UML Report

What is UML?

How to use UML Tools?

On 22 Mar 2013

Team Organization– T4

Kim, Sang Yoon	200811411	gdzergling@core-a.org
Oh, Na Yun	200814189	brilliantjay@naver.com
Lim, Min Woo	200910793	dn3108@gmail.com

Contents

1. Introduction

1.1. What is UML?

- UML(Unified Modeling Language: 통합 모델링 언어)는 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어이다. 이 표준은 UML을 고안한 객체 관리 그룹(OMG: Object Management Group)에서 관리하고 있다. UML은 소프트웨어 집약 시스템의 시각적 모델을 만들기 위한 도안 표기법을 포함한다.

- UML은 객체 지향 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화할 때 사용한다. UML은 아래와 같은 사항을 포함하여 시스템의 구조적 청사진을 시각화 하는 표준안을 제공한다.

  - ◆ 행위자(UML)
  - ◆ 비즈니스 프로세스
  - ◆ (논리적) 부품 (UML)
  - ◆ 행위 (UML)
  - ◆ 프로그래밍 언어 구문
  - ◆ 데이터베이스 스키마
  - ◆ 재사용할 수 있는 소프트웨어 부품

- UML은 데이터 모델링(개체-관계 다이어그램)과 비즈니스 모델링(업무 흐름), 객체 모델링, 부품 모델링의 최선의 기술을 조합한다. UML은 소프트웨어 개발 공정뿐만 아니라 다른 구현 기술의 모든 공정에서 사용될 수 있다. UML은 Booch 방법론의 객체 모델링 기법(OMT)와 객체 지향 소프트웨어 공학(OOSE)을 광범위하게 사용할 수 있는 단일한 공통 모델링 언어로 통합한다. UML의 목표는 동시적 분산 시스템을 모델링 하는 표준 언어다. UML은 산업의 실질적 표준으로서, 객체 관리 그룹(OMG)에 의해 개선되고 있다. 초기에 OMG가 엄격한 소프트웨어 모델링 언어를 만들기 위해 객체 지향 방법론적인 통지를 요청했고, 많은 산업 선구자가 UML 표준 제작을 돕기위해 진지하게 응답하였다.
  UML 모델은 객체 관리 그룹이 지원하는 QVT와 같은 변환 언어 등을 이용해 다른 표현(예를 들면 자바)으로 자동적으로 변환된다. UML은 확장할 수 있으며 커스터마이제이션을 위한 메커니즘인 프로파일 (UML), 스테레오타입 (UML)을 제공한다. 프로파일을 이용한 확장의 의미는 UML 2.0에서 개선되었다.

1.2. What are UML Components?

1.2.1. 소프트웨어 개발 방법론

- UML 그 자체는 개발 방법이 아니지만 그 당시 주도적이었던 객체 지향 소프트웨어 개발 방법론(예를 들면 Booch 방법론, 객체 모델링 기법, Objectory)과 잘 어울리도록 설계되었다. UML 발전해 감에 따라서 UML의 장점을 취하기 위해 몇몇 다른 방법론(예를 들면 객체 모델링 기법)이 개선되었다. 또 UML을 기반으로 한 새 방법론이 만들어지기도 했는데 IBM 래셔널 통합 프로세스(RUP)가 가장 유명하다. 이 외에도 추상 방법론(Abstraction Method), 동적 시스템 개발 방법론 등 더 특수한 해결책이나 다른 목적을 달성하기 위해 설계된 UML 기반의 방법론이 많이 있다.

1.2.2. 모델(Model)

- 모델은 이학 및 공학 분야에서 상당히 유용하게 쓰이는 개념으로서 가장 일반적인 의미로 말하면, "모델은 만들다"는 것은 잘 모르고 있는 것을 이해하는데 도움이 될 것으로 추측되는 어떤 것을 사용한다는 뜻이다. 어떤 분야에서는 이 모델의 일련의 수식(방정식)의 집합으로 정의되기도 하며, 다른 분야에서는 컴퓨터 시뮬레이션을 모델로 삼기도 한다.
  UML의 여러가지 그래픽 요소는 하나의 큰 그림, 즉 다이어그램을 그리는데 사용된다. UML은 언어이기 때문에, 이들 그래픽 요소들을 맞추는 데에는 규칙이 필요하다. 다이어그램의 목적은 시스템을 여러가지 시각에서 볼 수 있는 뷰(View)를 제공하는 것이며, 이러한 뷰의 집합을 모델(Model)이라고 한다. 시스템의 UML 모델은 건물을 짓는 건축가의 스케일 모델과도 비슷하다고 말할 수 있다. UML 모델은 시스템 자체의 “목적 행동”을 설명하는 언어이다. UML 모델은 시스템의 “구현 방법을 설명하는 수단”이 아니다.

1.2.3. 클래스 다이어그램(Class Diagram)

- 대부분의 사물은 자기만의 속성과 일정한 행동 수단을 지니고 있다. 이러한 행동을 오퍼레이션(Operation)의 집합으로 생각할 수 있다. UML에서는 두단어 이상으로 이루어진 클래스 이름은 단어 사이의 공백을 없애고, 각 단어의 처음 문자를 모두 대문자로 한다(예:Wikipedia). 속성과 행동의 이름 또한 마찬가지이지만, 가장 앞 단어의 처음 문자는 소문자로 한다(예:edit()).

1.3. What is UML for?

- UML로 디자인함에 있어서 최우선 목표는 다음과 같다. 사용자에게 즉시 사용가능하고 표현력이 강한 시각적 모델링 언어를 제공함으로써 사용자는 의미있는 모델들을 개발하고 서로 교환할 수 있다. 핵심적이 개념을 확장할 수 있는 확장성과 특수화 방법을 제공한다. 특정

개발 프로세스와 언어에 종속되지 않는다. 모델링 언어를 이해하기 위한 공식적인 기초를 제공한다. 객체 지향 툴 시장의 성장을 장려한다. 콜레버레이션(Collaboration), 프레임워크(Framework), 패턴(Pattern)과 Component와 같은 고수준의 개발 개념을 제공한다.

1.4. History of UML

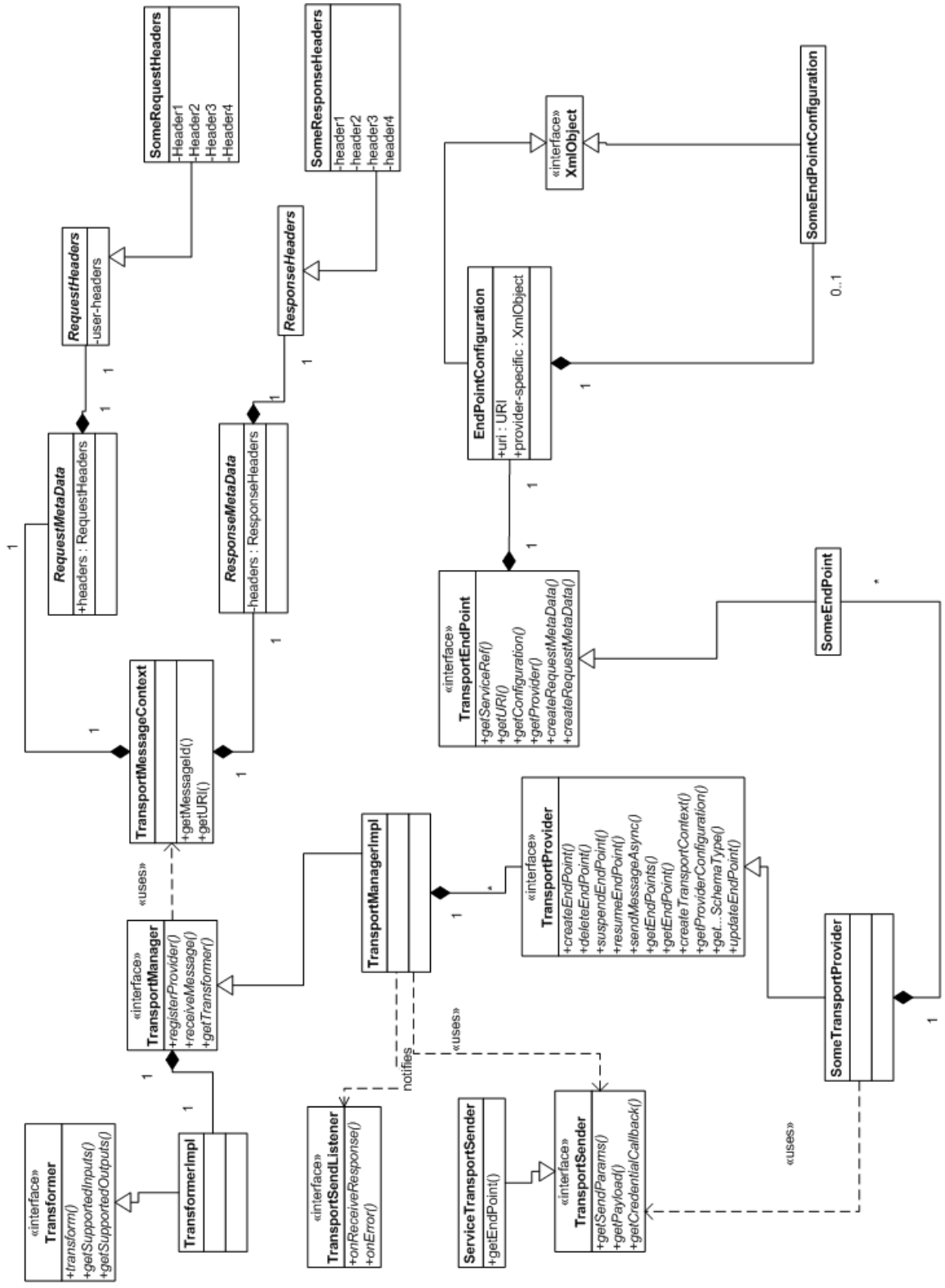
- UML은 그래디 부치(Grady Booch), 제임스 럼버(James Rumbaugh), 이바 야콥슨(Ivar Jacobson)의 머리에서 태어났다. 최근 "쓰리 아미고(Three Amigos-3인방)"이라고 불리는 이 세 사람은 80년대 전반 부터 90년대 초반까지 객체지향 분석 설계 분야에서 각자의 영역에서 방법론을 연구해 왔었다. 그들이 발표한 방법론은 동일한 분야의 다른 경쟁자들보다 항상 탁월한 위치에 있었으며, 세 사람은 90년대 중반에 이르러 각자의 아이디어를 교환하기 시작하였고, 결국 각자의 방법을 하나로 모아 합치기에 이른다.

  1994년, 럼버는 부치가 세운 래셔널 소프트웨어(Rational Software Corporation)에 영입 되었고, 야콥슨은 그로부터 1년 후에 래셔널 사에 들어가게 된다.

