인터넷 상의 공동작업 지원을 위한 UML CASE 도구

최환복* · 이은서** · 김윤호***

A UML CASE Tool for Collaboration Work on the Internet

Hwan-bok Choi* · Eun-ser Lee** · Yun-ho Kim***

요 약

본 논문에서는 원격지간에 수행되는 모델링을 지원하는 UML CASE를 제시하고자 한다. 공동작업 지원을 위해 필요한 기능을 설정하였으며, 기능에 기반하여 확장과 재사용성 증대를 위해 프레임워크 개념을 적용하고 이를 계층화시켰다. 또한 프레임워크를 기반으로 실제 UML CASE 도구를 구축하였다. 본 논문에서 제시하는 공동작업을 지원하는 UML CASE 도구는 여러 이해관계자의 의견을 실시간으로 반영하게 해줄 뿐만 아니라, 통합되고 일관성 있는 모델링에 기여할 것으로 기대된다.

ABSTRACT

This paper presents a UML CASE tool for collaboration work between remote place. It configured functions to support collaboration work and it applied framework concept to increase extension and reusability. It also constructed UML CASE tool based on framework. It result in not only reflecting various stakeholder's opinions but it also contributing to integrated and consistent modeling.

키워드

UML, 유스케이스 도구, 프레임워크, OOP, 객체지향 소프트웨어 시스템

Key word

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

* (주) 굿스플로 연구원
** 안동대학교 컴퓨터공학과 조교수
*** 안동대학교 컴퓨터공학과 정교수

접수일자 : 2009. 11. 26
심사완료일자 : 2010. 01. 06

Ⅰ. 서 론

CASE(Computer-Aided Software Engineering) 도구는 소프트웨어 개발을 구조화하고 제어하는데 있어 컴퓨터의 지원을 제공하는 소프트웨어이다. UML을 이용한 소프트웨어 모델링에서 컴퓨터의 지원을 위해 [1][2][3]와 같은 다양한 CASE도구들이 개발되었다. 하지만 시간이 지남에 따라 프로젝트의 규모가 커지고, 분산된 위치에서 이해관계자의 참여와 같은 환경의 변화로 공동작업 측면에서 기존 CASE 도구의 한계를 가져왔다.

기존 CASE 도구의 한계인 원격지 상의 공동작업을 지원하기 위해 [4][5][6]와 같은 연구가 제시되었다. 이들 연구는 인터넷을 기반으로 웹 브라우저 상에서 CASE 도구의 사용을 가능하게하고, 단일 저장소를 이용하여 공동작업이 가능하게 하였다. 하지만 이런 연구들은 특정 UML 다이어그램에 초점이 맞추어져 있어 확장과 재사용이 힘들다는 한계를 가지고 있다. 따라서 본 논문에서는 공동작업을 지원하며 확장과 재사용이 용이한 UML CASE 도구를 제안하고자 한다. 공동작업 지원을 위한 기능을 설정하고 확장과 재사용의 증대를 위해 기능을 프레임워크 형태로 구성한다. 또한 프레임워크를 바탕으로 실제 UML CASE 도구를 구축하고자 한다.

Ⅱ. 공동작업을 위한 프레임워크

2.1 프레임워크 설계를 위한 기능 설정

본 절은 프레임워크의 설계를 위해 프레임워크의 기능을 설정한 것을 기술한다. 프레임워크의 기능을 공동작업환경 제공과 UML에서 공통적으로 사용할 수 있는 요소로 나누어 기능을 설정한다. 프레임워크의 설계를 위해 설정한 기능을 표 1과 같다.

표 1. 프레임워크 설계를 위한 기능 설정
Table 1. Function configuration for framework design

구분	기능
공동작업환경 제공	실시간 다이어그램 연동
	책임관리
	작업 히스토리 관리
공통 UML 요소 제공	확장 가능한 UML 요소
	다이어그램 검증

공동작업 환경 제공은 원격지 상에서 수행되는 모델링을 지원하기 위한 기능으로 실시간 다이어그램 연동, 책임 관리, 작업 히스토리 관리와 같은 세부기능으로 구성된다.

실시간 다이어그램 연동 기능은 다이어그램 작성자가 원하는 시점에 다이어그램 공유를 요청하면 이를 원격지의 다른 클라이언트에 다이어그램을 공유해주는 기능이다. 기존에 제안된 연구는 웹 브라우저상에서 CASE 도구를 제공하고 공동 저장소를 이용하여 공동작업을 가능하게 하였다. 하지만 이들 연구는 다이어그램의 공유에만 집중하고 있어, 모델링 과정에서 여러 이해관계자들의 의견이 즉각적으로 반영되지 못한다. 이러한 문제를 해결하기 위해서 실시간으로 다이어그램을 공유할 수 있는 기능이 필요하다.

