# 유스케이스 모델링 도구의 설계 및 구현

최환복* · 김윤호**

## A Design and Implementation of Usecase Modeling Tool

Hwan-bok Choi* · Yun-ho Kim**

요 약

본 논문에서는 유스케이스 다이어그램과 기술서를 통합하여 작성하기 위한 도구를 제시하고자 한다. 또한 관계를 기준으로 하는 다이어그램 유지방법을 제시하였으며, 다이어그램 작성에서 항목 간 연결 유효성 검사를 제공하여 UML 표준을 따르는 다이어그램 작성이 가능하도록 하였다. 본 논문에서 제시하는 유스케이스 모델링 도구는 편리하고 효과적으로 유스케이스 모델링을 가능하게 해줄 뿐만 아니라, 통합되고 일관성 있는 유스케이스 모델링에 기여할 것으로 기대된다.

ABSTRACT

This paper presents a design and implementation of Usecase modeling tool to support integrated usecase diagramming and description writing. It also suggests diagram maintaining based on relationship and it makes UML standard diagramming by providing connection verification between diagram elements. It results in not only user-friendly and effective usecase modeling but it also contributing to integrated and consistent usecase modeling.

키워드

UML, Usecase CASE Tool, OOP, Object-oriented Software System

* 안동대학교 컴퓨터공학과 석사과정
** 안동대학교 전자정보산업학부 정교수

접수일자 2008. 12. 17
심사완료일자 2009. 02. 26

# Ⅰ. 서 론

유스케이스 모델링[1]은 시스템의 기능적인 요구사항을 이끌어내고 시스템과 시스템을 사용하는 사용자간의 통상적인 교류를 기술하여 시스템이 어떻게 사용되는지를 표현하는 분석기술이다. 유스케이스는 유스케이스의 시각적 목차와 유스케이스 간의 연관관계를 나타내는 유스케이스 다이어그램과 실질적인 사용사례를 문자를 이용하여 표현하는 유스케이스 기술서(description)로 나눌 수 있다. 기술서와 다이어그램은 서로 밀접한 연관이 있지만 현재의 모델링 도구는 유스케이스 다이어그램에 집중되어 있으며 기술서를 작성하기 위해서는 별도의 문서화 도구를 이용해야 한다[2][3][4]. 이렇게 유스케이스 다이어그램과 기술서가 분리된 모델링은 상호간의 접근이 단절되어 있어 일관성이 떨어진다. 이러한 단절성을 해결하기 위해서는 하나의 시스템에서 유스케이스 다이어그램과 기술서 모델링이 가능하고 이들 사이에 연결이 가능한 유스케이스 모델링 도구가 필요하다. 따라서 본 논문에서는 유스케이스 다이어그램과 기술서 모델링을 지원하고 상호간의 연결을 제공하는 유스케이스 모델링 도구를 제시하고자 한다. 또한 유스케이스 다이어그램에서 관계를 기준으로 한 유스케이스 다이어그램 저장방법과 다이어그램 항목 간 관계연결의 유효성 검사방법을 제시하고자 한다.

# Ⅱ. 유스케이스 모델링 도구

## 2.1 액터의 분류

Larman[5]는 액터를 그 역할에 따라 3가지로 분류하였다. 액터의 분류가 유스케이스 모델링에서 필수적인 것은 아니지만 역할을 분류함으로써 해당 액터의 역할이 무엇인지 명확하게 알 수 있다. 본 논문에서는 액터의 분류를 다이어그램에 표현하여 액터의 역할을 명확히 보이도록 한다. Larman이 분류한 액터의 종류를 다음과 같다.

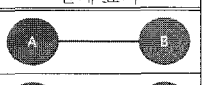
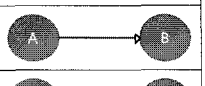
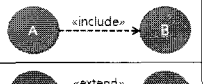
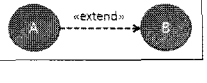
- 주요액터
- 지원액터
- 장외액터

액터의 역할에 대한 설명은 다음과 같다. 주요액터(primary actor)는 시스템의 서비스를 사용하여 사용자의 목적을 수행한다. 식별하는 이유는 유스케이스를 수행시키는 사용자의 목적을 알기 위해서이다. 지원액터(supporting actor)는 시스템에 서비스를 지원하는 액터로써 컴퓨터 시스템 또는 조직이나 사람이 될 수도 있다. 식별하는 이유로는 외부 인터페이스와 프로토콜을 명확히 하기 위해서이다. 장외액터(off-stage actor)는 유스케이스의 행위에서 이해관계를 갖는 액터이다. 식별하는 이유는 외부적으로 명명되지 않으면 빠뜨리기 쉽기 때문이다.