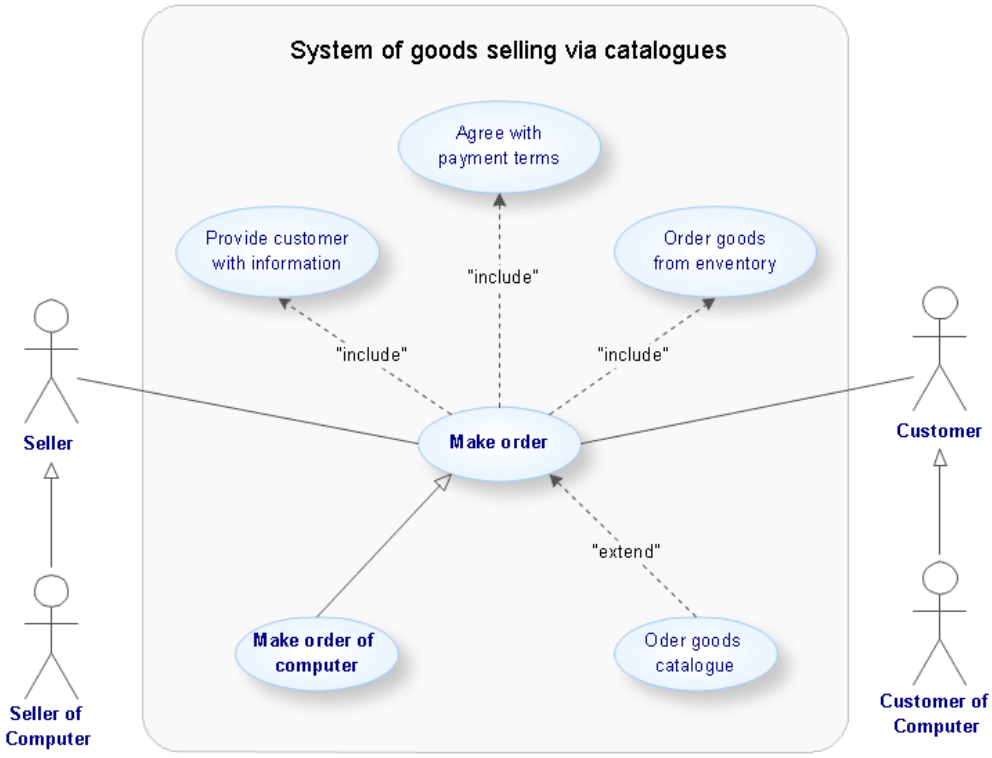
  나머지는 그들이 말하듯이, 역사(History)라고 말할 수 있다. UML의 초안(draft) 버전은 소프트웨어 업계를 뒤흔들기 시작했고, 그 결과로 돌아온 피드백은 바로 변경점에 반영되었다. "UML은 우리들의 전략에 딱 맞는다"라고 인식해 가는 회사가 늘어남에 따라 그 결과로 UML 컨소시엄도 발족하게 되었다. UML 컨소시엄의 멤버로는 디지털(DEC), 휴렛 팩커드(HP), 인텔리캅(Intellicorp), 마이크로소프트, 오라클, 텍사스 인스트루먼트(Texas Instruments), 래셔널 소프트웨어 등이 있었다. 1997년 UML 컨소시엄은 UML 버전 1.0을 만들어 내었고, 오브젝트 매니지먼트 그룹(OMG:Object Management Group)이 표준 모델링 언어의 제안서를 내라는 요구에 맞추어 이것을 제출 하였다.

  UML 컨소시엄은 계속 발전하였으며, OMG에 다시 상정된 UML 1.1d은 1997년 말에 표준 모델링 언어로 채택되었다. OMG는 UML의 관리 기법을 받아들여 1998년에 새로운 수정안을 발표하였다. UML은 소프트 웨어의 업계 명실 상부한 표준이 되었으며, 계속 수정 보안되고 있다. 버전 1.3과 1.4 그리고 1.5가 나와 있고, 최근에는 버전 2.0이 OMG에 의해 승인된 상태이다. 이전 버전들, 즉 버전 1.X는 현존하는 대부분의 모델 및 UML 모델링 책의 기본이 되어 왔다.

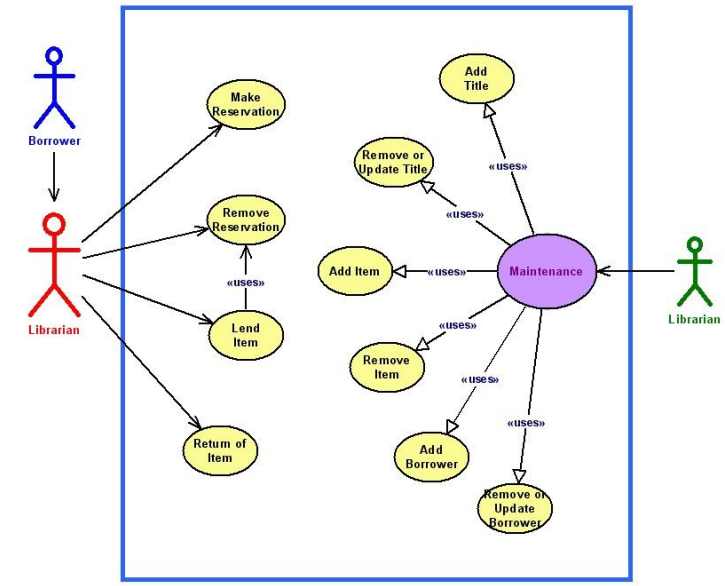
1.5. Example of UML(Unified Modeling Language)