책임관리 기능은 다이어그램과 더불어 작성자 정보를 추가로 유지하여 누가 다이어그램을 작성했는지 관리하는 기능이다. 다수의 이해관계자들이 참여하는 공동 모델링의 경우 여러 사람들의 의견이 반영이 되지만 추후 모델에 대한 책임 관계를 명확하게 하지 않으면 모델에 대한 혼란을 야기할 수 있다. 따라서 이러한 문제를 해결하기 위해서는 책임 관리 기능이 필요하다.

작업 히스토리 관리 기능은 작성한 다이어그램에 대해 변경 사항을 유지하는 것이다. 객체지향 모델링은 순차적인 폭포수 모델과는 다르게 전체 프로세스 내에서 보다 나은 모델을 찾기 위해 지난 모델을 검토하는 유연한 모델링이 수행된다. 이러한 유연한 모델링을 지원하기 위해서는 모델의 변경을 체계적으로 유지하고 관리하는 히스토리 기능이 필요하다.

UML 공통 요소 제공은 UML CASE 도구 작성시 필요한 기능들을 추상화시켜 공통적인 요소를 제공하기 위한 기능으로 확장 가능한 UML 요소와 다이어그램 검증과 같은 세부기능으로 구성된다.

확장 가능한 UML 요소는 다이어그램 작성 시 사용하는 요소들에서 공통적인 부분에 대한 기능을 제공하기 위함이다. 다이어그램 검증은 사용자가 UML 스펙을 준수하여 다이어그램을 작성하는지 검증하는 기능으로 사용자가 임의대로 다이어그램을 작성하여 혼란을 야기할 수 있기 때문에 검증 기능이 필요하다.

2.2. 프레임워크 설계

본 절에서는 2.1절에서 정의한 기능을 바탕으로 프레

임워크를 설계한다. 프레임워크는 소프트웨어의 특정 클래스에 대해 재사용이 가능한 설계를 지원하는 협업 클래스의 집합으로 대표적인 예로 San Francisco 프레임워크[7]가 있다. San Francisco는 기능의 목적에 따라 계층화시켰다. 계층화를 통해 얻을 수 있는 이점은 기능별로 구분되어있기 때문에 확장과 재사용이 용이하다는 것이다. 따라서 프레임워크를 기능에 따라 계층으로 나누기로 한다.

기능정의를 바탕으로 계층화 시킨 프레임워크는 그림 1과 같다. Foundation 계층은 공동작업 환경을 지원하기 위한 기능을 제공하는 계층이다. Application Support 계층은 Application 계층의 지원 및 Application과 Foundation 계층 사이의 통신을 중재하는 기능을 수행한다. Application 계층은 UML CASE 도구에서 사용할 수 있는 공통적인 요소를 제공하는 계층이다.

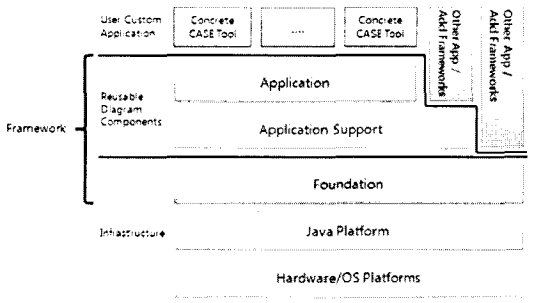

그림 1. 기능정의를 기반으로 구성한 프레임워크 계층
Figure 1. Framework layers based on function configuration

2.2.1 Foundation Layer

Foundation 계층은 프레임워크의 가장 하위에 위치하는 계층으로 공동작업을 수행할 때 필요한 서비스를 제공하는 계층이다. Foundation 계층에서 제공하는 서비스는 실시간 다이어그램 연동, 작성한 다이어그램의 책임관리, 작업 히스토리 관리이다. 그림 2는 Foundation 계층의 서비스를 제공하기 위해 구성한 클래스들이다.

CollaborationBroker는 Foundation 계층의 façade로써 상위 계층에서 필요로 하는 다이어그램 전송, 히스토리 검색과 같은 인터페이스를 제공한다. History 클래스는 원격지로 다이어그램 전송 또는 원격지에서 다이어그램이 도착했을 때 생성되는 클래스로 하나의 다이어그램과 이와 연관된 정보를 보관한다.

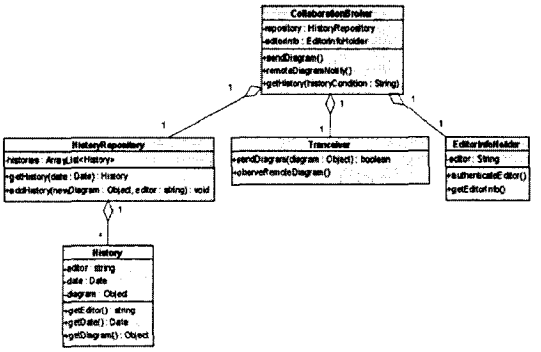

그림 2. Foundation 계층을 구성하는 클래스
Figure 2. Classes consisting Foundation Layer

HistoryRepository는 개별 History를 Collection 형태로 통합하여 유지하며, 히스토리 반환 요청 시 조건에 만족하는 히스토리를 반환한다. Tranceiver 클래스는 다이어그램을 원격지로 전송하며, 원격지에서 다이어그램이 도착했는지 감지하는 클래스로 원격지와의 통신을 담당한다. EditorInfo Holder는 현재 다이어그램 작성자의 정보를 유지하는 클래스로 책임관리를 위해 필요한 클래스이다.

