MVC 디자인 패턴에 기반한 클래스 다이어그램 저작도구의 설계

김재훈* · 김윤호**

The Design of a Class Diagram Authorization Tool based on the MVC Design Pattern

Jae-Hoon Kim* · Yun-Ho Kim**

요 약

본 논문에서는 MVC 패턴에 기반한 클래스 다이어그램 저작도구의 설계와 구현을 제시하고자 한다. 클래스 다이어그램 저작도구의 설계는 ClassInformation, ScreenDisplay, ToolManager에 대한 정의를 하고 구조를 기술한다. ClassInformation은 다이어그램의 정보를 가공하거나 처리하는 부분을 담당한다. ScreenDisplay는 저작도구의 화면을 구성하기 위해서 GUI를 담당한다. ToolManager는 저작도구의 입출력 처리위해 이벤트 처리를 담당한다. MVC 패턴을 바탕으로 저작도구의 ClassInformation, ScreenDisplay, ToolManager는 각각의 역할이 독립적으로 나눠져 있고, 결합도를 낮추기 때문에 새로운 기능 추가에 유연하게 대처한다.

ABSTRACT

This paper suggests a implements and a design of class diagram authorization tool based on the MVC pattern. It defines and descries the structure of ClassInformation, ScreenDisplay and ToolManager. ClassInformation is responsible for processing or handling information of a diagram. ScreenDisplay is responsible for GUI to configure the screen of the authorization tool. ToolManager is responsible for event handling to process I/O of the authorization tool. Based on MVC pattern, ClassInformation, ScreenDisplay and ToolManager of the authorization tool are assigned each role independently. It is flexible to new requirement, because of loose coupling.

키워드

UML, 클래스 다이어그램, MVC 패턴, 옵저버 패턴, 결합도

Key word

UML, Class Diagram, MVC Pattern, Observer Pattern, Coupling

* 안동대학교 컴퓨터공학과 석사과정(zoom50210@gmail.com)
** 안동대학교 컴퓨터공학과 정교수

접수일자 : 2010. 09. 16
심사완료일자 : 2010. 11. 29

Ⅰ. 서 론

UML은 OMG(Object Management Group)에서 표준으로 채택한 통합 모델링 언어이다.[1-3] 그리고 1980년대 후반부터 1990년대 초반까지 많은 객체지향 모델링 언어들이 쏟아졌는데 각각의 사용법과 정의가 다르기 때문에 상호 의사소통이 혼돈스러웠다. 시스템의 구조를 표현하고 모델링 개념에서 가장 큰 영향을 미치기 때문에 UML에서 Class Diagram은 사용빈도수가 높다.

디자인 패턴은 반복되는 문제들에 대한 솔루션을 기술하는 것이다.[4-5] 시스템을 개발 할 때, 검증된 패턴을 적용하면 솔루션을 재사용하므로 시간을 단축시킬 수 있고 시스템이 유연해진다.

그리고 소프트웨어는 항상 요구사항이 변경되고 새로운 기능을 추가하게 된다. 하지만 새로운 기능을 추가하는 것은 각각의 컴포넌트로 분리되어 있지 않아 소프트웨어를 변경하는 것이 힘들다. 따라서 본 논문에서는 새로운 기능 추가를 유연하게 대처하고자 컴포넌트로 분리되어 독립적으로 역할을 수행하는 MVC 패턴에 기반하여 클래스 다이어그램 저작도구의 ClassInformation과 ScreenDisplay 그리고 ToolManager의 역할을 기술하고 설계를 제시한다.

Ⅱ. 관련 연구

현재 UML을 이용하여 시스템을 개발할 때 사용자와 개발자간에 의사소통을 원활하게 하기 위해서 UML 저작도구를 사용하고 있다. 그리고 많은 연구가 되고 있다[6-9].

그림 1은 본 연구와 관련된 연구들을 비교하였다. 기존의 NutCase, Pounamu/Thin, BlueJ, Violet 도구들은 UML의 표준을 따르며 클래스 다이어그램을 지원한다. 하지만 다이어그램을 표하는 방법에서 attribute와 operation 지원, 복수개의 attribute와 operation 지원, 그리고 visibility 지원하지 않는 점과 관계 표현에서 몇 개의 관계만을 지원하는 도구들이 있다. 이러한 문제점을 지원하기 위해서 본 연구의 클래스 다이어그램 저작도구 설계를 제안한다.