## 2.2. 유스케이스 다이어그램에 표현되는 관계

UML[6]은 유스케이스 다이어그램에서 항목 간에 사용할 수 있는 관계를 정해두었다. 연관은 액터와 유스케이스가 서로 통신하는 것을 의미하며 액터와 유스케이스 사이에 실선으로 표시된다. 연관은 액터와 유스케이스 관계에서만 사용이 가능하다. A 항목에서 B항목에서의 일반화는 B 항목이 A 항목의 특수화를 의미하며 부모 쪽으로 향하는 속이 비고 닫힌 화살표와 실선으로 표시된다. 일반화는 액터와 액터, 유스케이스와 유스케이스 관계에서만 사용할 수 있다. A 항목에서 B 항목으로의 포함 관계는 B 항목에 명시된 행위를 A 항목이 포함하는 것을 의미하며 기반 항목으로 향하는 화살표와 점선과 «include» 레이블로 표시된다. 포함 관계는 유스케이스와 유스케이스 관계에서만 가능하다. A 항목에서 B 항목으로 확장 관계는 B 항목이 A 항목에 명시된 행위로 인해 증대되는 것을 의미하며 기반 항목으로 향하는 화살표와 점선과 «extend» 레이블로 표시된다. 확장 관계는 유스케이스와 유스케이스 관계에서만 가능하다. 표 1은 UML에 정의된 항목간의 관계와 표기를 나타낸 것으로 본 논문에서는 아래와 같은 관계를 처리하도록 하였다.

표 1. 유스케이스 다이어그램에서 표현되는 관계와 표기
Table 1. Relationships and notation in usecase diagram

| 관계 | 가능한 항목 | 관계 표기 |
|---|---|---|
| 연관 | 액터와 유스케이스 |  |
| 일반화 | 액터와 액터, 유스케이스와 유스케이스 |  |
| 포함 | 유스케이스와 유스케이스 |  |
| 확장 | 유스케이스와 유스케이스 |  |

### 2.3 액터와 유스케이스 유지를 위한 정보 설정

#### 2.3.1 다이어그램 표현에 필요한 정보 설정

유스케이스 다이어그램 모델링을 지원하기 위해서는 우선 다이어그램에 표현되는 항목을 식별해야 한다. 그림 1는 UML 명세(spec)에 기반을 둔 유스케이스 다이어그램을 보인 것이다. 유스케이스 다이어그램에는 액터, 유스케이스 그리고 액터와 유스케이스 같은 항목간의 관계가 나타난다. 다이어그램에서 액터에 표현되는 요소는 액터를 나타내는 그림, 액터이름, 액터 유형이 있다. 따라서 액터를 표현하기 위해 다음과 같은 요소를 사용하기로 한다.

- 액터 그림
- 액터 이름
- 액터의 유형

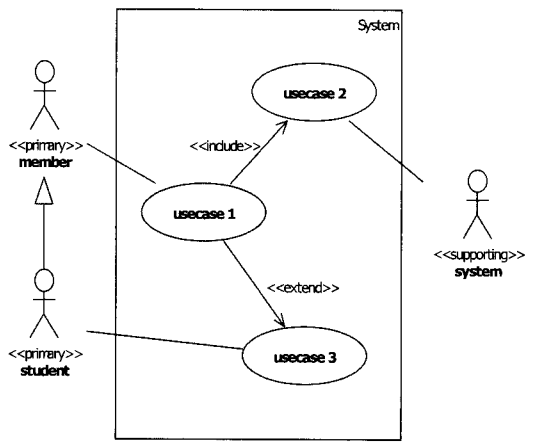



그림 1 유스케이스 다이어그램 예
Figure 1. A example of usecase diagram

다이어그램에서 유스케이스를 나타내기 위해서 유스케이스를 나타내는 타원, 유스케이스의 이름이 표현된다. 유스케이스를 표현하기 위해서 다음과 같은 요소를 사용하기로 한다.

- 유스케이스를 나타내는 타원
- 유스케이스 이름

다이어그램에서 항목간의 관계는 관계를 나타내는 선, 관계가 어느 쪽으로 향하는지에 대한 방향성 그리고 그 관계의 유형이 어떠한 것인지를 나타내는 레이블을 통해 표현된다. 이를 바탕으로 항목간의 관계를 표현하기 위해 다음과 같은 요소를 사용하기로 한다.

- 관계를 나타내는 선
- 관계의 방향성
- 레이블

어떠한 모양을 평면에 표시하기 위해서는 모양이 표현되기 위한 시작점과 모양의 공간을 나타내기 위한 너비와 높이를 필요로 한다. 유스케이스 다이어그램 또한 평면에 유스케이스와 관련된 항목들이 표현되는 것으로 각 항목 또한 시작점과 높이, 너비를 필요로 한다. 따라서 각 항목에서 공통적으로 적용해야 하는 항목은 다음과 같다.