2.2.2 Application Support Layer

프레임워크의 중간 계층인 Application Support 계층은 상위 계층인 Application 계층을 지원하고 Application 계층과 Foundation 계층을 연결하는 역할을 수행한다. Application Support 계층에서 제공하는 서비스는 Application 계층에서 요청한 다이어그램 공유에 필요한 정보를 생성하고, Foundation 계층으로 공유를 중개하며, Application 계층에서 작성한 다이어그램의 검증을 수행한다.

UML에서는 각각의 다이어그램에서 각 항목 간 연결할 수 있는 관계를 엄격하게 정해두었다. UML 표준을 따르는 다이어그램을 작성하기 위해서는 각 항목 간의 관계를 검사할 필요가 있다. 이를 위해 UML 표준에 의거하여 관계의 유형과 연결의 시작점과 도착점을 이용하여 가능 여부를 미리 설정하는 "연결 유효성 검사 테이블"을 적용하여 검증한다[8].

그림 3은 Application Support 계층의 서비스를 제공하기 위해 정의한 클래스이다.

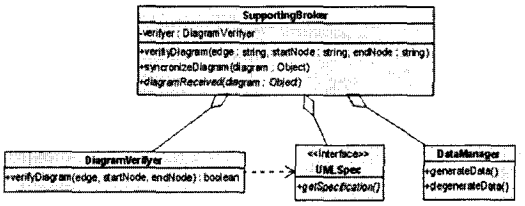

그림 3. Application Support 계층을 구성하는 클래스
Figure 3. Classes consisting Application Support Layer

SupportingBroker는 상위 계층인 Application 계층에서 내려오는 모든 요청을 받으며, 요청의 유형에 따라 하위 계층인 Foundation 계층으로 전달하거나 Application Support 계층에서 처리한다. DiagramVerifyer는 사용자가 작성한 다이어그램에 대해 UML 표준을 지키는지 검사하는 역할을 수행한다. 다이어그램 검증시 Diagram Verifyer는 UMLSpec을 참조하여 검증을 수행한다. UMLSpec은 다이어그램의 유효한 관계에 대한 정보를 가지는 연결 유효성 검사 테이블 역할을 수행한다. 다이어그램이 추가되었을 때 UMLSpec을 확장하여 연결 유효성 검사 테이블을 구성할 수 있다. DataManager는 원격지로 전송할 데이터의 생성 및 수신한 데이터를 분석한다. DataManager를 따로 구성한 이유는 원격지 공유시 다이어그램뿐만 아니라 작성자 정보와 같은 추가 정보가 필요하기 때문에 이를 묶어 전송하기 위해 구성하였다.

2.2.3 Application Layer

Application 계층은 제안하는 프레임워크에서 최상위에 위치하며 UML CASE 도구 개발에 필요한 확장 가능한 UML 요소를 제공한다. UML 요소를 제공하기 위해 정의한 클래스는 다음 표와 같으며, 이들간의 관계는 그림 4와 같다.

Node 클래스는 다이어그램 상의 노드를 표현에 필요한 인터페이스를 제공하는 클래스이다. 예를들면 유스케이스 다이어그램에서 액터와 유스케이스가 이에 해당한다. Edge 클래스는 노드와 노드를 연결하는 관계 표현에 필요한 인터페이스를 제공하는 클래스로써 예들들면 유스케이스 다이어그램에서 일반화, 포함, 확장 관계가 이에 해당한다. Node와 Edge 클래스는 각각 노드와 연결선을 공통적으로 묶을 수 있는 인터페이스를 제공하기 위해 정의하였다.

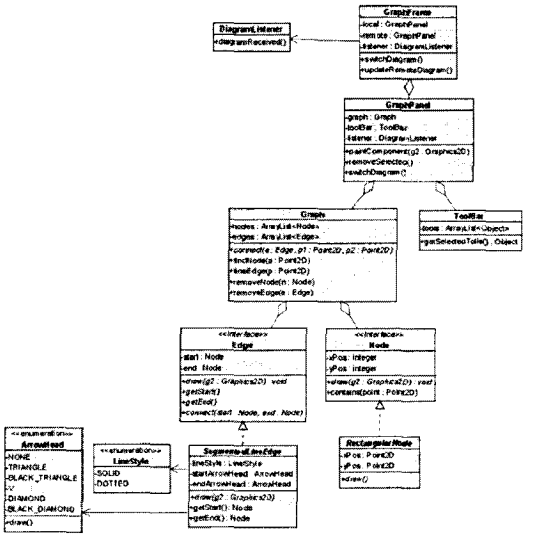

그림 4. Application 계층을 구성하는 클래스
Figure 4. Classes consisting Application Layer

ArrowHead 클래스는 관계의 화살표를 표현하는 클래스이며, LineStyle 클래스는 관계에서 선의 유형을 표현하는 클래스이다.