	NutCase	Pounamu /Thin	BlueJ	Violet
Class Diagram 지원	O	O	O	O
Attribute, Operation 지원	O	O	X	O
Multi – Attribute, Operation 지원	O	O	X	X
Visibility 지원	X	X	X	Attribute, Operation Visibility 지원
Relationship 지원	6 Relationship (관계 모두 지원)	1 Relationship (Association 지원)	2 Relationship (Dependence, Generalization 지원)	6 Relationship (관계 모두 지원)

그림 1. 관련 연구 비교
Figure 1. A Compares Related the Research

MVC패턴은[4] Model-View-Controller 부분으로 나눠서 각각의 역할을 독립적으로 수행하기 때문에 결합도가 낮고 응집력이 높은 장점이 있다. MVC 패턴을 적용함으로 얻을 수 있는 이점은 다음과 같다.

- 새로운 기능을 쉽게 추가할 수 있다.
- 소프트웨어의 유연성이 증가한다.
- 일관성 있고 컴포넌트화 된 코드가 된다.
- 각각의 컴포넌트 응집력이 높아진다.
- 유지보수에 유연하게 대처할 수 있다.
- 검증된 패턴이므로 소프트웨어의 안정성이 높아진다.

Ⅲ. 클래스 다이어그램 저작도구의 구성

본 장은 저작도구의 ClassInformation, ScreenDisplay, ToolManager의 구성을 MVC 패턴의 Model, View, Controller에 기반하여 기술하였다. 다이어그램의 정보를 저장하고 가공하기위해서 ClassInformation (Model)로 설정하였다. 그리고 저작도구의 화면에서 다이어그램을 표현하기 위해서 ScreenDisplay (View)로 설정하였다. 마지막으로 저작도구에서 다이어그램의 정보를 입력하고 출력하는 과정에서 발생하는 이벤트 처리를 위해 ToolManager (Controller)로 설정하였다.

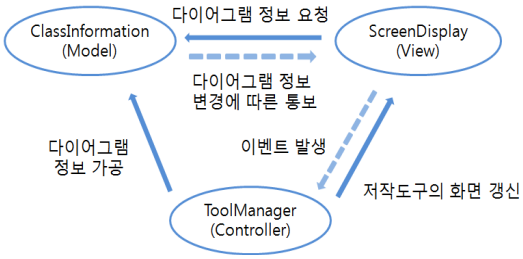

그림 2. 클래스 다이어그램 저작도구의 구조
Figure 2. The Structure of The Class Diagram Authorization Tool

ClassInformation는 클래스와 관계의 정보를 저장하고 가공하는 컴포넌트이다. View에서 Model의 데이터를 화면에 표현하고, Controller는 데이터의 흐름을 관리를 하며, 각각 독립적으로 수행하기 때문에 ScreenDiaplay와 ToolManager에게 의존하지 않고, 다이어그램의 정보처리를 수행하도록 하였다. 다이어그램 정보가 변경되면 다이어그램을 갱신하기 위해서 ScreenDisplay에게 다이어그램 변경통보를 한다.

ScreenDisplay는 다이어그램의 정보를 시각적인 요소로 처리하는 GUI 컴포넌트이다. View는 화면에 Model의 데이터만 표현하기 때문에 ScreenDisplay에서 다이어그램의 정보를 직접 가공하거나 제어하지 못하도록 하였다. 그리고 View는 Model에게 데이터의 상태를 요청할 수 있기 때문에 ScreenDisplay에서 ClassInformation에게 다이어그램의 정보를 요청할 수 있고, 반환받은 정보를 표현하도록 하였다. 입력된 다이어그램의 정보를 가공하거나 저장하기 위해서 ScreenDisplay는 ToolManager에게 이벤트 요청을 한다.

ToolManager는 ScreenDisplay와 ClassInformation 사이에서 다이어그램의 정보 가공 또는 이벤트가 발생하는 것을 관리하는 컴포넌트이다. Controller는 데이터의 흐름을 관리하기 때문에 ToolManager가 ScreenDisplay로부터 요청받은 이벤트를 파악하고, 다이어그램 정보를 가공하는 요청이면 ClassInformation에 그 정보를 전달하도록 하였다. 그리고 ScreenDisplay에서 요청한 이벤트가 저작도구의 화면 상태를 변경하는 경우에는 ClassInformation이 ScreenDisplay에게 화면갱신을 요청하도록 하였다.