- 시작점(x, y 좌표)
- 높이
- 너비

다이어그램 표현을 위해 설정한 액터, 유스케이스, 관계, 공통 항목 정보는 그림 2과 같다. 다이어그램 표현에 필요한 정보는 액터 정보, 유스케이스 정보, 관계 정보로 설정하였으며, 각각은 공통 정보인 시작점, 높이, 너비를 가지도록 하였다.

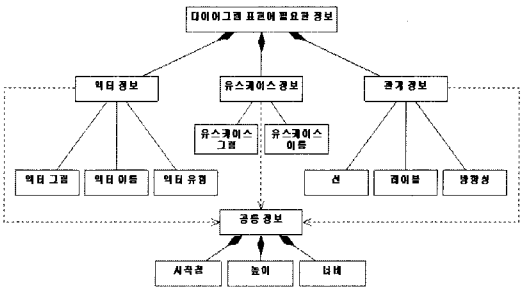



그림 2. 다이어그램 표현에 필요한 정보
Figure 2. The elements for representing usecase diagram

#### 2.3.2 기술서 작성에 필요한 정보 설정

유스케이스 기술서 작성을 지원하기 위해서는 기술서 작성에 필요한 요소를 식별해야 한다. 현재 기술서

작성에 대해서는 UML 명세에 정의되어 있지 않으며, 그래서 기술서를 작성하는 많은 방법이 존재한다. 본 논문에서는 현재 일반적으로 많이 사용되는 Cockburn이 제안한 기술서 항목[7]을 제공하기로 한다. 표 2는 Cockburn이 제안한 기술서 카탈로그를 나타낸 것이다. 본 논문에서는 기술서와 다이어그램 연결에 필수적인 이름 항목과 부수적인 범위, 수준, 주요액터만 사용하도록 한다.

표 2. Cockburn의 기술서 카탈로그
Table 2. Usecase description catalog suggested by Cockburn

| 항목 | 내용 |
|---|---|
| 이름 | 유스케이스의 이름 |
| 범위 | 설계중인 시스템 |
| 수준 | 사용자 목적 또는 세부기능 |
| 주요액터 | 목적을 수행하는 시스템의 서비스를 호출하는 주된 액터 |
| 이해관계자 및 관심사항 | 해당 유스케이스를 중요하게 생각하는 사용자와 이들이 원하는 사항 |
| 사전조건 | 유스케이스가 시작할 때 만족해야 하는 조건 |
| 성공보증 | 유스케이스가 성공적으로 끝났을 때 만족해야 하는 조건 |
| 주요 성공 시나리오 | 성공하기 위한 전형적이고 조건이 없는 경로의 시나리오 |
| 확장 | 성공이나 실패에 대한 대안 시나리오 |

### 2.4. 관계를 위한 연결 정보 설정 및 처리방법

유스케이스 다이어그램에서 액터와 유스케이스는 각각 독립된 형태로 의미를 가지지만 이들 간의 관계는 각 항목을 연결하는 역할을 한다. 따라서 항목간의 관계를 표현하기 위해서는 이에 대한 정보를 저장해야 한다. 이를 위해서 항목간의 연결 관계를 식별하고 각 노드와 관계 유지에 필요한 정보를 정의한다. 유스케이스 다이어그램은 다수개의 항목과 다수개의 관계로 구성된다. 하지만 관계로 기준으로 생각하면 다이어그램 상의 모든 항목은 그림 3과 같이 이항관계를 가진다고 볼 수 있다.

관계연결을 표현하는데 필요한 정보는 2.3절에서 식별한 관계의 유형, 레이블, 방향성과 함께 연결이 되는 두 항목에 대한 정보가 필요하다. 이러한 연결 정보를 각 항목에서 유지하도록 할 수 있다. 하지만 각 항목에서 연결 정보를 유지할 경우 양방향 연결을 위한 정보, 방향성을 위한 정보 등이 양쪽에서 모두 유지되어야 한다. 이렇게 두 개의 항목에서 정보를 유지할 경우 정보가 중복되기 때문에 일관성이 떨어지며, 이를 유지하기 위한 메커니즘이 복잡해진다. 따라서 본 논문에서는 연결을 하나의 독립된 형태로 구성을 하고 이를 관련된 항목에서 참조하도록 구성하도록 한다.