SegmentedLineEdge는 Edge 인터페이스를 실체화하는 추상 클래스로 연결선의 표현을 제외한 공통 메소드를 구현하고 있다. SegmentedLineEdge는 ArrowHead와 LineStyle을 이용하여 여러 형태의 관계를 표현할 수 있다. RectangularNode는 Node 인터페이스를 실체화하는 클래스로 다이어그램 상의 노드 정보를 포함하고 있다. RectangularNode 클래스 역시 노드의 형태표현을 제외한 모든 메소드가 구현되어 있다.

Graph 클래스는 다이어그램에 작성된 노드와 연결선을 유지하는 저장소 역할을 수행한다. GraphPanel은 실제 다이어그램이 표현되는 영역으로 작성한 다이어그램의 저장을 위해 Graph 클래스를 가진다.

GraphFrame은 ToolBar와 GraphPanel을 묶는 역할을 한다. 로컬에서 작성한 다이어그램의 표현과 원격지에서 전송된 다이어그램을 표현하기 위해 local과 remote 두 개의 GraphPanel을 가진다.

DiagramListener는 원격지에서 전송된 다이어그램이 있는지 감시하는 클래스로 만약 전송된 다이어그램이 있다면 GraphFrame에게 원격지간 공유되는 다이어그램을 갱신하도록 한다.

2.2.4 실시간 다이어그램 공유

본 절에서는 앞서 설계한 프레임워크의 클래스를 이용해 실시간으로 다이어그램이 공유되는 과정을 보인다. 그림 5는 실시간 다이어그램 공유에서 송신 측의 과정을 시퀀스 다이어그램을 이용해서 나타낸 것이다.

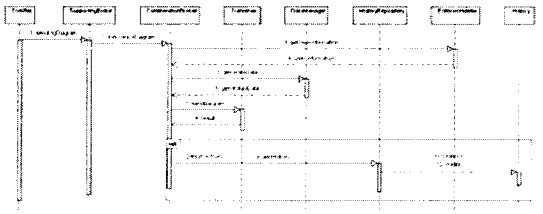

그림 5. 실시간 다이어그램 공유 순서 (송신)
Figure 5. Realtime diagram sharing sequence (sender)

다이어그램 작성자가 다이어그램을 작성 후 SupportingBroker를 통해 다이어그램 동기화를 요청한다. 요청을 받은 SupportingBroker는 원격지에 전송할 데이터 생성을 DataManager에게 요청하고 그 결과를 반환받는다. 그 후 SupportingBroker는 반환 받은 데이터를 Foundation 계층의 façade인 CollaborationBroker에게 전달한다. CollaborationBroker은 원격지로의 다이어그램 전송을 Tranceiver에게 요청한다. 성공적으로 요청이 수행되었으면 이를 저장하기 위해 HistoryRepository에게 히스토리 추가를 요청하고 HistoryRepository는 히스토리의 단위가 되는 History를 생성하고 collection 형태로 유지를 한다.

그림 6은 실시간 다이어그램 공유에서 수신측에서 수행되는 작업을 시퀀스 다이어그램을 이용해 나타낸 것이다.

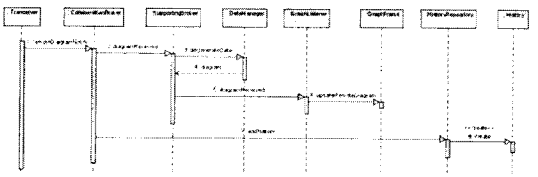

그림 6. 실시간 다이어그램 공유 순서 (수신)
Figure 6. Realtime diagram sharing sequence (receiver)

Tranceiver는 원격지에서 전송된 다이어그램이 있는지 계속 검사를 하고 만약 다이어그램이 전송되었다면 다이어그램 수신을 CollaborationBroker에게 알린다. 다이어그램을 전달받은 CollaborationBroker는 Application Support 계층의 SupportingBroker에게 다이어그램을 다시 전달한다. SupportingBroker는 Data Manager에게 추가 데이터와 다이어그램의 분리를 요청하고 그 결과로 다이어그램을 받고 추출한 다이어그램을 Application 계층의 DiagramListener에게 다이어그램을 전달한다. DiagramListener는 동기화가 있음을 감지하고 Graph Frame에게 원격지 다이어그램 갱신을 지시한다. 마지막으로 CollaborationBroker는 히스토리 관리를 위해 HistoryRepository에게 히스토리 추가를 요청한다.

Ⅲ. 프레임워크 기반의 CASE 도구의 설계 및 구현

본 장에서는 2장에서 설계한 프레임워크를 바탕으로 유스케이스 모델링 도구, 클래스 모델링 도구와 같은 실제 UML CASE 도구를 설계하고 구현한 내용을 기술한다. UML CASE 도구는 Java의 swing[9]을 이용해 구현하였으며 원격지간의 통신은 TCP/IP를 이용해 구성하였다.