Ⅳ. 클래스 다이어그램 저작도구의 설계

4.1 ClassInformation

ClassInformation 컴포넌트는 다이어그램의 정보를 가공하고 저장한다. 변경된 다이어그램의 정보를 화면에 나타내기 위해서 ScreenDisplay 컴포넌트에게 변경되었다고 통보한다. 그리고 요청받은 ScreenDisplay 컴포넌트는 저작도구의 화면 상태를 변경한다. 저작도구의 ClassInformation 컴포넌트 요약과 구조는 표 1, 그림 3과 같다.

표 1. ClassInformation 컴포넌트 요약
Table 1. The Summary Component ClassInformation

클래스이름	클래스역할
ModelInterface	ClassInformation 컴포넌트의 인터페이스 정의
Model	ModelInterface의 실체화이며, 다이어그램의 정보, 파일의 정보를 가공하고 처리
Inforamtion Storage	클래스와 관계와 파일의 정보를 저장하는 저장소
Class Inforamtion	클래스의 정보를 저장하기 위한 클래스의 정보 정의
Attribute	속성의 정보를 정의
Operation	오퍼레이션의 정보를 정의
Parameter	파라미터의 정보를 정의
Relationship Information	관계의 정보를 저장하기 위한 관계의 정보 정의
FileInformation	클래스와 관계의 정보를 파일로 저장하기 위한 파일 정보 정의

ModelInterface는 다이어그램의 정보 요청과 정보를 가공하는 요청을 처리하기 위해서 ClassInformation 컴포넌트의 인터페이스를 정의하였다. 정의된 인터페이스는 Model에서 수행한다. 클래스, 관계, 파일의 정보를 저장하기 위해서 InforamtionStorage이 그 정보를 저장하도록 하였다. ClassInforamtion는 클래스의 정보를 저장한다. Attribute는 속성의 가시성과 이름 그리고 타입을 저장한다. Operation은 오퍼레이션의 가시성과 이름 그리고 리턴타입을 저장하도록 하고, Parameter는 파라미터의 이름과 타입을 저장한다. 관계의 정보를 저장하기 위해서 RelationshipInformation은 관계의 이름과 타입, 시작클래스, 도착클래스를 저장한다. FileInformation은 파일에 클래스와 관계의 정보를 저장한다.

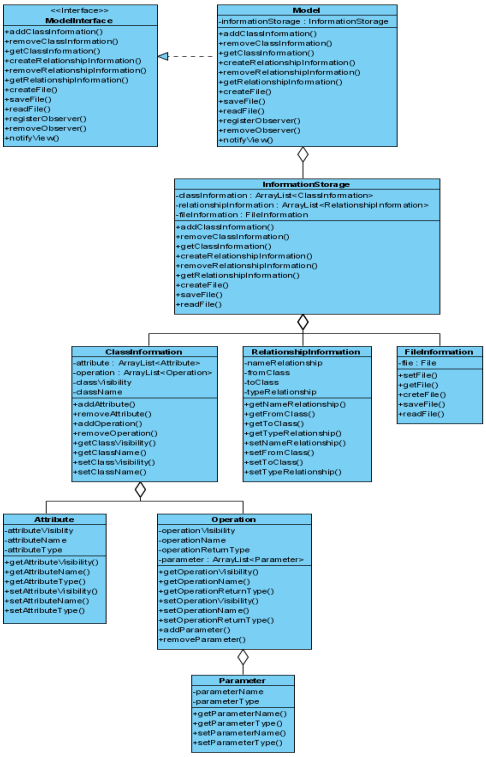

그림 3. ClassInformation 컴포넌트 구조
Figure 3. The Structure Component ClassInformation

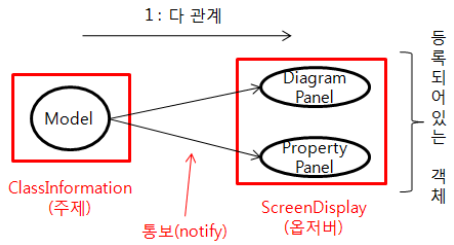

그림 4. ClassInformation과 ScreenDisplay의 관계
Figure 4. A Relationship of the ClassInformation and ScreenDisplay

그림 4는 ClassInformation과 ScreenDisplay의 컴포넌트 관계를 보인다. 다이어그램 정보가 변경되면 저작도구의 화면에서 이 정보를 표현하는 각각의 클래스 상태를 갱신하기 위해서 Observer 패턴[10]을 사용하였다.