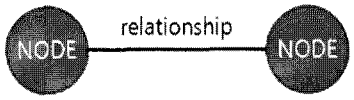



그림 3. 관계가 기준이 된 이항관계
Figure 3. Relationship centered binary connectivity

연결을 독립된 형태로 구성하기 위해서는 연결을 단순히 독립된 형태로 구성하는 것뿐만 아니라 관련 항목이 연결을 일관성 있게 참조하도록 이를 관리하기 위한 방법이 필요하다. 따라서 연결에 대한 정보를 모아서 관리하는 "연결 테이블"을 구성하고 항목에서는 연결 테이블을 참조하도록 한다. 연결 테이블에 유지하는 정보는 연결 유형 그리고 방향성과 연결된 항목을 위한 시작점(source)과 도착점(destination)으로 한다. 항목 간 연결이 추가될 경우에는 시작점, 도착점 그리고 연결 유형에 대한 정보를 연결 테이블에 추가하고 연결 테이블은 연결을 유일하게 식별할 수 있는 연결 ID를 부여한다. 그리고 연결과 관련된 항목은 연결 테이블이 부여한 연결 ID를 유지한다.

예를 들어 그림 4와 같이 노드 N1이 N2와 연관 관계를 가지고 노드 N2는 노드 N3과 포함 관계가 있다면 이에 대한 연결 테이블은 표 3과 같이 연결 ID가 link 1인 연결은 시작점이 N1이며 도착점이 N2이고 방향성은 N1에서 N2로 표현되며 연결 유형은 포함 관계가 있다는 정보를 가지게 된다.

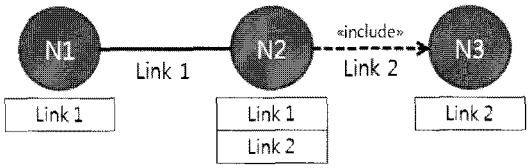



그림 4. 노드 구성의 예와 노드에 저장되는 연결 정보
Figure 4. Node configuration and maintenance link information

표 3. 연결 테이블 예
Table 3. Example of link table

| 연결 ID | 시작점 | 도착점 | 연결유형 |
|---|---|---|---|
| link 1 | N1 | N2 | 연관 |
| link 2 | N2 | N3 | 포함 |

그림 5는 본 논문에서 제시하는 관계를 기준으로 한 항목 간의 연결설정을 나타낸 것이다. 각 노드는 연결 테이블이 부여한 연결 ID를 저장하기 위한 속성(linkInfo)를 가져야 한다. 일관성있는 연결정보를 제공하기 위해서는 연결정보를 각 항목에 유지해서는 안 된다. 따라서 관계의 자체적인 속성을 가지는 연관 클래스를 이용하여 연결 정보를 유지하도록 하며, 연결 테이블은 이 연관 클래스를 이용하여 구성한다. 연관 클래스는 연결을 표현하는데 필요한 시작점, 도착점 그리고 연결 유형을 속성을 가진다.

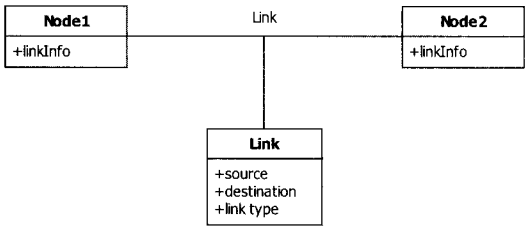



그림 5. 두 노드간의 연결정보 설정
Figure 5. Link information between two nodes

새로운 연결에 저장 절차는 우선 새로운 연결이 유효한지에 대한 유효성 검사를 수행한다. 다음으로 시작점과 도착점 노드에 대한 정보 그리고 연결 유형 정보를 이용하여 새로운 연결을 연결 테이블에 추가한다. 연결 테이블에서는 새로운 연결에 대한 연결 ID를 부여하고 관계가 있는 노드에게 연결 ID를 알려준다. 그림 6은 새로운 연결을 저장하는 알고리즘을 보인 것이다.

| | |
|---|---|
| Step 1. | 새로운 연결에 대한 유효성을 검사한다. |
| Step 2. | 시작점과 도착점 노드에 대한 정보를 이용하여 새로운 연결을 추가한다. |
| Step 3. | 새로운 연결에 연결 ID를 부여한다. |
| Step 4. | 관계가 있는 노드에 연결 ID를 할당한다. |

그림 6. 연결 정보 저장 알고리즘
Figure 6. Link information saving algorithm

노드에서 존재하는 연결에 대한 정보 회수 절차는 우선 노드에서 회수하고자 하는 연결에 대한 연결 ID를 확인한다. 그리고 연결 테이블에서 동일한 연결 ID를 검색을 하고 동일한 연결 ID가 발견될 경우 해당 연결 ID에 대한 정보를 회수한다. 그림 7은 연결 테이블에서 연결 정보를 회수하는 알고리즘을 나타낸 것이다.

| | |
|---|---|
| Step 1. | 노드에서 회수하고자 하는 연결 ID를 확인한다. |
| Step 2. | 연결 테이블에서 동일한 연결 ID를 가진 연결을 검색한다. |
| Step 3. | 연결 ID에 대한 정보를 회수한다. |