3.1 유스케이스 모델링 도구

3.1.1 유스케이스 다이어그램 구성요소의 식별

유스케이스 모델링은 시스템의 기능적인 요구사항을 이끌어내고 시스템과 시스템을 사용하는 사용자간의 통상적인 교류를 기술하여 시스템이 어떻게 사용되는지를 표현하는 분석기술이다. 유스케이스 모델링 CASE 도구를 설계하기 위해서는 먼저 다이어그램에 사용되는 요소를 식별해야 한다. 그림 7은 유스케이스 다이어그램의 예이다.

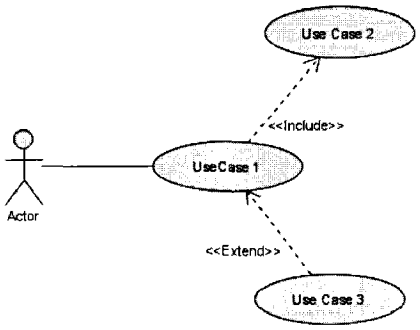

그림 7. 유스케이스 다이어그램의 예
Figure 7. An example of usecase diagram

그림 7에서 보는 바와 같이 유스케이스 다이어그램은 유스케이스와 액터, 그리고 유스케이스와 유스케이스 간의 관계를 표현한다. UML의 유스케이스 다이어그램에서 액터와 유스케이스의 표기법은 표 2와 같다.

표 2. 유스케이스 다이어그램을 구성하는 노드
Table 2. Nodes for usecase diagram

노드	표기법
액터	Actor Name
유스케이스	Use Case Title

액터는 시스템 사용의 주체로써 선을 이용하여 사람을 표현하고 아래쪽에 액터의 이름이 위치한다. 유스케이스는 시스템이 제공해야 하는 기능을 의미하며 타원으로 유스케이스를 표현하고 타원 내부에 유스케이스 명을 표기한다.

UML은 유스케이스 다이어그램에서 요소들 간에 사용할 수 있는 관계를 정해두었다. 다이어그램 작성과 검증을 위해 관계 또한 식별해야 한다. 식별한 관계는 표 3과 같다.

연관은 액터와 유스케이스의 관련이 있음을 의미하며, 액터와 유스케이스 사이에 실선으로 표시된다. A항목에서 B항목에서의 일반화는 B항목이 A항목의 특수화를 의미하며 부모쪽으로 향하는 속이 비고 닫힌 화살표와 실선으로 표시된다. A항목에서 B항목으로의 포함관계는 B항목에 명시된 행위를 A항목이 포함하는 것을 의미하며, 기반 항목으로 향하는 화살표와 점선 그리고 «include» 레이블로 표시된다. A항목에서 B항목으로 확장관계는 B항목이 A항목에 명시된 행위로 인해 증대되는 것을 의미하며, 기반 항목으로 향하는 화살표와 점선 그리고 «extend» 레이블로 표시한다.

표 3. 유스케이스 다이어그램을 구성하는 관계
Table 3. Relationships of usecase diagram

관계	표기법
연관	Actor — Use Case
일반화	Parent ◁— Child
포함	Use Case 1 <<Include>> Use Case 2
확장	Use Case 1 <<Extend>> Use Case 2

3.1.2. 유스케이스 모델링 도구의 설계 및 구현

식별한 유스케이스 다이어그램의 요소를 바탕으로 프레임워크에 기반하여 설계한 내용은 그림 8과 같다. 붉은색 영역은 프레임워크 부분을 나타낸다.

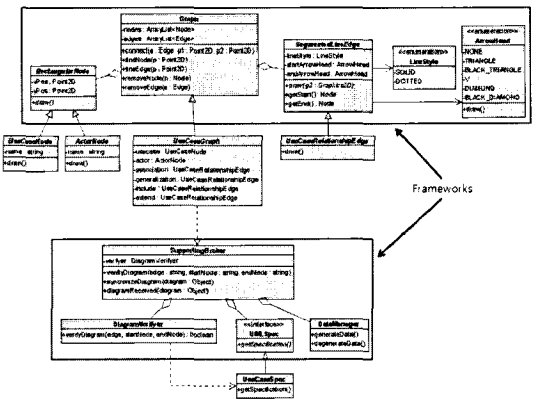

그림 8. 프레임워크 기반의 유스케이스 모델링 도구의 설계
Figure 8. A design of usecase modeling tool based on framework

프레임워크의 RectangularNode를 이용하여 Actor Node와 UseCaseNode를 정의하여 다이어그램상의 액터와 유스케이스를 표현하도록 하였다. 또한 Segemented LineEdge를 상속받아 UseCaseRelationshipEdge를 정의하여 유스케이스 다이어그램에서 사용할 수 있는 관계를 제공한다. UseCaseGraph는 유스케이스 다이어그램을 작성할 수 있는 영역을 제공하는 클래스로 프레임워크의 Graph 클래스를 상속 받아 정의하였다. Use CaseGraph 클래스에는 유스케이스 다이어그램에서 사용하는 액터, 유스케이스, 연관, 확장, 포함, 일반화 관계를 가지고 있다. UseCaseSpec 클래스는 UMLSpec 클래스를 상속받았으며 유스케이스 다이어그램을 검증하는 유스케이스 다이어그램 유효성 검사 테이블을 가지도록 하였다. 유스케이스 다이어그램을 검증하기 위해 [8]에서 사용한 연결 유효성 검사 테이블은 표 4와 같다.