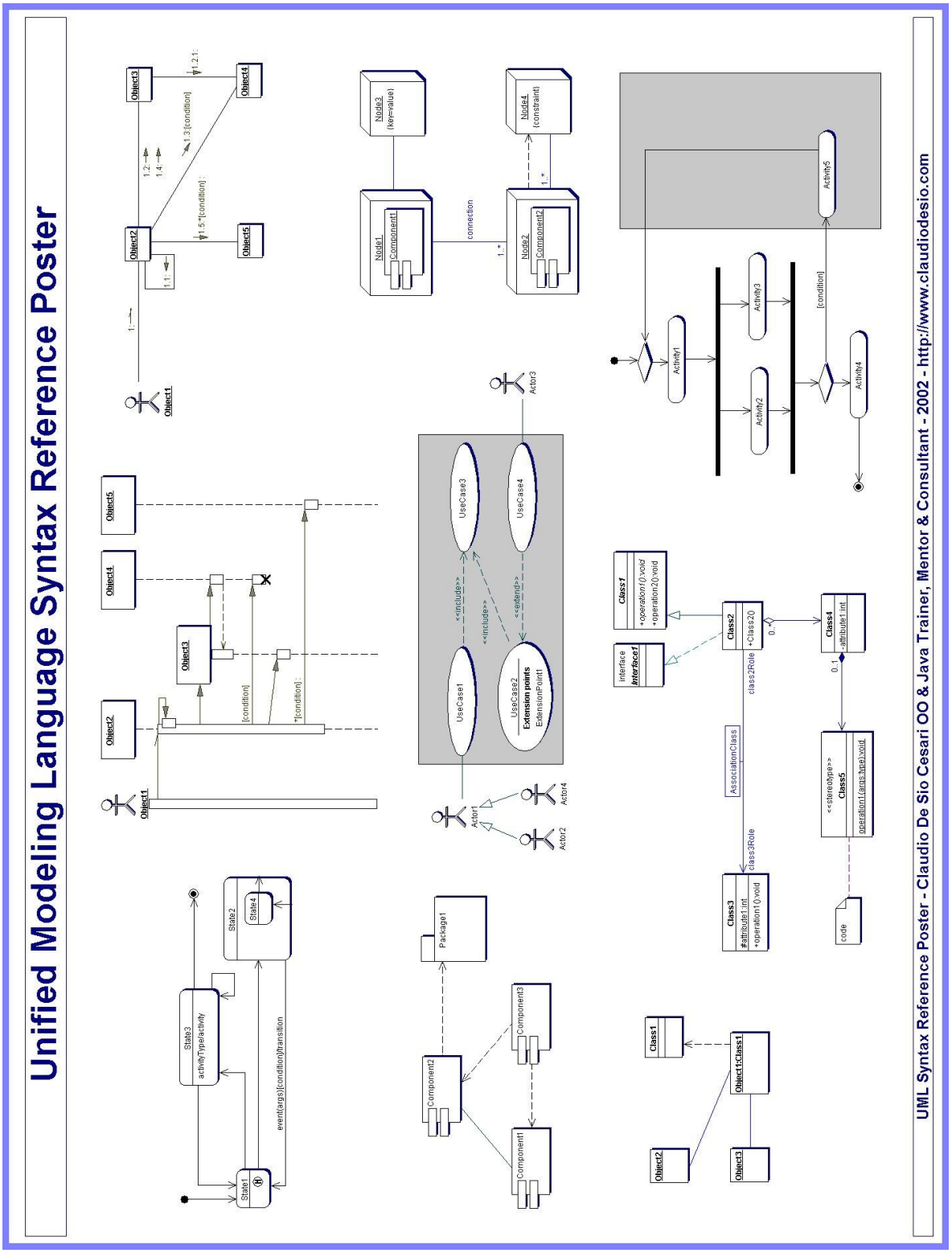
<Figure1. UML Class Diagram>



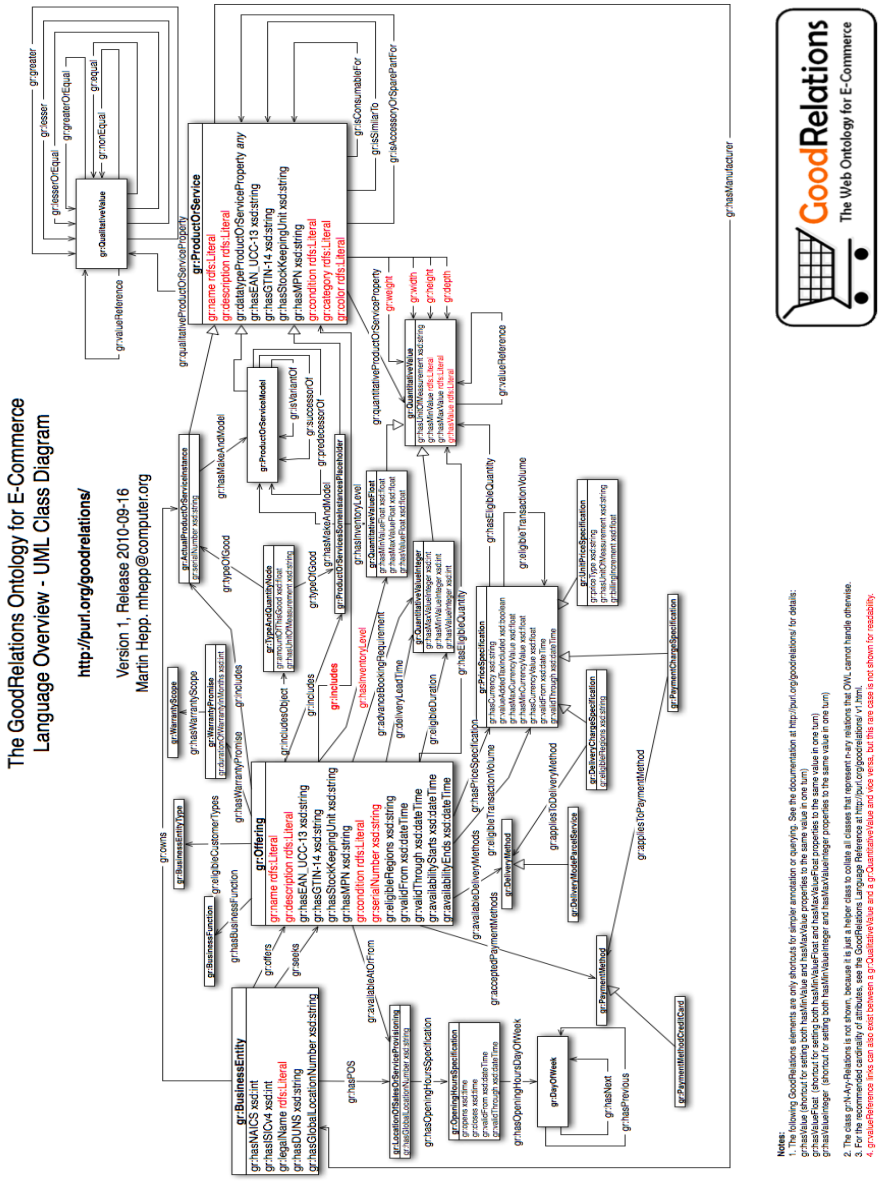
<Figure2. UML Use Case Diagram>



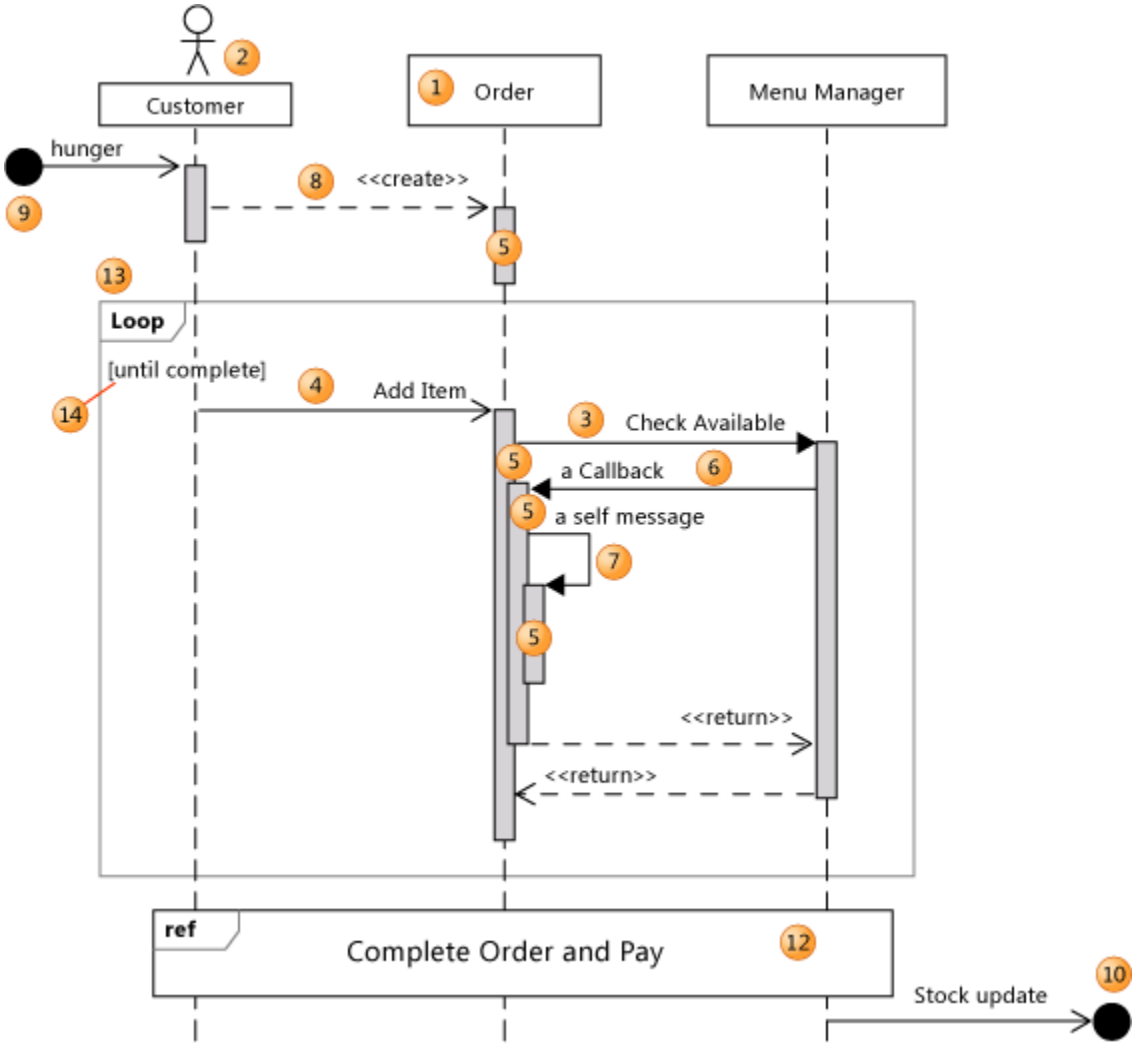
<Figure3. UML Use Case Diagram>



<Figure4. UML Syntax Reference Poster>



<Figure 5. UML Example>



<Figure 6. UML Example>.

2. Use Case

2.1. What is Use Case?

- Use Case는 우리말로 쓰임새라고 한다. Use-Case는 말 그대로 '쓰이는 경우' 혹은 '용도' 같은 의미로 받아들여도 큰 무리가 없다고 보여진다. 즉, 어떤 일에 쓰느냐 하는 것을 의미한다. 시스템이 쓰여지는 용도를 모아서 시스템을 만들어낸다면 다용도 시스템이 만들어진다. 이처럼 Use-Case들을 모아서 시스템으로 연결시키는 것을 개발 과정의 간단한 정의로 보아도 무리가 없을 만큼 Use-Case는 가치 있는 것이다.

    ◆ Use-Case사용자 시각에 맞춘 분석이다.

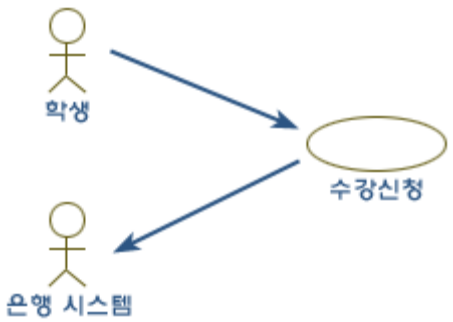
- 어떤 시스템을 만드느냐를 사용자 입장에서 조망하는 것이라고 할 수 있다. Use-Case 시스템 보다는 그것을 사용하는 인간, 즉 사용자의 입장을 우선해서 시스템이 어떠해야 하는가를 알아보는 것이다.

    ◆ Use-Case는 시스템의 행위를 결정하는 것이다.

- 구체적으로는 시스템의 기능을 정의하고, 범위를 결정함으로써 시스템과 외부 환경 변수를 구분하고, 상호 관계를 정립하는 것이라고 볼 수 있습니다.

2.2. What is Use Case Model?

- Use-Case를 나타내는 Use-Case 모델(Model)은 Use-Case Diagram 으로 표현된다. Use-Case 다이어그램은 액터(Actor, 행위자), Use-Case, 그리고 관계(Relationship)로 나타난다.



<온라인 수강신청 시스템 예제>

- 위의 Use-Case 다이어그램은 온라인으로 수강신청을 하는 시스템의 일부를 나타낸 것이다. 위 다이어그램에서 액터는 학생과 은행시스템이고, Use-Case는 수강신청이다. 화살표는 액터와 Use-Case간의 관계를 나타낸 것인데, 위의 예제에서는 단방향으로의 관계만 나타내고 있다. 위의 다이어그램은 학생이 웹에 접속해서 수강신청을 하면, 그 정보가 은행 시스템에 입력이 되는 과정을 나타내고 있다.

- Use Case Model의 기본 구성 요소

  (1) 액터

  시스템과 교류하는 사람이나 사물을 말한다.

  (2) 유스케이스

  시스템이 액터를 위해서 수행하는 가치 있는 일을 말한다.

  (3) 유스케이스 설명

  기본흐름: 사건 흐름에서 가장 중요한 부분으로서 유스케이스 목적을 달성하는 정상적인 방법을 설명한다.

  대안흐름: 기본 흐름에 더하여 다양하고 변칙적인, 그리고 예외적인 경로들을 설명한다.

  서브흐름: 원래 흐름 중에서 따로 떼어낸 것으로서 복잡한 사건 흐름을 읽기 쉽게 만든다. 이는 이름을 붙인 독자적인 미니 흐름으로서, 원래 흐름 자리에는 서브 흐름의 이름을 참조할 수 있도록 삽입하는 것으로 대신한다.

  선조건: 유스케이스를 시작할 당시 시스템과 액터의 상태를 표현

  후조건: 유스케이스를 종료할때 시스템 상태를 표현

2.3. What is Actor?

- 액터는 시스템의 일부가 아니다.액터는 시스템과 상호작용을 하는 모든 것들을 나타낸다. 시스템을 사용하게 될 사람은 물론이고, 연관된 다른 시스템도 액터이다. 액터는 다이어그램 상에서 막대인간으로 표현된다. 대체로 액터의 행위는 정보의 입력과 출력으로 살펴볼 수 있다. 정보를 입력 하거나 출력하는 액터가 있고, 입출력을 모두 행하는 액터가 있을 것

이다. 액터를 뽑아내는 일은 매우 중요한 일입니다. 모든 주요 액터를 고려해야만 모두에게 가치 있는 시스템이 될 수 있다. 다음과 같은 질문들이 요구사항 분석에서 액터를 뽑아내는 데 도움을 준다.

<액터(Actor)의 표기법>

- 특정 요구사항에 이해관계자는 누구인가?
- 어떠한 부서나 집단에서 시스템을 사용하는가?
- 시스템을 사용함으로써 이익을 얻는 이는 누구인가?
- 누가 시스템에 정보를 입력하고 사용하고 삭제하는가?
- 누가 시스템의 유지보수를 수행하는가?
- 시스템이 외부 자원을 사용하는가?
- 한 사람이 복수의 역할을 수행하는가?
- 여러 사람이 한 가지 역할을 수행하는가?
- 시스템이 기존 시스템(legacy system)과 상호작용하는가?

2.4. How to find/write Use Case

유스케이스와 액터는 서로 긴밀하게 연결되어 있다고 생각해야 한다. 시스템은 액터에게 가치를 제공하기 위하여 존재하고, 유스케이스는 시스템이 그 가치를 어떻게 제공하는 지 설명한다.

(1) 액터의 목적을 식별하는 것부터 시작한다.

(2) 시스템과 사용자의 정보 요구를 고려한다.

(3) 같은 내용이 반복되는 현상에 신경쓰지 않는다(적어도 처음에는)

(4) 유스케이스를 '기능'과 혼동하지 않는다.

(5) 가치에 초점을 맞춘다.

(6) 시스템 비전에서 유스케이스를 찾아낸다.

(7) 지원 유스케이스와 운영 유스케이스도 잊지 않는다.

(8) 액터와 부속 명세서와 함께 유스케이스 집합을 진화 시킨다.

아래는 유스케이스를 식별할 때 물어볼 수 있는 질문들을 요약한 것이다

- 시스템이 식별한 각 액터에게 제공해야 할 목적은 무엇인가?
- 액터가 시스템에게 알려야 할 갑작스러운 외부 변화가 있는가?
- 식별한 유스케이스들을 가지고 모든 시스템 특징을 수행할 수 있는가?
- 어떤 유스케이스가 시스템을 시작, 종료, 조정 지원, 유지할 수 있는가?
- 시스템에서 수정하거나 생성할 정보는 무엇인가?
- 시스템이 알아야 할 사건에는 어떤 것들이 있나?
- 시스템이 추적해서 액터에게 알려야 할 사건에는 어떤 것들이 있나?
- 유스케이스 모델이 모든 이해 당사자의 이해를 대변하는가?

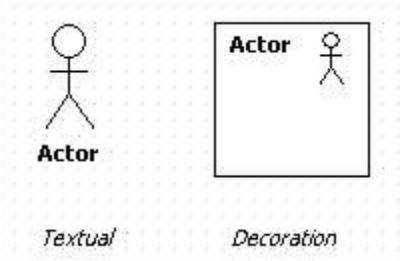
2.5. What is Use Case for?

- 유스케이스 다이어그램은 시스템 요구사항을 유스케이스 용어로 설명하는 모형이다. 유스케이스 다이어그램은 다음과 같은 목적을 가진다. 시스템의 의도된 기능 및 환경 모형으로 고객과 개발자 사이의 합의 및 시스템 개발 전반에 걸쳐 줄거리를 통합하는데 도움을 준다. 고객 또는 최종 사용자에게 시스템 행위를 전달한다. 결론적으로 이해하기 쉬워야 한다. 사용자 및 시스템과 교류하는 타시스템은 액터이다. 그들은 시스템 사용자를 대표하기 때문에 액터는 시스템의 범위를 정하는데 도움을 주고 예정된 일에 대한 명확한 상황을 제공한다. 유스케이스는 액터의 요구를 기반으로 개발된다. 이것은 사용자가 기대하는 시스템이 될 수 있도록 한다.