그림 7. 연결 정보 회수 알고리즘
Figure 7. Link information retrieving algorithm

### 2.5. 연결 유효성 검사

UML에서는 유스케이스 다이어그램에서 각 항목 간 가능한 관계를 엄격하게 정해두었다. UML 표준을 따르는 다이어그램을 작성하기 위해서는 각 항목 간의 연결을 검사 할 필요가 있다. UML에는 항목마다 가능한 관계의 유형이 정해져 있음으로 관계 유형과 항목의 유형을 비교함으로써 연결의 유효성을 검사할 수 있다. 이를 위해 UML 표준에 의거하여 관계 유형과 연결의 시작점과 도착점을 이용하여 가능 여부를 표 4와 같은 테이블 형태의 "연결 유효성 검사 테이블"로 구성한다. 두 항목 사이에서 연결이 발생했을 때 유효성 검사 테이블에서 관계 유형, 시작점, 도착점을 이용하여 하나씩 조건을 검사해서 유효성을 판단한다.

표 4. 연결 유효성 검사 테이블
Table 4. Link information verification table

| 관계 | 시작점 | 도착점 | 가능여부 |
|---|---|---|---|
| 연관 | actor | actor | X |
| | | usecase | O |
| | usecase | actor | O |
| | | usecase | X |
| 일반화 | actor | actor | O |
| | | usecase | X |
| | usecase | actor | X |
| | | usecase | O |
| 포함 · 확장 | actor | actor | X |
| | | usecase | X |
| | usecase | actor | X |
| | | usecase | O |

연결의 유효성 검사는 우선 요청된 연결의 유형을 확인하고 연결의 시작점과 도착점의 유형을 확인한다. 마지막으로 유효성 검사 테이블에서 연결 유형, 시작점, 도착점을 이용해서 가능한 연결인지를 확인 후 가능하다면 연결을 수립하고 불가능할 경우 오류를 발생시킨다. 그림 8은 유효성 검사를 위해 정의한 알고리즘을 나타낸 것이다.

| | |
|---|---|
| Step 1. | 요청된 연결 유형을 확인한다. |
| Step 2. | 시작점(source)의 유형을 검색한다. |
| Step 3. | 목적지(destination)의 유형을 검색한다. |
| Step 4. | 유효성 검사 테이블에서 결과를 확인하여 유효한 연결에 대해서는 새로운 연결을 수립하고, 유효하지 않은 연결에 대해서는 사용자에게 오류를 통보한다. |

그림 8. 연결 유효성 검사 알고리즘
Figure 8. Link verification algorithm

### 2.4. 다이어그램과 기술서의 연결

다이어그램과 기술서를 연결하기 위해서는 연결에 필요한 정보가 있어야 한다. 기술서는 유스케이스에 대한 여러 정보를 포함하고 있어 다이어그램으로 전환이 쉽지만 다이어그램에서 기술서로의 전환은 다이어그램에 표현되는 정보가 제한되어 있기 때문에 전환이 어렵다. 다이어그램에서 표현되는 유스케이스 정보는 유스케이스 이름으로 이를 이용해서 기술서와 연결을 해야한다. 그림 9는 본 논문에서 제시하는 다이어그램과 기술서의 연결을 처리하는 절차를 보인 것이다.

| | |
|---|---|
| Step 1. | 다이어그램에서 유스케이스 이름을 획득한다. |
| Step 2. | 유스케이스를 유지하는 저장소(repository)에서 이름과 일치하는 항목을 검색한다. |
| Step 3. | 동일한 이름의 유스케이스의 정보를 기술서의 형태로 표현한다. |

그림 9. 유스케이스 다이어그램과 기술서 연결 처리절차
Figure 9. Link processing algorithm between diagram and description

## III. 유스케이스 모델링 도구의 구현

본 장에서는 유스케이스 모델링 도구를 구현한 내용을 기술하며 그림 10은 시스템의 구현 내용을 클래스 다이어그램으로 표현한 것이다.