이 패턴에서 주제가 되는 Subject 객체는 Observer 객체들을 등록하거나 삭제할 수 있기 때문에 주제가 되는ClassInformation 컴포넌트의 Model 객체에서 옵저버역할을 하는 ScreenDisplay의 객체들을 등록하거나 삭제할 수 있도록 하였다. 그리고 주제 객체의 상태가 변경되면 등록되어 있는 Observer객체에게 통보를 하여, Observer 객체들은 업데이트 내용을 전달 받기 때문에 ClassInformation 컴포넌트의 Model객체에게 ScreenDisplay 컴포넌트의 객체들을 등록하여, 변경된 다이어그램의 정보를 통보받도록 하도록 하였다.

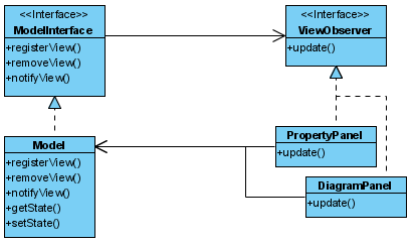

그림 5. 변경된 다이어그램을 화면에 처리하는 구조
Figure 5. The Structure to process The NewDiagram on The Screen

그림 5는 변경된 다이어그램을 화면에 표현하는 절차를 클래스 다이어그램으로 보인다. 옵저버 등록과 삭제 그리고 통보를 하기 위해서 ModelInterface에서 등록, 삭제, 통보하는 인터페이스를 정의하였다. Model객체에게 DiagramPanel, PropertyPanel을 등록한다. 그리고 다이어그램의 정보가 변경되면 화면을 갱신하기위해서 ViewObserver를 정의하였다. Model객체 상태가 변경되는 경우에는 Model객체에 의존하고 있는 PropertyPanel과 DiagramPanel에게 통보를 하고, PropertyPanel과 DiagramPanel이 화면을 갱신한다. 저작도구의 화면구성이 변경되어서 ScreenDisplay 컴포넌트의 클래스를 변경하는 경우에 ViewObserver를 구현한다는 조건만 만족된다면 서로에게 영향을 미치지 않도록 하였다.

4.2 ScreenDisplay

View 컴포넌트는 다이어그램의 정보를 시각적인 요소로 표현하고, 저작도구의 화면을 담당한다. 저작도구의 화면에 관련된 모든 클래스의 요청을 처리하기 위해서 Frame은 ScreenDisplay 컴포넌트의 모든 클래스들을 관리하도록 하였다. 변경된 다이어그램의 정보를 표현하는 화면을 갱신하기위해 ViewObserver는 갱신하는 인터페이스를 정의하였다. DiagramPanel과 PropertyPanel은 정의된 인터페이스를 수행한다.

DiagramPanel은 화면에 다이어그램을 표현하고, PropertyPanel은 클래스, 관계의 정보를 각각의 유형에 맞게 테이블을 통해 나타낸다. ToolBar는 다이어그램을 화면에 작성하기 위해서 클래스와 관계의 메뉴를 포함한다. 다이어그램에서 클래스와 관계를 표현하기 위해 GDrawStorage는 GNode와 GEdge를 저장한다. 클래스를 표현하기위한 인터페이스는 GNode에 정의하였고, GClass가 정의된 인터페이스를 수행한다. 관계를 표현하기위한 인터페이스는 GEdge에 정의하였다. 그리고 GRelationship이 ArrowHead와 LineStyle을 사용하여 관계를 표현하는 인터페이스를 수행한다. 한다. name, attribute, operation을 입력하기 위해서 PropertyDialog는 클래스의 속성 값을 입력하도록 하였다. 저작도구의 ClassInformation 컴포넌트의 요약과 구조는 표 2와 그림 6과 같다.

