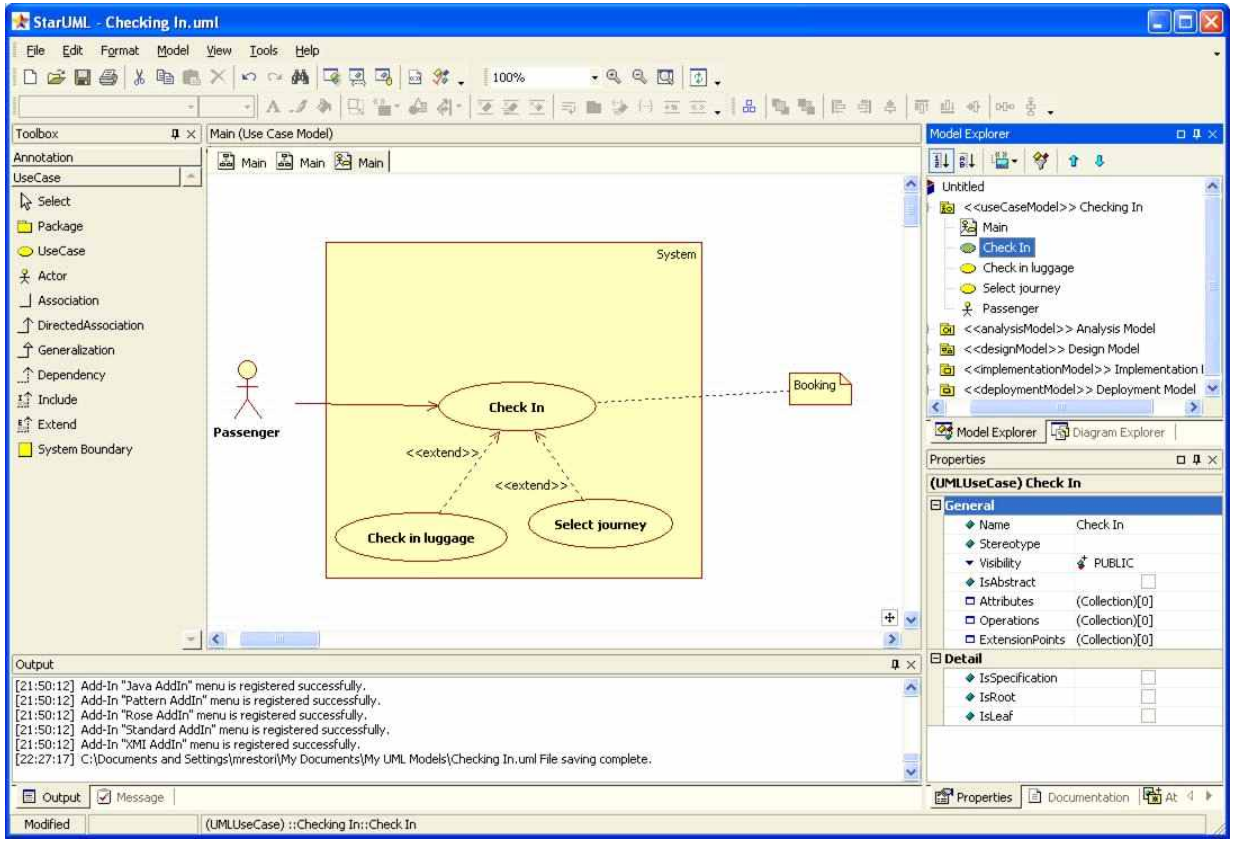
적합한 하드웨어에 최종 시스템 설치하기
– (필요한 경우) 프로그래머의 도움을 받아 시스템 설치.
– 결과물 : 완전히 설치되어 연동되는 시스템.

설치된 시스템의 시험
[list]– 시스템이 처음에 의도한 대로 동작하는 지를 검사.
– 백업/복구 조치가 제대로 이루어지는 지를 검사.

발표
– 발표
– 작업 결과물은 개발팀이 알아서 챙겨 모은다.

6) 정리
GRAPPLE은 일반적(general) 과정에서 세부적(specific) 과정으로 이동한다.
도메인의 개념적인 이해에서 출발해, 추상적 기능을 정의하고, 각 유스케이스의 시나리오를 작성하며, 모델과 설계 사항을 손질하고, 시스템을 개발하고, 배포한다.

5. UML Tools – StarUML



<StarUml의 인터페이스>

StarUML은 UML 모델링 도구로 Use Case Diagram, Class Diagram, Sequence Diagram 등 다양한 Diagram을 간편한 UI를 이용하여 쉽고 빠르고, 유연하고, 확장가능하며, 풍부한 기능에 Win32 플랫폼에서 무료로 사용할 수 있는 UML/MDA 플랫폼(툴)을 개발하기 위한 오픈 소스 프로젝트입니다.
StarUML 프로젝트의 목적은 Rational Rose, Together와 같은 상업적 도구를 비싼 돈을 들여 사용하지 않더라도 그에 준하는 기능을 갖춘 오픈 소스 소프트웨어 모델링 도구 및 플랫폼을 개발하는 것입니다.

StarUML의 특징으로는 도구를 사용하기 위한 학습시간이 짧아 초보자도 쉽게 사용할 수 있습니다.
유료소프트웨어 못지 않은 다양한 UML 작성 기능을 가지고 있습니다.
Reverse Engineering을 지원합니다.
UML 2.0 지원
MDA (Model Driven Architecture) 지원
누구든지 COM과 호환가능한 언어(C++, Delphi, C#, VB 등)에서 플러그인 모듈을 개발할 수 있게 단순하며 강력한 플러그인 아키텍쳐를 제공합니다.
퀵 다이얼로그, 키보드 조작, 다이어그램 오버뷰 등과 같이 많은 사용자들에게 친숙한 특징을 제공할 수 있도록 적용되었습니다.

6. 참조문헌

Unified Modeling Language™ (UML®) – http://www.uml.org/

Wikipedia UML 항목 - http://en.wikipedia.org/wiki/Unified_Modeling_Language

examines the Unified Modeling Language.SimVentions, Inc.
– http://www.simventions.com/whitepapers/uml/3000_borcon_uml.html

개발이 하고싶어요.Tistory Blog - http://hyeonstorage.tistory.com/13
http://hyeonstorage.tistory.com/14
http://hyeonstorage.tistory.com/15

창조 프로젝트.Tistory Blog – http://crproj.tistory.com/5

SIGI 의 Programming Note.Tistory Blog - http://prosigi.tistory.com/141

그냥 그런 블로그.Naver Blog - http://blog.naver.com/lifeisforu/80022419859

StarUML – http://staruml.sourceforge.net/ko/

StarUML Manual.한국소프트웨어진흥원 – http://staruml.sourceforge.net/ko/