표 4. 유스케이스 다이어그램 검증을 위한 연결 유효성 검사 테이블
Table 4. Relationship verification table for valid usecase diagram

관계	시작점	도착점	가능여부
연관	Actor	Actor	X
		Use Case	O
	Use Case	Actor	O
		Use Case	X
일반화	actor	Actor	O
		Use Case	X
	Use Case	Use Case	X
		Use Case	O
포함・확장	actor	Actor	X
		Use Case	X
	Use Case	Actor	X
		Use Case	O

설계를 바탕으로 구현한 유스케이스 CASE 도구는 다음 그림9와 같다. A영역은 메뉴바로써 유스케이스 다이어그램 작성에 필요한 노드와 관계를 포함하고 있다. B영역은 다이어그램 작성 영역으로 A영역의 메뉴를 이용해 작성한 다이어그램이 표현된다. B영역은 로컬에서 작성한 다이어그램을 표현하는 B-1영역과 원격지에서 전송한 다이어그램을 표현하는 B-2로 구성되어 있다.

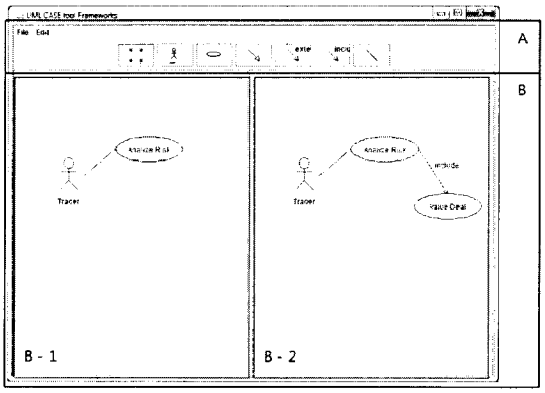

그림 9. 유스케이스 모델링 도구의 구현
Figure 9. An implementation of usecase modeling tool

3.2 클래스 모델링 도구

3.2.1 클래스 다이어그램 구성요소 식별

클래스 다이어그램은 시스템을 구성하는 요소들과 이들의 관계를 정적으로 표현하는 다이어그램이다. 클래스 다이어그램은 클래스, 인터페이스와 이들간의 관계를 표현한다. 클래스 다이어그램 CASE 도구의 설계를 위해서는 우선 다이어그램에서 사용되는 UML 요소를 식별해야 한다.

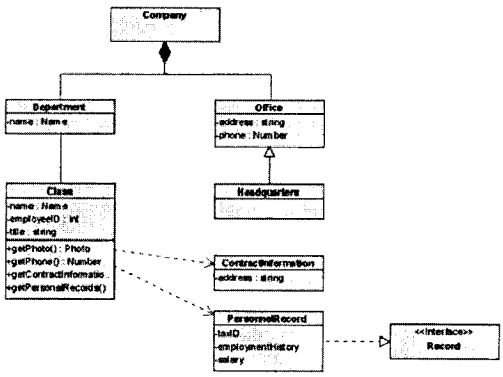

그림 10. 클래스 다이어그램의 예
Figure 10. An example of class diagram

그림 10에서 보는 바와 같이 클래스 다이어그램은 클래스와 클래스, 인터페이스와 인터페이스 그리고 클래스와 인터페이스 간의 관계를 표현한다. UML의 클래스 다이어그램에서 클래스와 인터페이스의 표기법은 표 5와 같다.

표 5. 클래스 다이어그램을 구성하는 노드
Table 5. Nodes for class diagram

노드	표기법
클래스	Class Name -attribute1 #attribute2 +attribute3 -operation() #operation2() +operation3()
인터페이스	<<Interface>> Interface Name -attribute1 #attribute2 +attribute3 -operation1() #operation2() +operation3()

클래스는 사각형으로 표현하며 클래스 이름, 속성, 오퍼레이션을 표기하는 영역으로 나누어진다. 인터페이스는 클래스의 특수한 형태로 인터페이스의 표현과 구획 구성은 클래스와 동일하지만, 클래스의 이름 영역에 인터페이스를 뜻하는 «interface» 레이블이 추가적으로 표현된다. 클래스와 인터페이스의 속성과 오퍼레이션에는 가시성 표기가 가능하며 +는 private, #은 protected, +는 public 가시성을 의미한다.

UML은 클래스 다이어그램에서 요소들 간에 사용할 수 있는 관계를 정해두었다. 다이어그램 작성과 검증을 위해 관계 또한 식별해야 한다. 클래스 다이어그램에서 사용할 수 있는 관계는 표 6과 같다.