표 2. ScreenDisplay 컴포넌트 요약
Table 2. The Summary Component ScreenDisplay

클래스 이름	클래스 역할
Frame	저작도구의 화면을 나타내기 위해 Screen Display 컴포넌트의 클래스들을 묶어서 관리
View Observer	각각의 옵저버들(PropertyPanel, DiagramPanel)에게 화면을 갱신하도록 하는 인터페이스 정의
Property Panel	클래스, 관계의 정보를 표현하는 영역
Diagram Panel	다이어그램이 작성되는 영역
ToolBar	클래스, 관계를 작성하기 위한 메뉴
GDraw Storage	GNode와 GEdge를 저장하는 저장소
GNode	클래스를 표현하기위한 인터페이스 제공
GClass	GNode 인터페이스의 실체화이며 클래스를 표현하는 것을 수행
GEdge	관계를 표현하기위한 인터페이스 제공
GRelationship	GEdge 인터페이스의 실체화이며, LineStyle과 ArrowHead을 사용해서 하나의 관계 표현을 수행
LineStyle	Relationship을 표현하는 선 종류(실선, 점선)를 출력(display)
ArrowHead	화살표 머리 모양 종류(삼각형, 다이아몬드, 꺽쇠)를 표현
Property Dialog	클래스의 속성(name, attribute, operation) 값을 입력하도록 하는 다이얼로그

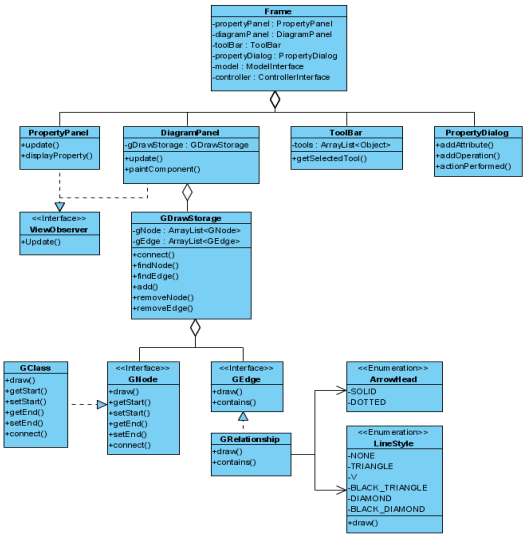

그림 6. ScreenDisplay 컴포넌트 구조
Figure 6. The Structure Component ScreenDisplay

4.3 ToolManager

ToolManager 컴포넌트는 다이어그램의 정보 가공 또는 이벤트가 발생하는 것을 관리한다. 그리고 Screen Display 컴포넌트의 특정 클래스에 해당하는 이벤트를 처리하기 위해서 각각의 컨트롤러를 만들고, 발생한 이벤트의 유형에 따른 컨트롤러에게 요청을 위임하여 처리하도록 하였다. 다이어그램의 유효성을 검증하기 위해서 다이어그램 검증을 처리하도록 하였다. 그리고 필요에 따라 입력된 값을 ClassInformation 컴포넌트의 다이어그램의 정보와 관련이 없는 경우를(그 정보를 가공하지 않고 화면에 표현하는 경우)위해서 ToolManager 컴포넌트에서 ScreenDisplay 컴포넌트에게 저작도구의 화면 갱신을 요청하도록 하였다. ToolManager 컴포넌트의 요약과 구조는 표 3, 그림 7과 같다.

표 3. ToolManager 컴포넌트 요약
Table 3. The Summary Component ToolManager

클래스 이름	클래스 역할
Controller Interface	ToolManager 컴포넌트의 인터페이스로써, ScreenDisplay, ClassInformaion 컴포넌트와 상호작용하는 역할
Manage Controller	ControllerInterface의 실체화이며, 각각의 컨트롤러에게 데이터 입력 또는 발생된 이벤트를 위임하는 역할

클래스 이름	클래스 역할
DiagramPanel Controller	다이어그램을 그리기 위해서 DiagramPanel에서 발생하는 이벤트 수행
PropertyDialog Controller	클래스의 이름, 클래스의 속성, 클래스의 오퍼레이션의 입력된 데이터를 처리하기 위해서 PropertyDialog에서 발생하는 이벤트 수행
File Controller	클래스의 정보와 관계의 정보를 파일로 저장하기 위해서 파일에 관련된 이벤트 수행
GDrawStorage Controller	다이어그램을 그리기 위한 GNode와 GEdge를 저장하고 있는 GDrawStorage에 관련된 이벤트 수행
Inspection Interface	다이어그램의 검사를 수행하는 인터페이스 정의
Diagram Inspection	InspectionInterface의 실체화이며, 다이어그램 검사를 수행