2.6. Use Case Diagram With StarUML

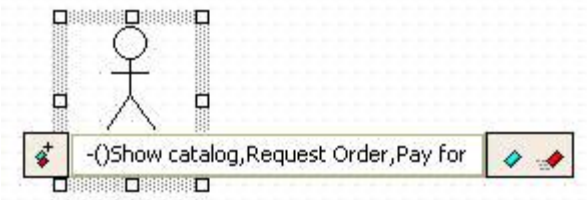
- Use-Case 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.
    - ◆ Actor
    - ◆ UseCase
    - ◆ Association
    - ◆ Derected Association
    - ◆ Generalization
    - ◆ Dependency
    - ◆ Include
    - ◆ Extend
    - ◆ System Boundary
    - ◆ Package
- Actor
    - ◆ 의미: 액터(Actor)는 일반적으로 시스템 외부에 존재하면서 시스템과 상호작용하는 개체다. 액터는 사람이거나 기계 혹은 소프트웨어 등이 될 수 있다.
    - ◆ 생성 방법: Actor를 생성하려면, Toolbox>UseCase의 Actor 버튼을 클릭하고 Main 윈도우창에서 Actor가 위치할 곳을 클릭합니다. Actor는 Stick Man 형태로

표현되지만, 사각형 모양에 오른쪽 상단에 아이콘이 포함된 Decoration View 형태로 사용되기도 합니다. Actor를 Decoration View 형태로 보여지도록 하기 위해서는 [Format] -> [Stereotype Display] -> [Decoration] 메뉴 아이템을 선택하거나 툴바의 버튼에서 [] 콤보 버튼의 [Decoration] 항목을 선택합니다.

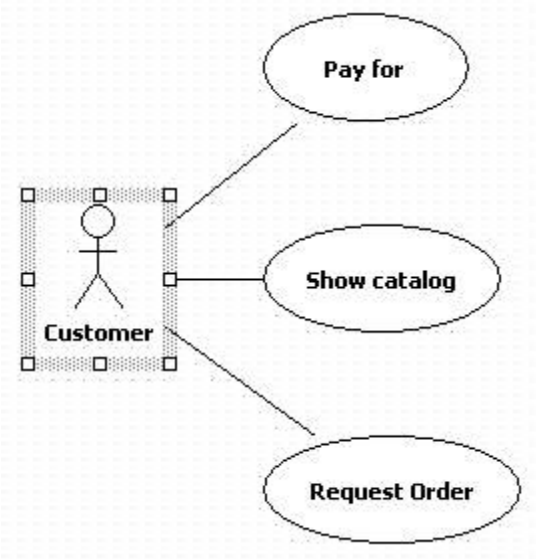


◆ Actor가 사용하는 UseCase를 한번에 여러 개 생성하는 방법:

(1) Actor를 더블 클릭해서 퀵다이얼로그가 나타나면, 퀵다이얼로그에서 "-()" 문자열 다음에 생성하려는 UseCase의 이름을 입력합니다. 각 UseCase 이름은 "," 문자로 구분해서 입력합니다.



(2) 그리고 [Enter]키를 누르면 Actor와 연관 관계를 가지는 여러개의 UseCase가 수직으로 자동 배열되어 생성됩니다.

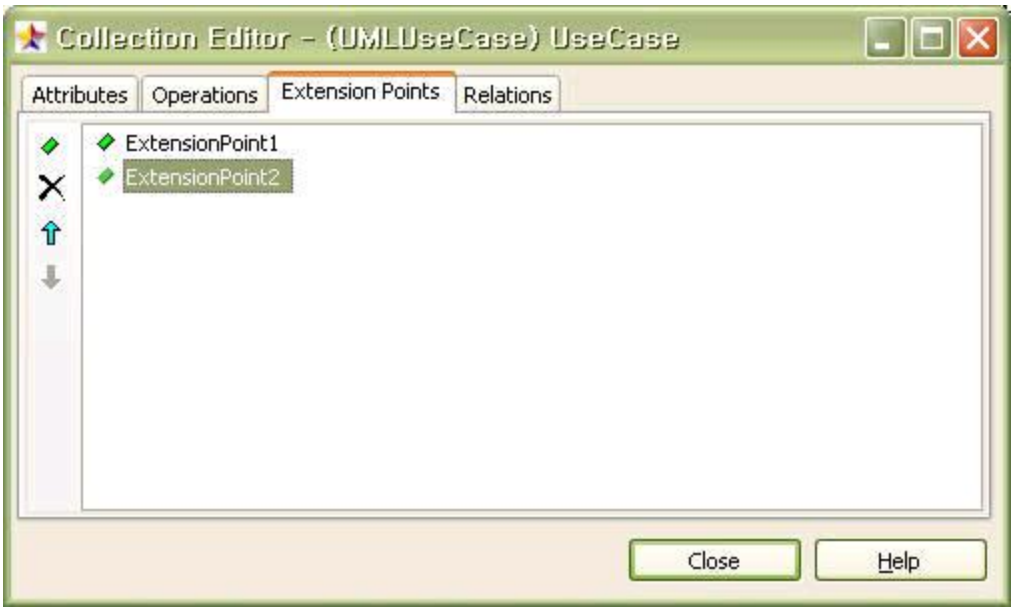


- Use Case

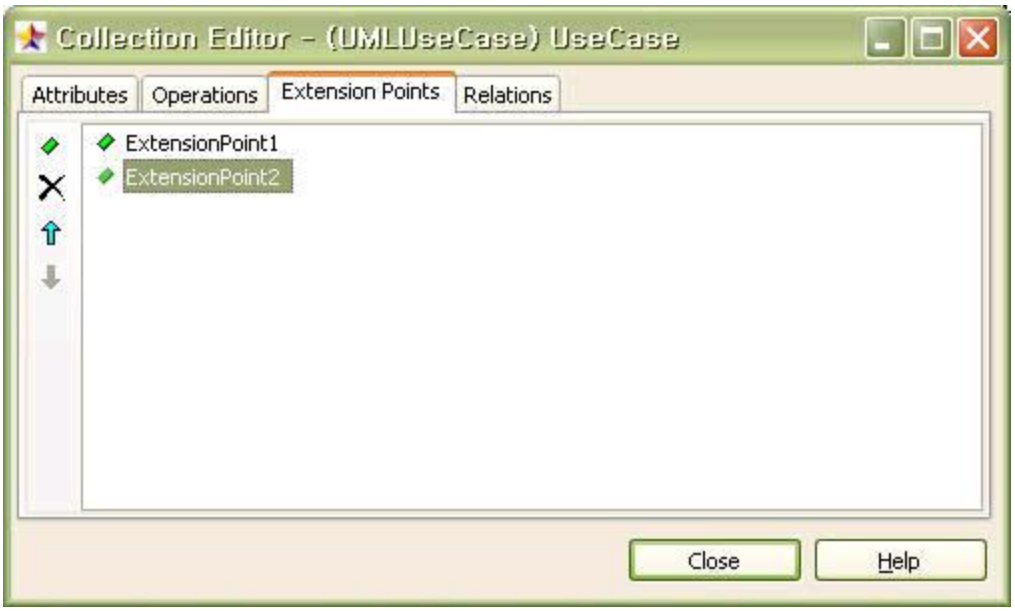
  ◆ Use-Case를 생성하는 방법: UseCase를 생성하려면, [Toolbox] -> [UseCase] -> [UseCase] 버튼을 클릭하고 Main 윈도우창에서 UseCase가 위치할 곳을 클릭합니다. UseCase는 Textual, Decoration, Iconic의 3가지 형태로 표현 가능합니다. [Format] -> [Stereotype Display]의 하부 메뉴 아이템을 선택하거나 [ ] 버튼의 아이템을 선택하면, UseCase의 스타일을 변경할 수 있습니다.

  ◆ Extension 추가하는 방법: 확장점은 유스케이스에서 확장되어지는 하나 또는 여러개의 위치를 참조한다.

◆ UseCase에 ExtensionPoints를 입력하려면 UseCase의 [Collection Editor...] 팝업 메뉴를 클릭하거나 UseCase의 ExtensionPoints 컬랙션 속성의 버튼을 클릭해서 [Collection Editor]에서 값을 수정합니다.



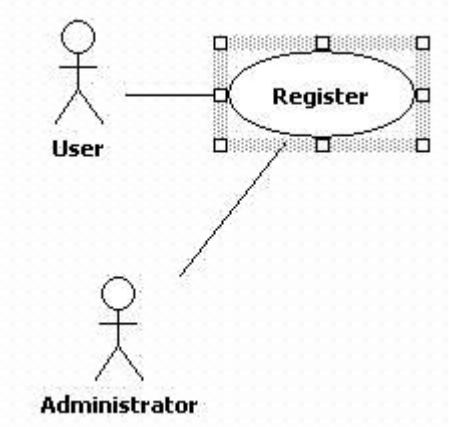
◆ UseCase Specification 속성 입력 방법: UseCase 작성시 많이 사용되는 속성들인 BasicFlow, AlternativeFlow등을 입력하기 위해서는 [Tagged Values...] 팝업 메뉴를 선택하거나 Ctrl+F7 버튼을 클릭하여 Tagged Value Editor의 UseCaseSpecification을 선택하여 필요한 속성의 값을 입력합니다.



◆ UseCase로부터 Actor 생성하는 방법: 현재 선택된 UseCase와 연관 관계를 가지는 Actor 여러개를 한꺼번에 만들려면 UseCase의 단축 생성 구문을 사용합니다.

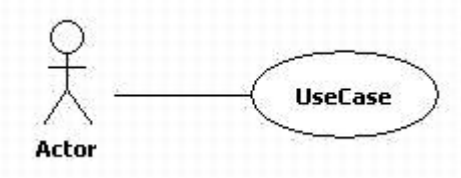
(1) UseCase를 더블 클릭하거나 UseCase를 선택하고 [Enter]키를 누른다. Quick Dialog가 나타나면, Quick Dialog에서 "()-" 문자열 다음에 연관된 Actor의 이름을 입력합니다. 각 Actor 이름은 "," 문자로 구분해서 입력합니다.

(2) 그리고 [Enter]키를 누르면 UseCase와 연관 관계를 가지는 Actor들이 생성됩니다.

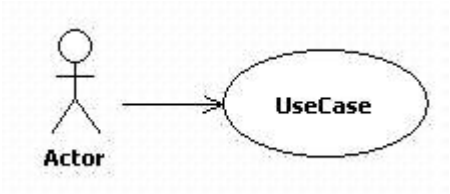


- Association / Derected Association

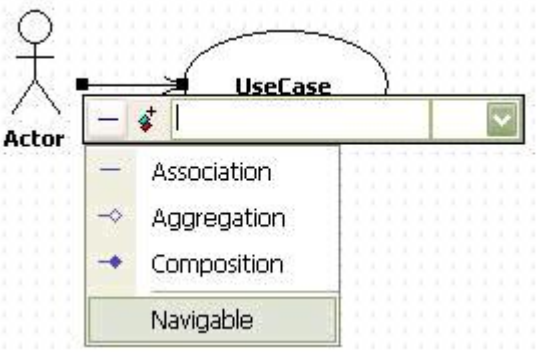
◆ 의미: 연관(Association)은 클래스류(Class, Interface, Enumeration, Signal, Exception, Component, Node, UseCase, Actor) 사이의 의미적 관계를 정의합니다.