그림 10의 클래스의 설명은 다음과 같다. Data 클래스는 기술서에서 사용되는 요소와 다이어그램에서 사용되는 요소를 저장하는 클래스로써 각 항목의 공통사항인 이름과 다이어그램 상에 요소를 표현하기 위한 그래

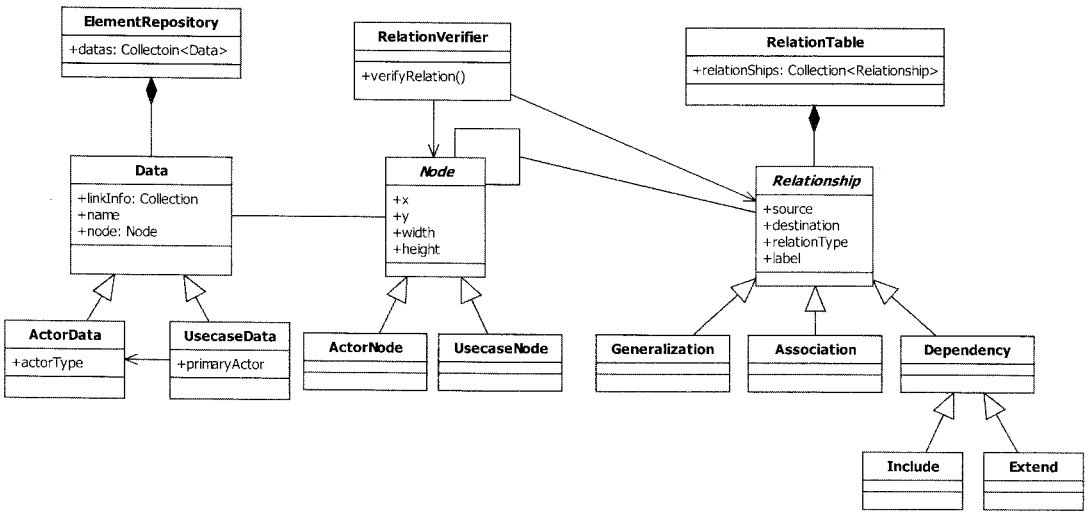



그림 10 유스케이스 모델링 도구의 구성
Figure 10. Configuration of the usecase modeling tool

픽정보의 참조 그리고 다이어그램 상의 관계정보를 집합형태로 저장한다. 이 클래스는 액터 정보를 저장하는 ActorData 클래스, 유스케이스 정보를 저장하는 Usecase Data가 상속 받는다. ElementRepository 클래스는 모델링 도구에서 사용되는 모든 Data 객체를 저장하는 역할을 한다. Node 클래스는 액터와 유스케이스 같은 항목의 다이어그램 표현에 필요한 정보를 저장하는 클래스로 x, y 좌표, 너비, 높이에 관한 정보를 가진다. 액터의 그래픽 표현에 사용되는 ActorNode와 유스케이스의 그래픽 표현에 사용되는 UsecaseNode 클래스가 이 클래스를 상속한다. Relationship 클래스는 다이어그램에서 두 항목간의 관계를 저장하는 클래스이다. Relationship 클래스는 관계의 시작점, 목적점 그리고 항목간의 관계유형에 대한 정보를 가진다. 본 클래스는 관계 유형별로 구분된 Generalization, Association, Dependency 클래스로 세분화되며, Dependency 클래스는 또 다시 Include, Extend 클래스로 세분화 된다. Relationship 클래스는 각 항목간의 관계에 대한 연관 클래스가 된다. RelationVerifier 클래스는 두 항목간의 연결의 유효성을 검사하는 클래스로 두 개의 항목과 관계유형을 이용하여 유효성을 검사한다. RelationTable은 다이어그램 상의 모든 관계를 저장하는 연결 테이블 역할을 하는 클래스이다. 테이블 구성을 위해 각각의 관계정보인 Relationship 클래스를 집합형태로 관리한다.

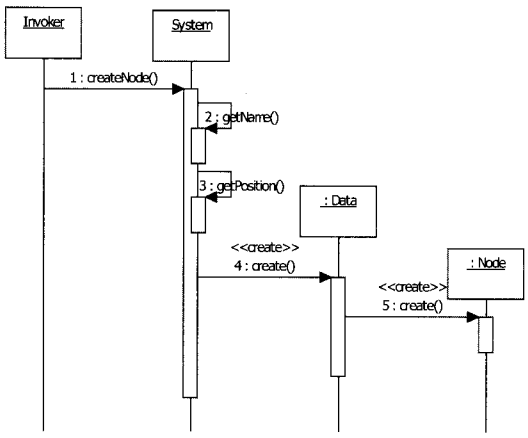



그림 11. 다이어그램 항목 생성 처리절차
Figure 11. Procedure of element creation

그림 11은 다이어그램에서 새로운 항목의 추가에 대한 처리과정을 나타낸 것이다. 새로운 항목 추가요청을 시스템이 수신하고 입력한 이름과 위치를 획득한다. 그 후 시스템은 이름, x, y 좌표를 이용하여 새로운 데이터 객체를 생성한다. 생성된 데이터 객체는 x, y 좌표를 이용하여 다이어그램에 자신을 표현하는 Node 객체를 생성한다.

그림 12는 다이어그램과 기술서의 접근을 보인 것이다. 기술서 연결요청을 시스템이 수신하고 시스템은 선택한 항목을 판단하고 항목 저장소에서 유스케이스 데이터를 획득하고 이를 이용하여 기술서를 보여준다.