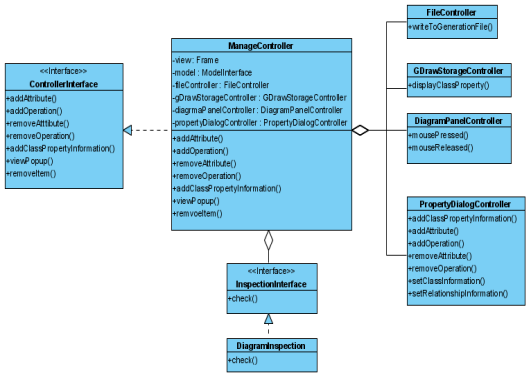

그림 7. ToolManager 컴포넌트 구조
Figure 7. The Structure Component ToolManager

ScreenDisplay 컴포넌트, ClassInformation 컴포넌트와 상호작용 하기위해서 ControllerInterface를 정의 하였다. ControllerInterface에서 정의된 인터페이스를 Manage Controller가 수행한다. DiagramPanelManager는 Diagram Panel에서 발생하는 이벤트를 수행하기 때문에 다이어그램을 표현하는 이벤트를 처리한다. PropertyDialog Manager는 PropertyDialog에서 발생하는 이벤트를 처리하기 때문에 클래스의 속성 값을 입력하는 이벤트를 처리한다. 클래스의 정보와 관계의 정보를 파일로 저장하기 위해서 FileManager는 파일에 관련된 이벤트를 수행하게 하였다.

GDrawStorageManager는 GDrawStorage에서 발생하는 이벤트를 처리한다. 다이어그램의 유효성 검사를 하기 위해서InspectionInterface를 정의하였다. Inspection Interface에서 정의된 인터페이스는 DiagramInspection이 검증한다.

V. 클래스 다이어그램 저작도구의 구현

본 절에서는 3장, 4장의 설계를 바탕으로 구현하였으며, Java 언어를 사용하였다. 그림 8은 저작도구에서 클래스와 관계를 사용하여 모델링하는 예를 보인다. 다이어그램 화면에서 선택한 클래스 또는 관계의 정보를 타입에 맞춰 테이블로 표현하기 위해서 A 영역에 그 정보를 표현하였다. 그리고 클래스 또는 관계를 작성하기 위한 메뉴바를 B 영역에 표현하였다. B 영역의 메뉴바를 통해 클래스 또는 관계를 표현하기 위해서 C 영역에 클래스와 관계를 표현하였다. 클래스의 속성을 입력을 하기 위해서 D 영역에 클래스 이름과 속성 그리고 오퍼레이션을 표현하였다. 그리고 입력된 속성을 C 영역에 표현하였다.

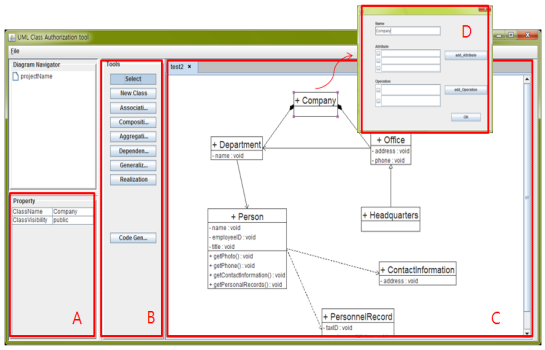

그림 8. 클래스다이어그램 저작도구의 모델링 작성 예
Figure 8. The Example Description Modeling of The Class Diagram Authorization Tool

5.1 클래스 다이어그램의 화면 작성 처리

저작도구의 화면에서 클래스를 작성하는 절차를 그림 9와 같이 보인다.

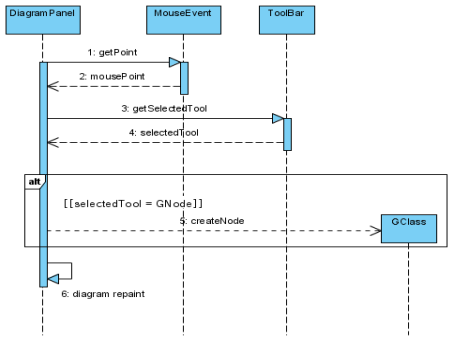

그림 9. 화면에 클래스 작성하기
Figure 9. The Make a Class in the Screen

사용자가 저작도구의 화면에 클래스를 작성하면 DiagramPanel이 MouseEvent에게 다이어그램 위치를 요청하고 반환받는다. 그리고 DiagramPanel은 ToolBar에게 선택된 메뉴를 요청하고 반한 받는다. 반환받은 것이 클래스이면 GClass를 생성한다. 그리고 DiagramPanel은 화면에 클래스를 표시하기 위해서 화면을 갱신하고 화면에 표시한다.