◆ Association 생성하는 방법: Association를 생성하려면, [Toolbox] -> [UseCase] -> [Association] 버튼을 클릭하고 Main 윈도우창에서 연결하려는 첫번째 요소에서 두번째 요소로 마우스를 누르고 드래그하면 됩니다.



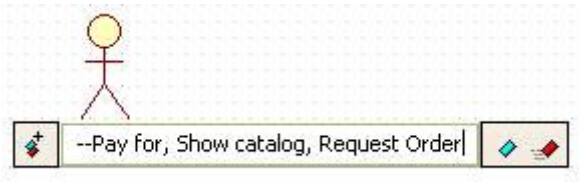
◆ DirectedAssociation 생성하는 방법: Association 생성방법과 동일하며, 두 요소간 마우스 드래그를 화살표 방향으로 합니다.



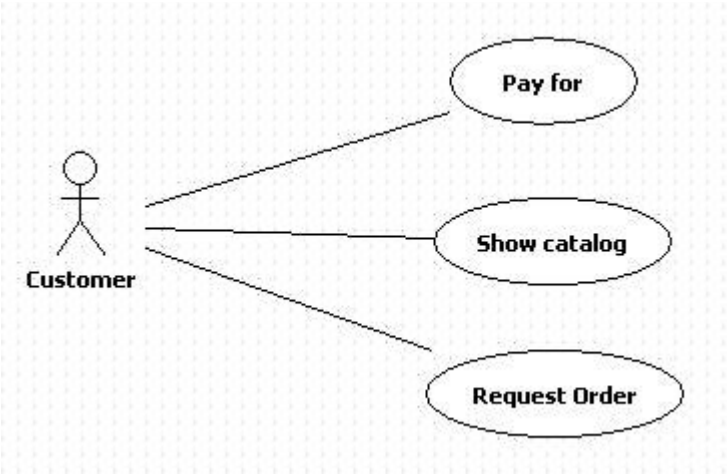
또는 Association을 생성하고 Actor쪽 assocation의 끝을 클릭하고 Quick Dialog의 Navigable의 체크를 취소하면 DirectedAssociation으로 변합니다.

◆ 요소로부터 Association/Directed Associatiopn 관계의 요소 생성하는 방법: 현재 선택된 요소로부터 Assocation/DirectedAssociation 관계를 갖는 요소를 만들려면 요소의 단축 생성 구문을 사용합니다.

(1) 요소를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "--" 또는 "->" 문자열 다음에 Association/DirectedAssociation 관계를 갖는 다른 요소의 이름을 입력합니다. 여러개의 요소와 관계를 맺기 위해서는 각 요소 이름은 "," 문자로 구분해서 입력합니다.



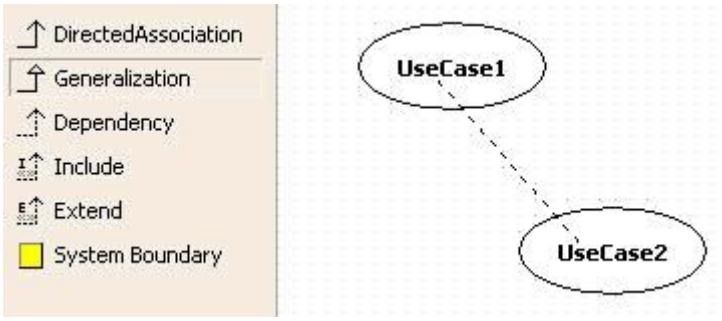
(2) 그리고 [Enter]키를 누르면 선택된 요소와 Association/DirectedAssociation 연관 관계를 가지는 여러 요소들이 생성되고 자동 배열되어 생성됩니다.



- Generalization

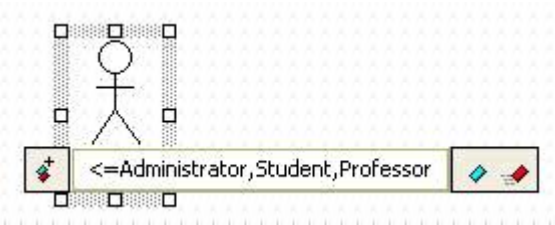
  ◆ 의미: 일반화(Generalization)">일반화(Generalization)는 더 일반적인 요소와 더 구체적인 요소를 연결하는 관계입니다.

◆ Generalization 생성하는 방법: Procedure for creating generalization
Generalization를 생성하려면, [Toolbox] -> [UseCase] ->[Generalization] 버튼을 클릭하고 Main 윈도우창에서 연결하려는 자식 요소에서 부모 요소로 마우스를 누르고 드래그하면 됩니다.

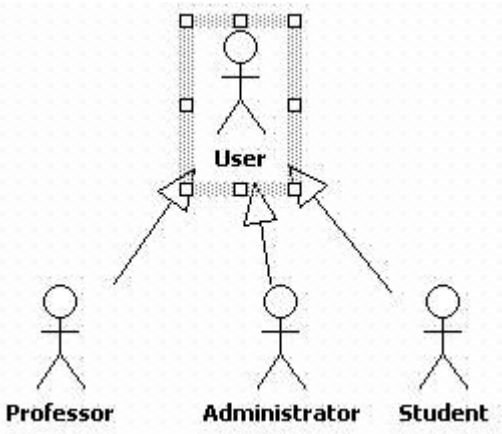


◆ Actor를 상속하는 여러 개의 자식 Actor 생성하는 방법:

(1) 특정 요소를 상속하는 하위 요소가 여러개일 경우에 Quick Dialog의 단축 생성 구문에서 다음과 같이 입력하면 현재 요소를 상속하는 여러개의 하위 요소를 한꺼번에 생성합니다.



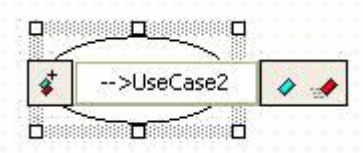
(2) 하위 요소들은 선택된 요소의 아래에 생성되면서 정렬됩니다.



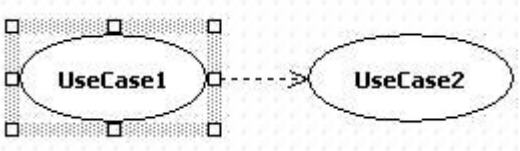
만약 상속할 상위 요소가 여러개인 경우에는 Quick Dialog의 단축 생성 구문에서 "<=" 대신에 "=>" 문자열을 사용한다.

- Dependency

  - ◆ 의미: 의존관계(Dependency)">의존관계(Dependency)는 어떤 요소의 구현이나 기능을 위해 다른 요소의 존재가 요구 되어지는 의존적인 관계를 의미합니다.
  - ◆ Dependency 생성 방법: Dependency를 생성하려면, [Toolbox] -> [UseCase] ->[Dependency] 버튼을 클릭하고 Main 윈도우창에서 요소에서 의존 하는 요소로 마우스를 누르고 드래그하면 됩니다.
  - ◆ UseCase로부터 의존하는 다른 UseCase 생성하는 방법:
    퀵다이어로그의 단축 생성구문을 다음과 같이 입력하면 됩니다.



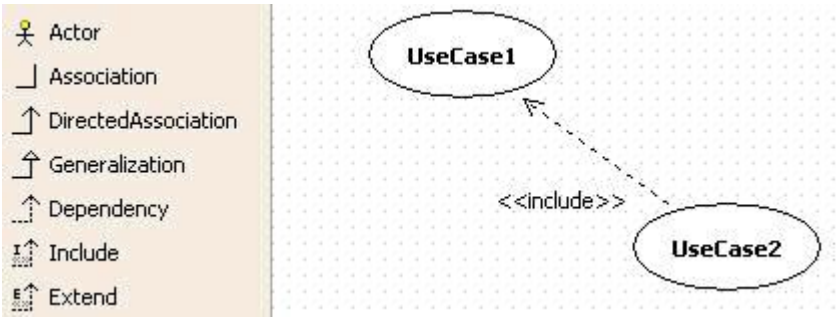
그러면 다음과 같이 두 요소간의 Dependency가 생성됩니다.



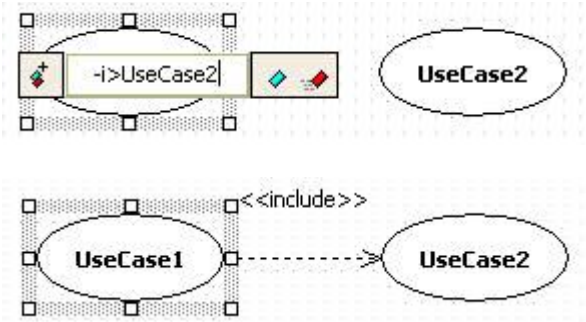
- Include(포함)

  - ◆ 의미: 포함관계(Include)는 어떤 유스케이스가 특정 유스케이스의 행위를 포함한다는 것을 정의합니다.

◆ Include 생성 방법: Include를 생성하려면, [Toolbox] -> [UseCase] -> [Include] 버튼을 클릭하고 Main 윈도우창에서 요소에서 포함할 요소로 마우스를 누르고 드래그하면 됩니다.



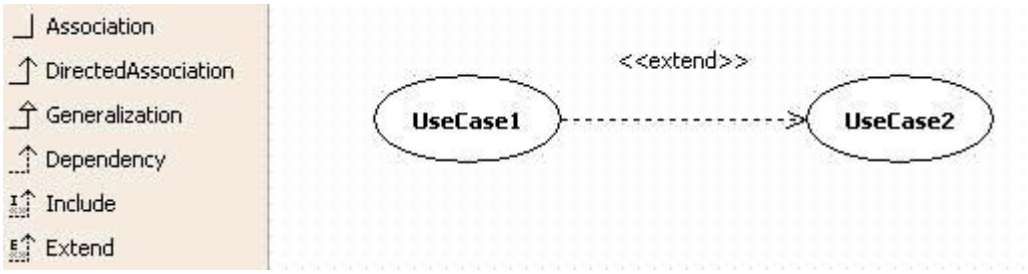
◆ UseCase로부터 Include 관계의 다른 UseCase 생성하는 방법:
퀵다이어로그의 단축 생성구문을 다음과 같이 입력하면 됩니다.



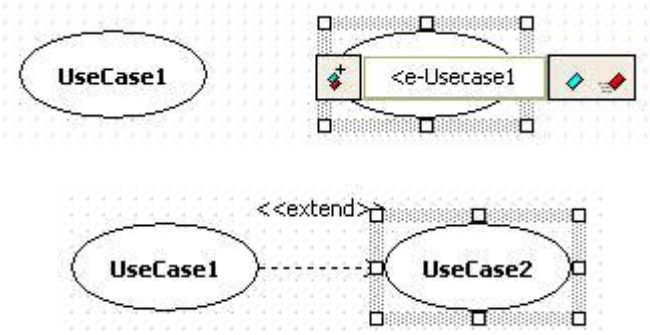
- Extend

◆ 의미: 확장관계(Extend)">확장관계(Extend)는 어떤 유스케이스가 특정 유스케이스에 정의된 행위로 추가 확장될 수 있다는 것을 나타냅니다.

◆ Extend 생성 방법: Extend를 생성하려면, [Toolbox] -> [UseCase] -> [Extend] 버튼을 클릭하고 Main 윈도우창에서 요소에서 확장할 요소로 마우스를 누르고 드래그하면 됩니다.