표 6. 클래스 다이어그램을 구성하는 관계
Table 6. Relationships of class diagram

관계	표기법
연관 (association)	A — B
집합 (aggregation)	A ◇— B
합성 (composition)	Class 1 ◆— Class 2
의존 (dependency)	Class 1 ⇢ Class 2
일반화 (generalization)	Class 1 —▷ Class 2
실체화 (realization)	<<Interface>> Interface 1 ⇢▷ Class 2

연관은 클래스와 클래스를 연결하는 구조적인 관계로써, 동일한 클래스 혹은 서로 다른 클래스를 연결하는 실선으로 표현한다. 집합은 전체에 속하는 객체가 부분에 속하는 객체를 갖는 관계를 나타낸다. 집단화는 연관의 특수한 형태로 연관의 전체 부분에 흰색 다이아몬드(hollow diamond)를 붙여서 표현한다. 합성은 연관의 특수한 경우로 집합의 변경이다. 전체에 대한 부분의 강한 소유권과 동일한 생명주기를 의미한다. 합성은 전체 부분에 흑색 다이아몬드(filled diamond)를 붙여서 표현한다. 의존은 한 요소의 정의에 대한 변경이 다른 요소에 대한 변경을 유발할 때 두 요소간에 존재하는 관계이다. 의존 관계는 사용하는 클래스(소스)에서 사용되는 클래스(타킷)으로 향하는 방향성을 가지는 점선으로 표현된다. 일반화는 일반적인 클래스(base class)와 세부적인 클래스(sub class)간의 관계이다. 일반화는 일반적인 클래스로 향하는 방향성을 갖는 실선으로 표현하며, 실선의 끝에는 삼각형의 화살표를 붙인다. 실체화는 인터페이스에 명시된 오퍼레이션을 클래스가 실제 구현하는 관계를 의미한다. 실체화는 클래스에서 인터페이스로 방향성을 가지는 점선으로 표현하며, 끝에는 삼각형의 화살표를 붙인다.

3.2.2. 클래스 모델링 도구의 설계 및 구현

식별한 클래스 다이어그램의 요소와 관계바탕으로 프레임워크에 기반하여 설계한 내용은 그림 11과 같다. 붉은색 영역은 프레임워크 부분을 나타낸다.

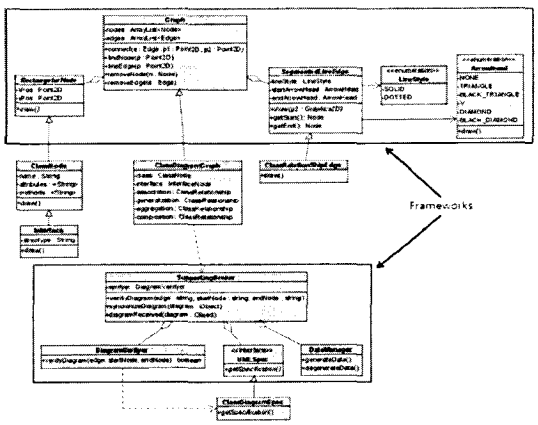

그림 11. 프레임워크 기반의 클래스 모델링 도구의 설계
Figure 11. A design of class modeling tool based on framework

RectangularNode를 상속받아 다이어그램에서 클래스를 표현하는 ClassNode를 정의 하였다. 클래스 다이어그램에서 인터페이스는 스테레오타입을 가지는 클래스의 특수한 형태라고 할 수 있다. 따라서 ClassNode를 상속받고 스테레오타입을 저장하는 속성을 추가하였다. ClassRelationshipEdge는 클래스 다이어그램에서 사용할 수 있는 관계들을 일반화 시킨 클래스로 관계들을 묶어서 관리하기 위해 정의하였다. ClassDiagramGraph는 클래스 다이어그램에서 사용되는 요소들을 저장하기 위한 저장소이다. ClassDiagramGraph에는 클래스, 인터페이스와 같은 노드와 연관, 일반화와 같은 관계를 포함하고 있다. ClassDiagramSpec은 클래스 다이어그램을 검증하기 위한 연결 유효성 검사 테이블 역할을 수행한다. 클래스 다이어그램을 검증하기 위해 정의한 연결 유효성 검사 테이블을 표 7과 같다.

표 7. 클래스 다이어그램 검증을 위한 연결 유효성 검사 테이블
Table 7. Relationship verification table for valid class diagram

관계	시작점	도착점	가능여부
연관	Class	Class	O
		Interface	O
	Interface	Class	O
		Interface	O
일반화	Class	Class	O
		Interface	X
	Interface	Class	X
		Interface	O
실체화	Class	Class	X
		Interface	O
	Interface	Class	X
		Interface	O
집합·조합	Class	Class	O
		Interface	X
	Interface	Class	O
		Interface	X

설계를 바탕으로 구현한 클래스 다이어그램 CASE 도구는 그림 12과 같다. A영역은 메뉴바로써 클래스 다이어그램 작성에 필요한 노드와 관계를 포함하고 있다. B영역은 다이어그램 작성 영역으로 A영역의 메뉴를 이용해 작성한 다이어그램이 표현된다. B영역은 로컬에서 작성한 다이어그램을 표현하는 B-1영역과 원격지에서 전송된 다이어그램을 표현하는 B-2영역으로 구성되어 있다.