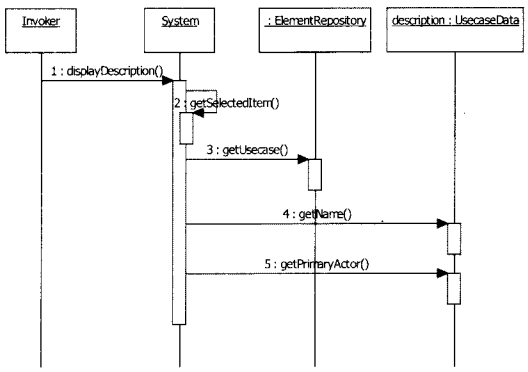



그림 12. 다이어그램과 기술서 접근 처리절차
Figure 12. Access processing between diagram and description

그림 13은 새로운 연결이 추가되었을 때 유효성 검사 테이블을 이용하여 연결의 유효성을 검사하는 절차를 나타낸다. 새로운 연결요청을 시스템이 수신하고 시스템은 연결요청의 대상인 각 데이터에서 그래픽 요소를 나타내는 Node 정보를 가져온다. 유효성을 검사하는 RelationVerifier 객체에게 두 노드의 정보와 연결이 요청된 관계의 유형을 전달하여 유효성 검사를 요청한다. RelationVerifier는 시스템이 전달한 요소를 바탕으로 유효성을 검사하여 그 결과를 시스템에게 알려준다. 시스템은 검사결과를 바탕으로 유효한 연결에 대해서는 두 항목에 linkID를 부여하고 유효하지 않은 연결에 대해서는 사용자에게 유효하지 않은 연결임을 알려준다.

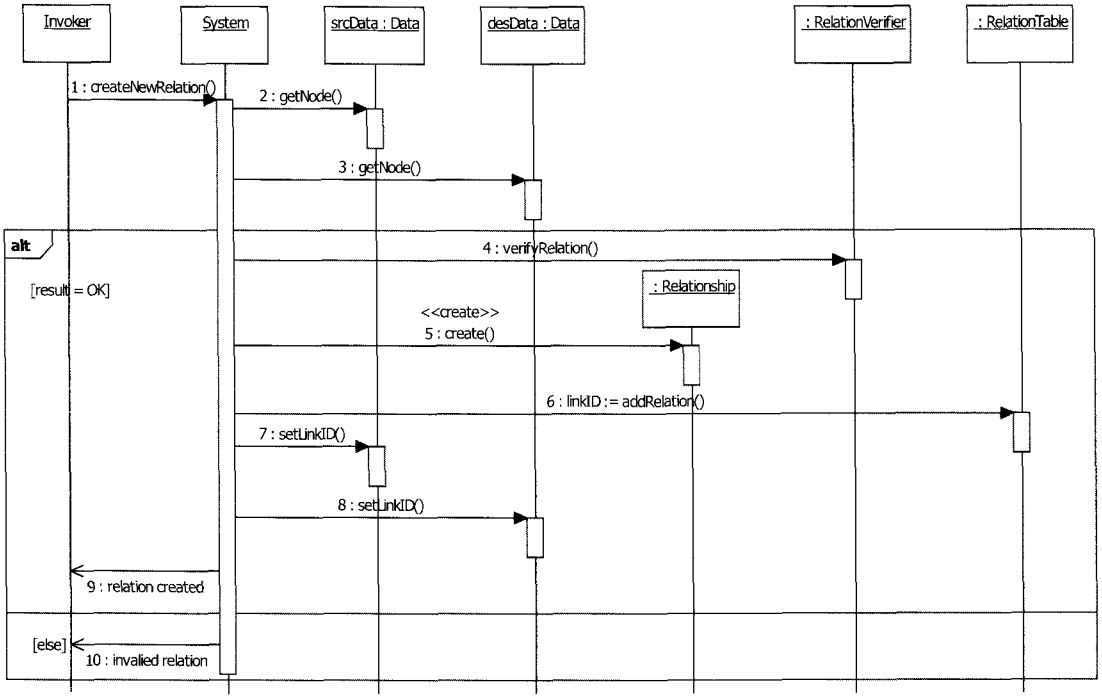



그림 13. 연결 유효성 검사 처리절차
Figure 13. Sequence of link verification

그림 14는 유스케이스 모델링 도구를 swing[8]을 이용하여 구현한 것이다. 기존의 모델링 도구는 모델에서 그림 형태로 변환을 제공했다[9]. 하지만 본 논문에서는 다이어그램 항목의 이동과 크기조정이 가능하도록 하였다. A영역은 다이어그램 작성에 필요한 유스케이스, 액터, 관계 명령을 포함하도록 구성한 것으로 명령을 이용하여 다이어그램을 작성할 수 있다. B 영역은 다이어그램과 기술서를 연결하는 명령을 포함하고 있으며, C 영역은 A 영역의 명령을 이용하여 다이어그램을 작성하는 부분이다. D는 B 영역에의 다이어그램과 기술서 연결을 보여주는 창으로 B 영역에서 선택한 유스케이스에 대한 기술서를 보여준다.