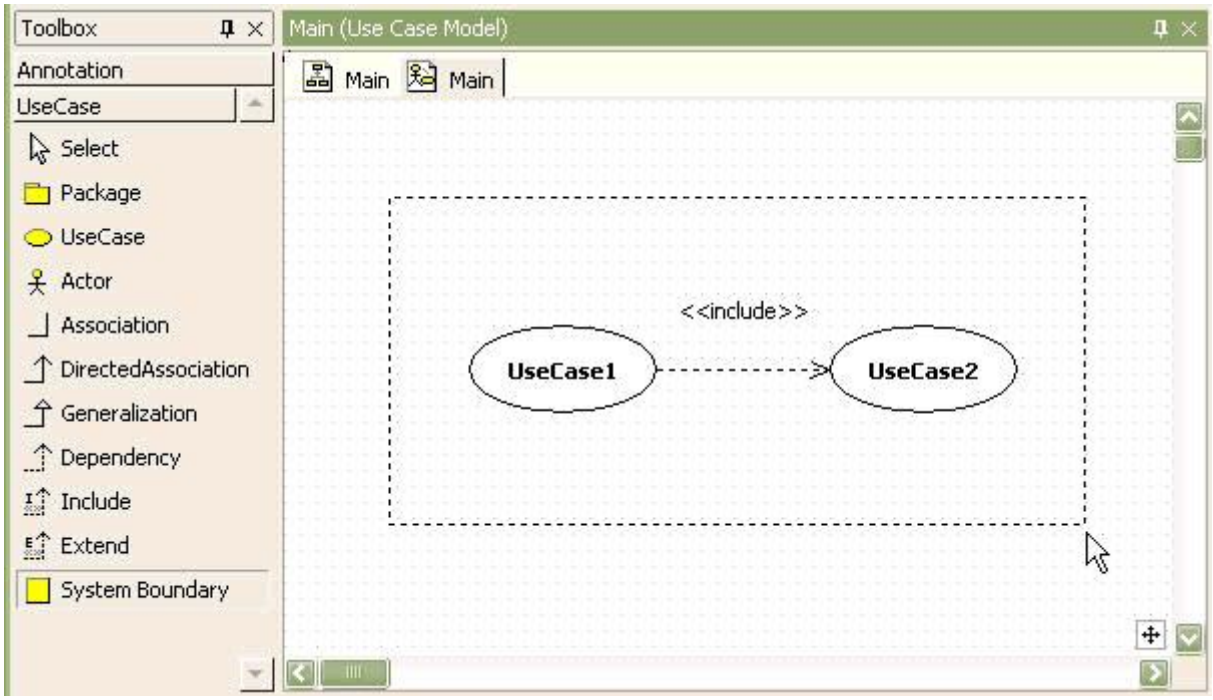


◆ UseCase로부터 Extend 관계의 다른 UseCase 생성하는 방법:
퀵다이어로그의 단축 생성구문을 다음과 같이 입력하면 됩니다.



- System Boundary

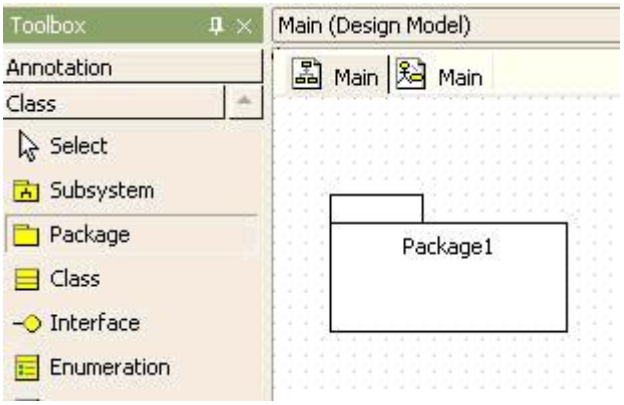
◆ 생성하는 방법: System Boundary를 생성하려면, [Toolbox] -> [UseCase] -> [System Boundary] 의 System Boundary 버튼을 클릭하고 Main 윈도우창에서 System Boundary가 삽입될 위치에 마우스를 클릭하고 생성될 크기 만큼을 드래그합니다.



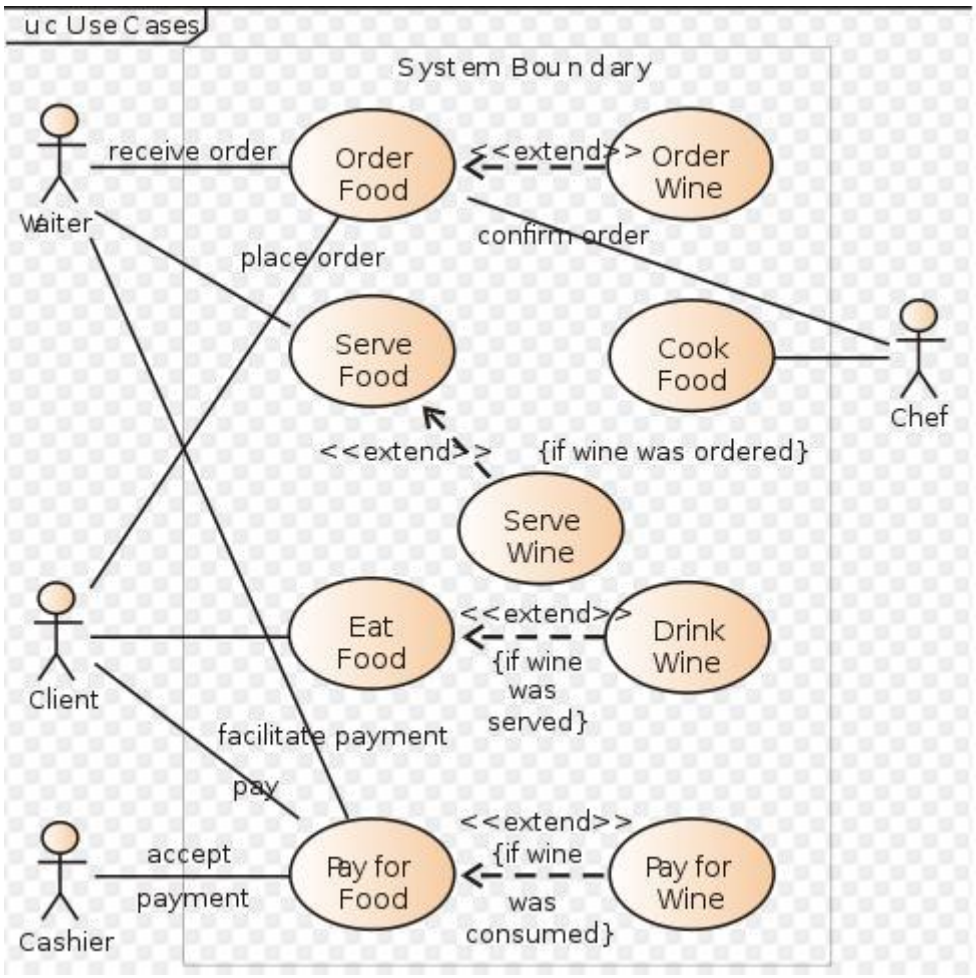
- Package

  ◆ 의미: 패키지(Package)는 모델 요소들을 논리적으로 그룹화하여 관리하기 위한 요소입니다. 패키지는 요소들을 조직화하기 위한 어떠한 용도로 사용되어도 무방한 매우 일반적인 요소입니다. 패키지 대신 모델(Model), 서브시스템(Subsystem)의 더욱 특수화된 요소를 사용할 수도 있습니다.

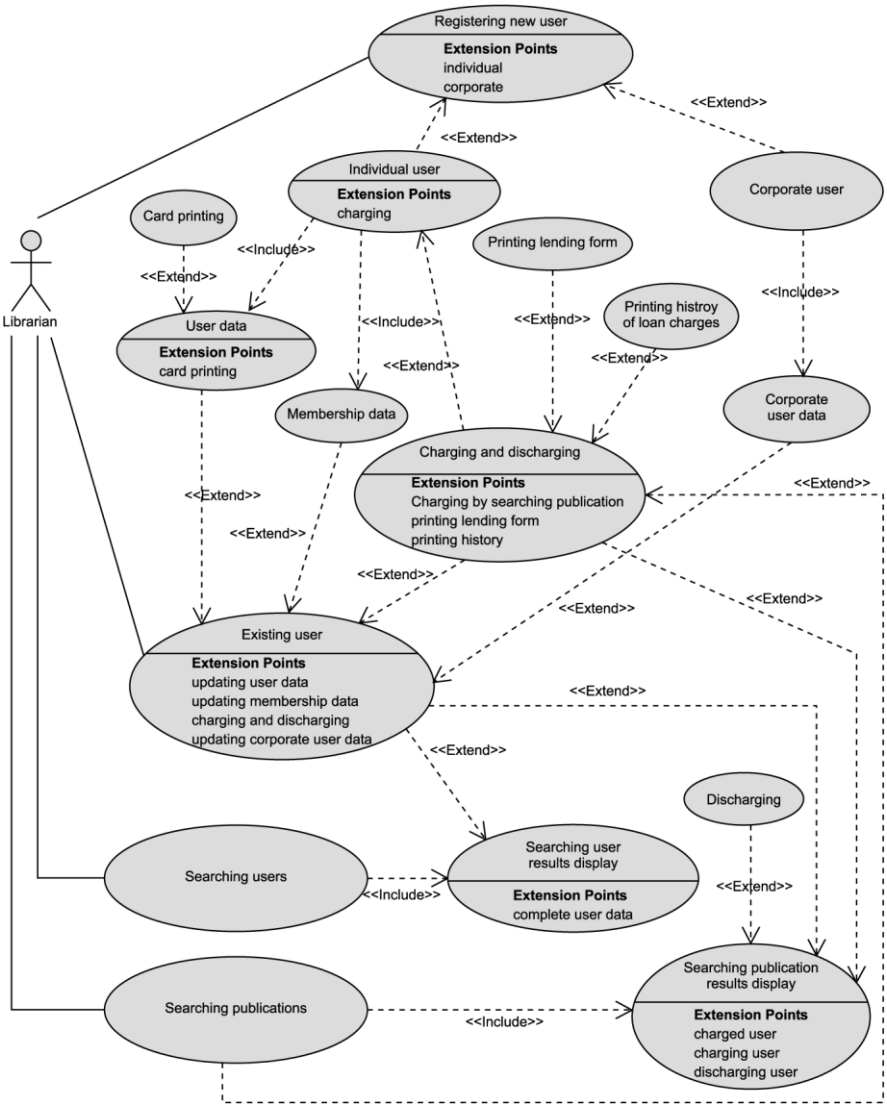
  ◆ Package 생성하는 방법: Package를 생성하려면, [Toolbox] -> [UseCase] -> [Package] 버튼을 클릭하고 Main 윈도우창에서 Package가 위치할 곳을 클릭합니다.