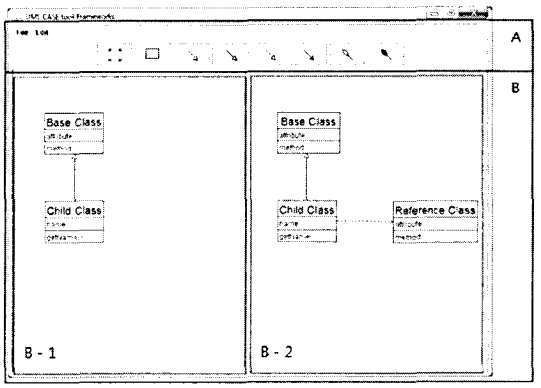

그림 12. 클래스 모델링 도구의 구현
Figure 12. An implementation of Class modeling tool

Ⅳ. 결 론

UML이 발표된 후 이를 활용한 소프트웨어 모델링에서 컴퓨터의 지원을 제공하기 위해 다양한 UML CASE 도구가 개발되었다. 하지만 지리적으로 떨어진 원격지에서 이해관계자들의 참여와 같이 모델링 환경의 변화로 인해 공동작업 측면의 한계를 드러냈다. 공동작업을 지원하기 위해 다양한 연구가 제시되었지만 특정 다이어그램에 초점이 맞추어져 있어 도구의 확장과 재사용이 어렵고 또한 실시간 공유를 지원하지 못해 이해관계자의 의견을 즉각적으로 반영하지 못한다. 본 논문에서는 실시간 의견 반영이 가능한 공동작업을 지원하는 UML CASE 도구를 제안하였다. 공동작업 지원을 위해 실시간 다이어그램 연동, 작업 히스토리 관리, 책임관리 기능을 정의하였고, CASE 도구 개발 시 필요한 공통 UML 요소와 다이어그램 검증 기능을 설정하고 확장과 재사용성의 증대를 위해 프레임워크를 구성하였다. 마지막으로 프레임워크를 기반으로 공동작업을 지원하는 유스케이스 모델링 도구와 클래스 모델링 도구를 보였다. 공동작업을 지원하는 UML CASE 도구는 여러 이해관계자의 의견을 반영하게 해줄 뿐만 아니라 통합되고 일관성 있는 모델링에 기여할 것으로 기대된다.

참고문헌

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

[2] Visual Paradigm International, Visual Par-adigm for UML, www.visual-paradigm.com

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

[4] Cao, S., Grundy, J., Hosking, J., Stoeckle, H., Tempero, E. and Zhu, N., "Generating Web-based User Interfaces for Diagramming Tools", *Proceedings of the Sixth Australasian conference on User interface,* Vol. 40, pp63-72, 2005

[5] Khaled, R., Mackay D., Biddle, R., Nobble, J., "A Lightweight Web-Based Caase Tool for Sequence Diagrams", *Proceedings SIGCHI-NZ Symposium on Computer-Human Interaction,* pp.55-60, 2002

[6] Mackay, D., Noble, J., Biddle, R., "A Lightweight Web-Based case tool for class diagrams", *Proceedings of the Fourth Australasian user interface conference on User interfaces,* Vol. 18, pp95-98, 2002

[7] IBM, "San Francisco Frameworks", *IBM Systems Journal,* Vol.37 No.2, 1998

[8] 최환복, 김윤호, "유스케이스 모델링 도구의 설계 및 구현", 한국해양정보통신학회 논문지, Vol.13 No.5, 2009

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

저자소개

최환복(Hwan-Bok Choi)

2008 안동대학교 컴퓨터공학과 공학사

2010 안동대학교 컴퓨터공학과 공학석사

2010.3-현재 (주) 굿스플로 연구원

※관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 리팩토링 등

이은서(Eun-Ser Lee)

2001~현재 ISO/IEC 15504 국제 심사원

2004년 중앙대학교 컴퓨터공학과 (박사)

2004년-현재 임베디드 산업협회 전문 위원

2004년-현재 한국정보통신기술협회 위원

2005년-2007년 숭실대학교 정보미디어기술연구소 연구 교수

2008년-현재 안동대학교 컴퓨터공학과 조교수

※관심분야 : CBD, Formal method, Quality model, SPI(Software Process Improvement), Defect Management 등

김윤호(Yun-Ho Kim)

1983 경북대학교 전자공학과 공학사

1993 경북대학교 대학원 컴퓨터공학과 공학석사

1997 경북대학교 대학원 컴퓨터공학과 공학박사

1997-현재 안동대학교 전자정보산업학부 교수

※관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 병렬처리 등