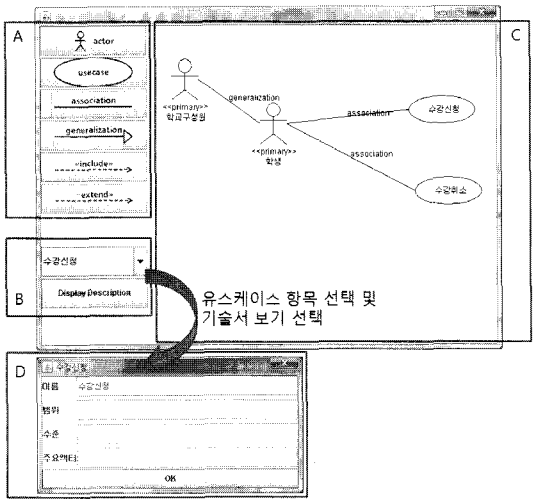



그림 14. 유스케이스 모델링 도구의 화면 구성 및 다이어그램 작성 예
Figure 14. GUI of the usecase modeling tool and example of usecase diagramming

## Ⅳ. 결 론

현재의 유스케이스 모델링 도구는 UML 명세에 정의된 다이어그램을 지원을 기본으로 하고 있다. 이 때문에 기술서를 작성하기 위해서는 다른 문서화 도구를 이용해야 하는 불편함이 따른다. 이렇게 다이어그램과 기술서가 분리된 형태로 모델링을 수행하면 상호간의 일관성이 결여되는 문제가 발생한다. 이러한 문제를 해결하기 위해서는 다이어그램과 기술서를 한 곳에서 작성할 수 있고 이들 간의 연결을 제공하는 모델링 도구가 요구된다. 본 논문에서는 유스케이스 모델링 도구를 제시하였다. 유스케이스 다이어그램 표현과 기술에 필요한 요소를 선정하였고, 다이어그램과 기술서의 연결방법을 제시하였다. 또한 다이어그램에서 항목간의 관계 연결을 유지하는 방법과 항목간의 연결의 유효성을 검사하는 연결 유효성 검사를 제시하였다. 연결의 효과적인 유지를 위하여 연결을 독립된 형태로 저장하고 관계를 가지는 항목들이 이를 참조하도록 하였으며, 유효성 검사를 위해 UML 명세를 기반으로 항목과 관계의 유형에 따라 결과를 확인할 수 있는 유효성 검사 테이블을 구성하였다 .마지막으로 모델링 도구 구현에 필요한 클래스와 이들 간의 연관관계와 모델링 도구 UI와 도구의 사용 예를 보였다. 향후 과제로는 유스케이스 모델링이 사용자의 요구사항을 추출하는 분석도구임으로 다수의 사용자가 공동으로 유스케이스 모델링을 할 수 있도록 하는 방법이 필요하다.

## 참고문헌


[1] Ivar Jacobson,. "Basic Use Case Modeling." *Report on Object Analysis and Design*, 1994.

[2] IBM, Rational Rose, www.ibm.com

[3] Borland, Together, www.borland.com

[4] Oracle, JDeveloper, www.oracle.com

[5] Craig Larman, *Applying UML and Patterns,* 3rd Ed., Prentice-Hall, 2005

[6] OMG Group, UML *specification*, www.omg.org

[7] Alstair Cockburn, *Writing Effective Use Cases,* Addison-Wesley, 2001

[8] Sun Microsystems, Java Standard Edition, java.sun.com

[9] Sellappan P., Louis L., "Web-based CASE Tool for Automated Rendering of UML Models," *International Journal of Computer Science and Network Security,* 2008


## 저자소개


최 환 복(Hwan-Bok Choi)

2008.2 안동대학교 컴퓨터공학과 공학사
2008.3 ~ 현재 안동대학교 컴퓨터공학과 석사과정

※관심분야 : 객체지향 분석/설계/프로그래밍

김 윤 호(Yun-Ho Kim)

1983. 2. 경북대학교 공학사
1993. 2. 경북대학교 공학석사
1997. 2. 경북대학교 공학박사
1997. 8 ~ 현재 안동대학교 전자정보산업학부 교수

※관심분야 : 인터넷 컴퓨팅, 객체지향 분석/설계/프로그래밍, 분산객체시스템, 병렬처리