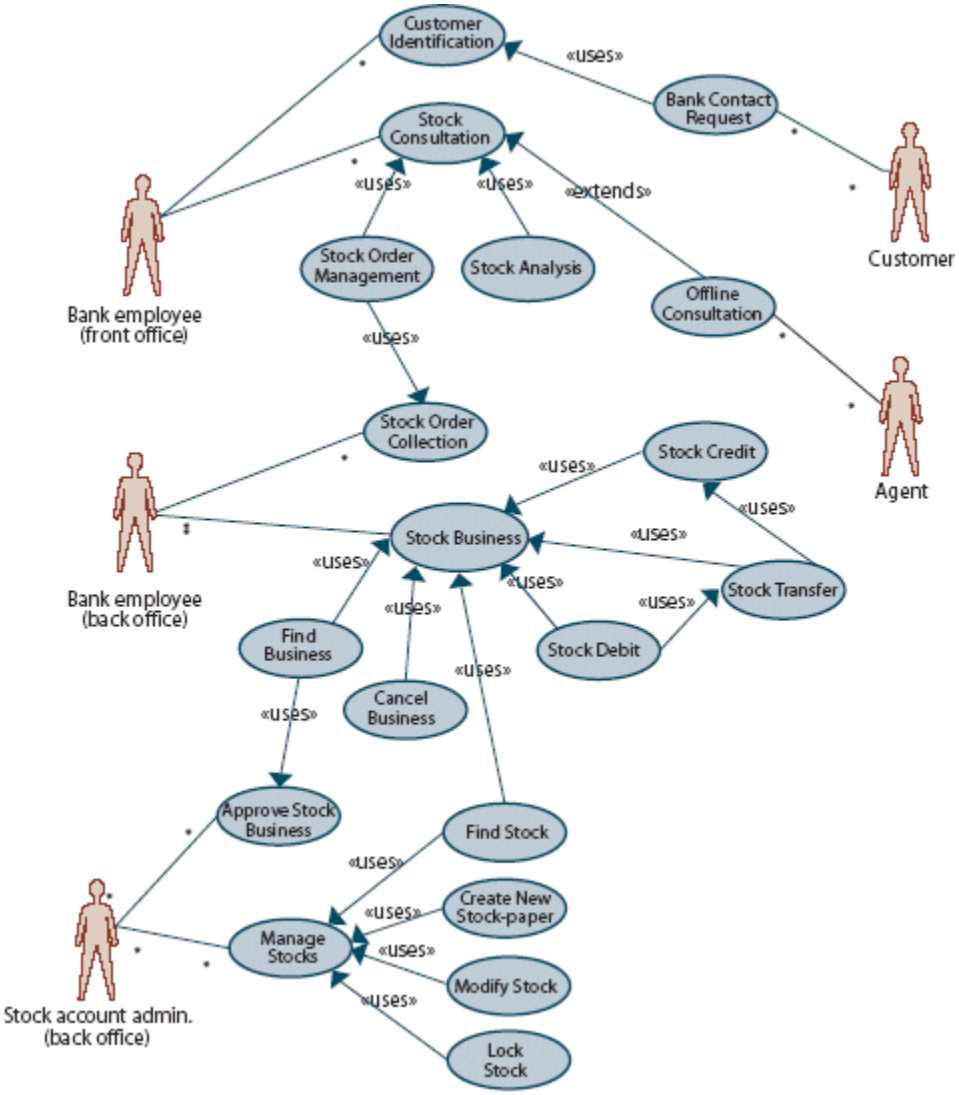
2.7. Examples of Use Case Diagram



<Figure 7. Use Case Diagram>

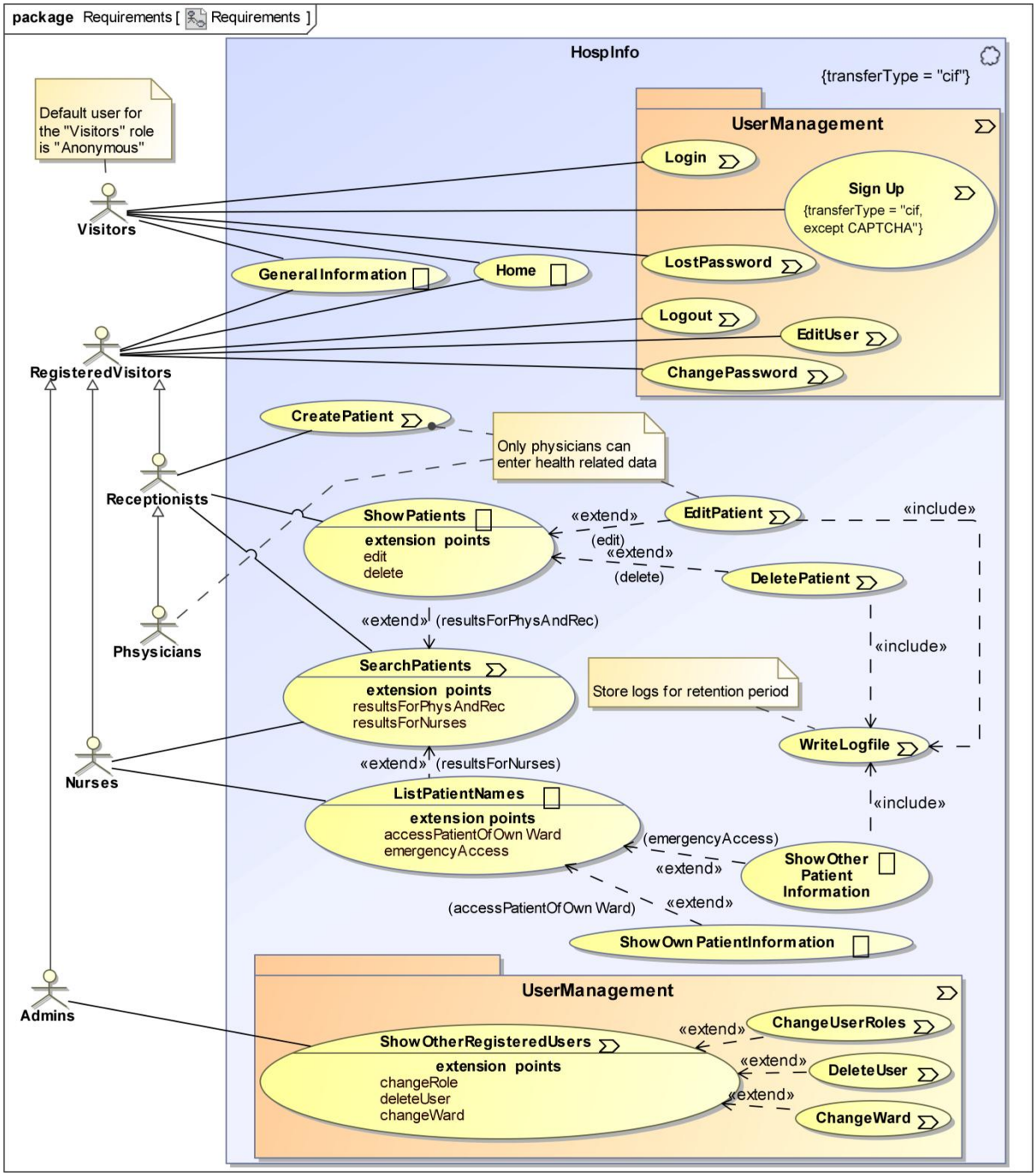


<Figure 8. Use Case Diagram>

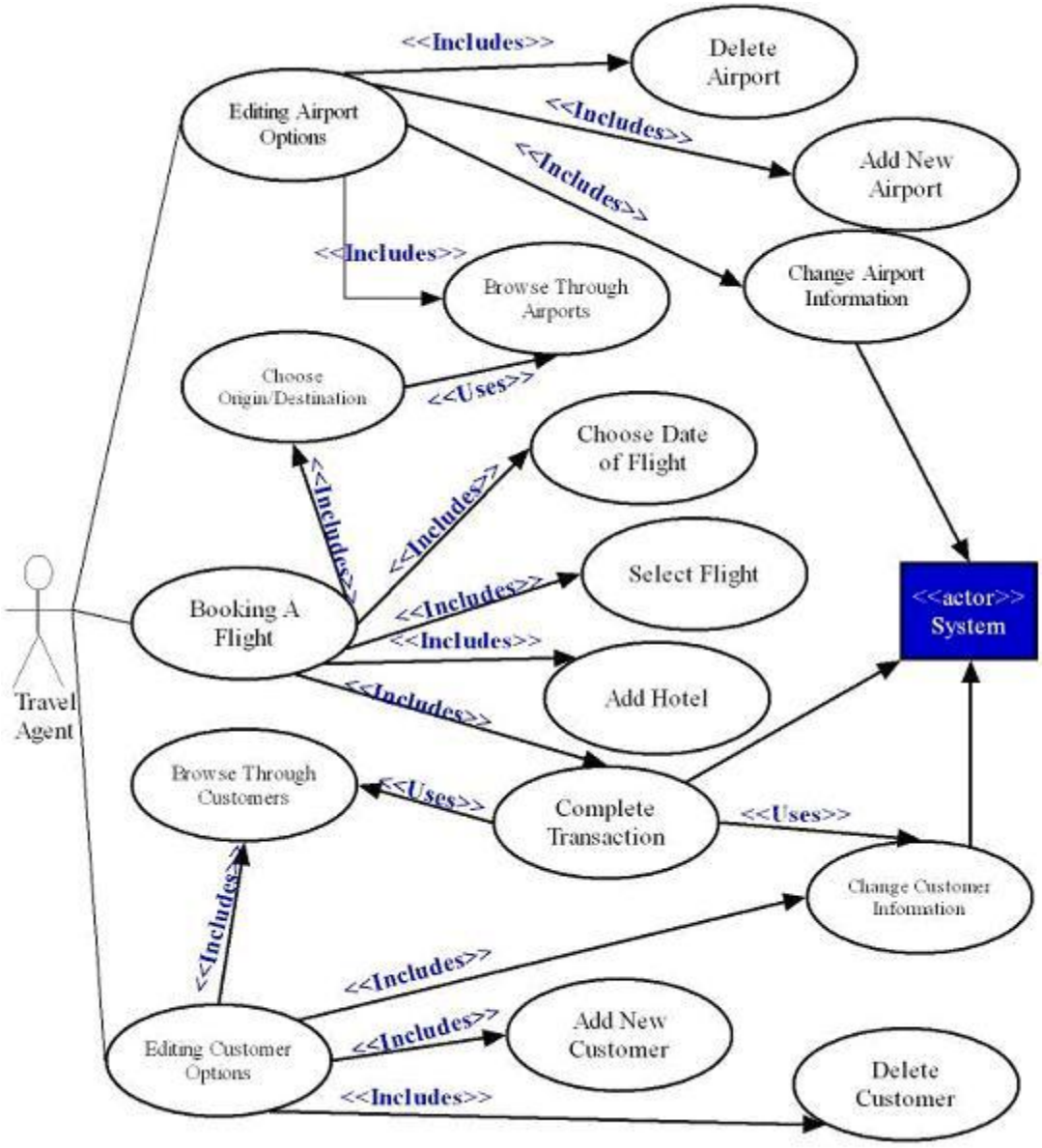


<Figure 9. Use Case Diagram>

<Figure 10. Use Case Diagram>



<Figure 11. Use Case Diagram>



<Figure 12. Use Case Diagram>

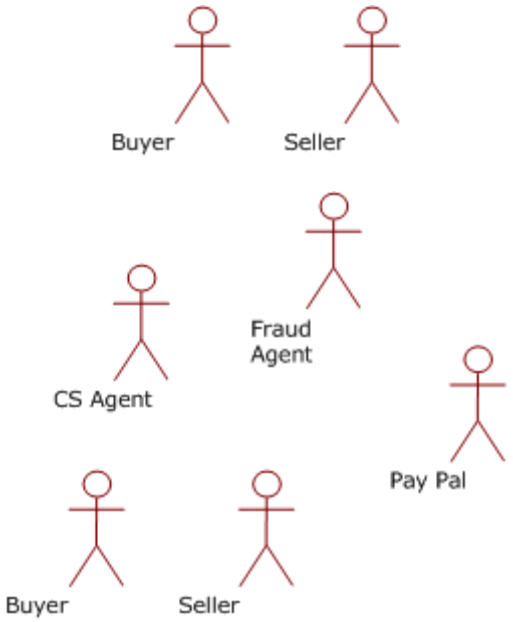
2.8. 13 tips for Use case

2.8.1. Use Case 만들 때, 완벽해 하지 말고 생산적인 자세가 되어야 한다.

- Be Agile! 실수를 두려워 하지 말고 가능한 많은, Use Case 작성에 주저함이 없어야 한다. 많은 설계자, 혹은 제안자들은 Use Case 초기 작성 단계에서부터 너무 완벽(Perfect)하려고 만 하는 경향이 있는데 이는 Use Case 작성에 비효율적이다. 가능한 모든 경우를 생각하여 도안에 그려낸 후, 실 경우에 적용하여 비교 대조 한다.

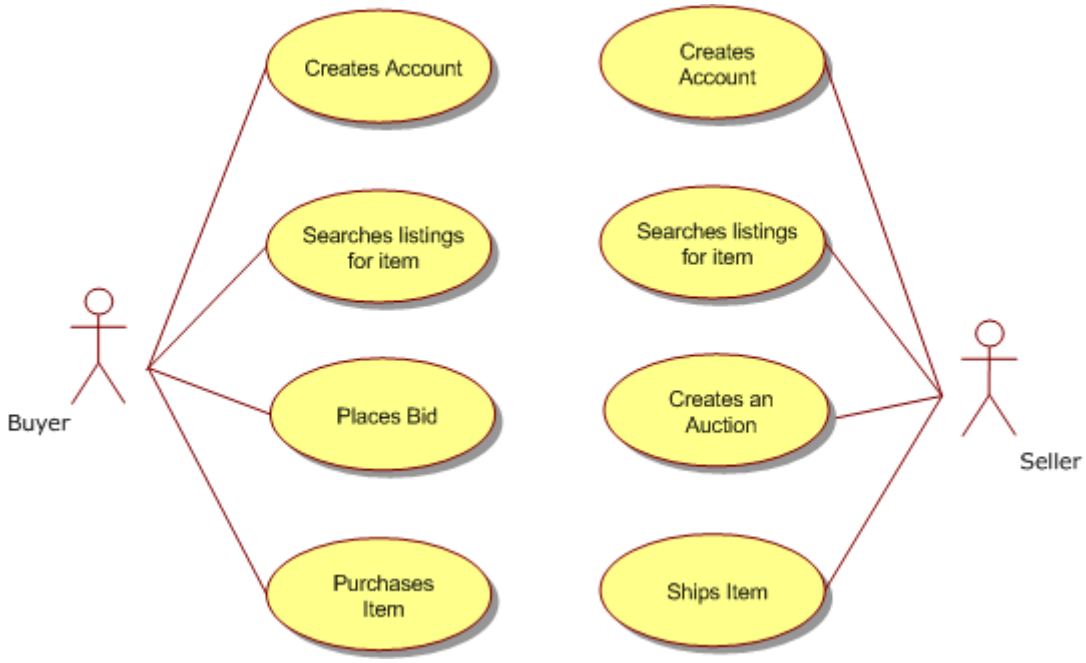
2.8.2. Actor를 정의한다.

- Actor 의 정의는 문서 상단에 정의해 두었으며, Use Case 정의에 있어 가장 중요한 부분이다. 해당 소프트웨어 혹은 시스템을 누가 이용하는지, 어떤 Actor에 의해 연산되고 프로세싱 되는지 명확히 정해져야 Use Case 설계에 큰 효과를 줄 수 있다.



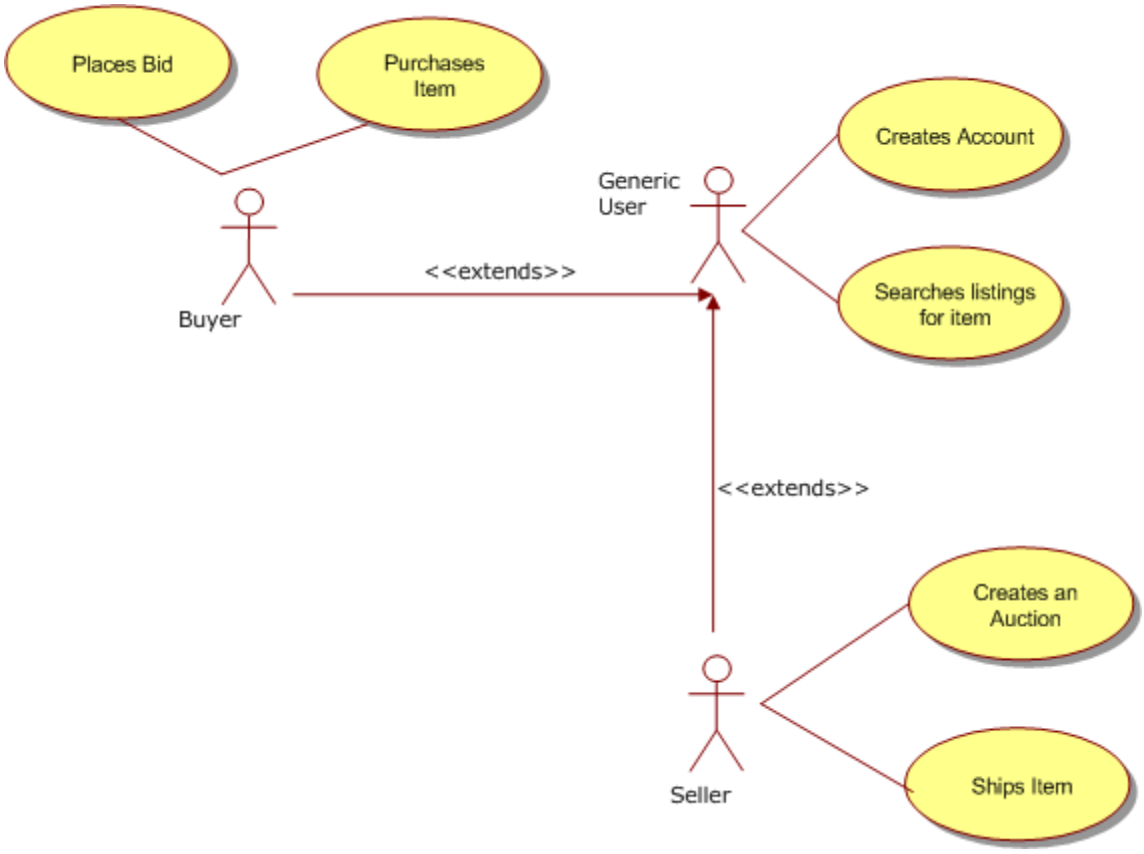
2.8.3. Primary Use Case 를 정의한다.

- Use Case 중 가장 주요한, 혹은 주된 액션이 되는 Use Case를 정의한다. 이를 정의하는 것은 Rainy Day 라고도 하며 이를 정의하여 큰 도안을 확보한다.



2.8.4. Use Case 의 Reuse(재사용) 될 것들을 정의한다.

Use Case는 재사용 혹은 여러 번 사용되야 할 경우가 많기에 이를 미리 명확히 해두어야만 Use Case 작성에 큰 도움을 준다.



2.8.5. Use Case Index를 만든다.

Use Case ID	Use Case Name	Primary Actor	Scope	Complexity	Priority
1	Places a bid	Buyer	In	High	1
2	Purchase an item	Buyer	In	High	1
3	Creates Account	Generic User	In	Med	1
4	Searches listings	Generic User	In	Med	1
5	Provides Feedback	Generic User	In	High	1
6	Creates an auction	Seller	In	High	1
7	Ships an item	Seller	In	High	2

2.8.6. Use Case 의 Key Components 를 정의한다.

Use Case Element	Description
Use Case Number	ID to represent your use case
Application	What system or application does this pertain to
Use Case Name	The name of your use case, keep it short and sweet
Use Case Description	Elaborate more on the name, in paragraph form.
Primary Actor	Who is the main actor that this use case represents
Precondition	What preconditions must be met before this use case can start
Trigger	What event triggers this use case
Basic Flow	The basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The exceptions will be handled in the "Alternate Flows" section.
Alternate Flows	The most significant alternatives and exceptions

2.8.7. Use Case 의 설명을 작성한다.

Use Case Number	1
Use Case Name	Buyer Places a Bid
Description	An EBAY buyer has identified an item they wish to buy, so they will place a bid for an item with the intent of winning the auction and paying for the item.

2.8.8. Use Case 의 기본 흐름(Basic Flow)을 작성한다.

2.8.9. Use Case 의 대체 흐름(Alternate Flow)을 작성한다.

2.8.10. Use Case 문서를 작성한다.

2.8.11. Use Case Model Diagram 을 작성한다.

2.8.12. User Story가 필요하다면 작성한다.

- Use Story란, 간단하게 사용자 입장에서 시스템을 이용할 때 쓰는 일종의 시나리오(각본)이다. 도면만 보고는 사용자가 어떤 액션(동작)을 취해서 시스템 혹은 소프트웨어를 이용할지 명확하지 않으므로 간단한 시나리오를 작성하는 것을 의미한다.

2.8.13. Use Case 를 바탕으로 구현한다.

3. Conclusion

- UML(Unified Modeling Language)는 객체 지향 시스템 혹은 소프트웨어 제작 시 반드시 필요한 도식화 모델이자 도구이다. 본 언어로 쉽게 도식화를 시키려면 필요한 소프트웨어들이 많은데 StarUML을 추천하는 바이다. 본 문서 2. Use Case 부분에 With StarUML 항목이 존재한다.

- 소프트웨어 모델링 및 분석 수업에서 처음으로 UML 을 사용하여 소프트웨어를 모델링하게 되었는데 UML 에 대한 전반적인 개념과 이해는 본 문서에 모두 포함시켜 놓았다. 나아가 실무에 반드시 적용할 수 있고 학문적인 발전에 있어서도 반드시 필요한 부분이라 생각한다.

- 소프트웨어를 구현 개발하는 것도 아무리 강조해도 지나치지 않을 만큼 중요하다. 하지만 그 이전에 앞서 소프트웨어나 시스템을 모델링, 즉 설계하는 부분에 있어서는 소프트웨어 공학 개발 방법론에 의거하여 UML 을 사용하여 도식화, 혹은 설계하는 것에 초점을 둔 이번 수업에 깊은 발전의 뜻이 있길 바란다.

4. References

- **Wikipedia, Unified Modeling Language,**
  http://en.wikipedia.org/wiki/Unified_Modeling_Language
- **Daum Blog, Objective of UML**
  http://blog.daum.net/nanhjb/2390920
- **Wikipedia, Use Case Diagram**
  http://en.wikipedia.org/wiki/Use_Case_Diagram
- **Wikipedia, Use Case**
  http://en.wikipedia.org/wiki/Use_case
- **UML Image, Oracle**
  http://docs.oracle.com/cd/E21764_01/doc.1111/e15866/img/run_uml3.gif
- **Wikispace, UML Use Case Diagram**
  http://cct355-f07.wikispaces.com/file/view/UML_Use-case-diagram.gif/31642357/UML_Use-case-diagram.gif
- **Pacestar, UML Use Case Diagram**
  http://www.pacestar.com/uml/SMP_USE.JPG
- **Claudiodesio, UML Syntax Reference Poster**
  http://www.claudiodesio.com/ooa&d/UMLPoster/UMLPoster.jpg
- **Personal tistory Blog, What is Use Case?**
  http://gisulsa.tistory.com/234
- **SoureceForge, Use Case Diagram Modeling**
  http://staruml.sourceforge.net/docs/user-guide(ko)/ch05_1.html
- **GatherSpace, Tip13 for Use Case**
  http://www.gatherspace.com/static/use_case_example.html#13
- http://www.comscigate.com/tutorial/KjellStyle/OmarKhan/Chapter%203/UML%20Use%20Cases.JPG
- http://uwe.pst.ifi.lmu.de/examples/HospInfo/diagrams/Use_Case_Diagram__Requirements__Re

quirements.png

- http://coreyleong.files.wordpress.com/2010/01/100105_real_estate_project_mgmt_use_cases_470x5881.png?w=632
- http://i.msdn.microsoft.com/dynimg/IC119797.gif
- http://www.emeraldinsight.com/content_images/fig/2630270112001.png
- http://www.businessanalystfaq.com/blog/wp-content/uploads/2011/02/use-case-diagram-example1.jpg
-