5.2 클래스 다이어그램의 클래스 정보 처리

저작도구의 화면에 작성된 노드(클래스)에 새로운 클래스 정보를 저장하기 위해서 작성된 노드에 저장하는 과정을 그림 10과 같이 보인다.

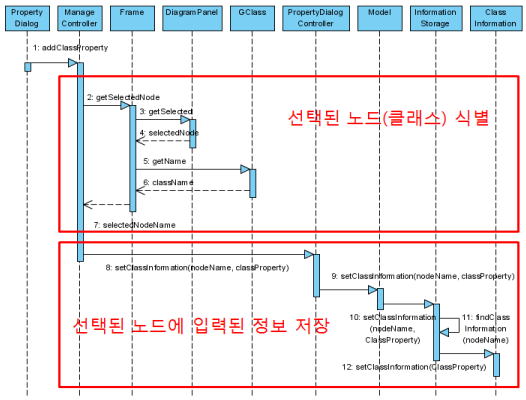

그림 10. 작성된 클래스 정보 저장
Figure 10. The Class Information Save

PropertyDialog에서 새로운 정보를 입력되면 Manage Controller에게 이 정보를 저장하는 요청을 한다. 그리고 작성된 노드에 새로운 정보를 저장하기 위해서 선택된 노드 식별을 먼저 하도록 하였다. ManageContorller는 Frame에게 선택된 노드를 요청한다. Frame은 Screen Dispaly 컴포넌트의 모든 클래스 요청을 관리하기 때문에 요청을 받도록 하였다. Frame은 선택된 노드를 식별하기 위해서 DiagramPanel에게 요청하고 반환 받는다. 클래스정보 저장소에 노드의 이름으로 각각의 클래스 정보를 식별하도록 하였기 때문에 반환받은 Frame은 GClass에게 선택된 노드의 이름을 요청하고 반환받는다. 반환받은 노드의 이름을 Managecontroller에게 다시 반환 한다. Managecontroller는 PropertyDialog의 이벤트를 처리하는 PropertyDialogController에게 위임한다. 다음으로 클래스 다이어그램의 정보를 처리하기 위해서 클래스의 정보를 저장하도록 하였다.

PropertyDialogController는 새로운 정보를 저장하기 위해서 ClassInformation 컴포넌트의 모든 요청을 받는 Model에게 선택된 노드의 이름을 전달한다. Model은 클래스의 정보를 저장하고 있는 InformationStorage에게 노드의 이름을 전달한다. InformationStorage는 전달받은 노드의 이름으로 ClassInformation을 식별한다. 마지막으로 식별된 ClassInformation에게 새로운 클래스의 정보를 저장하고 저장과정을 마치도록 하였다.

그림 11은 클래스정보 저장소에 저장된 새로운 클래스 정보를 불러와서 저작도구의 화면에 다시 표현하는 과정을 보인다.

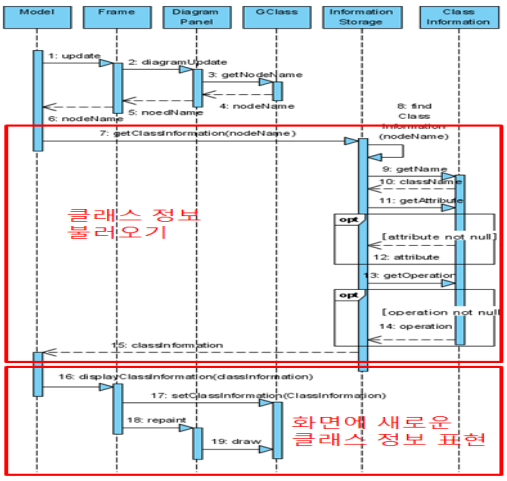

그림 11. 새로운 클래스 정보를 불러오기
Figure 11. The Relationship Information Read

클래스의 정보가 변경되었기 때문에 Model은 Frame에게 화면을 갱신하도록 요청한다. Frame은 다이어그램을 작성하는 DiagramPanel에게 화면 갱신을 요청한다. 그리고 DiagramPanel은 정보가 변경된 노드에 이름을 GClass에게 요청하고 반환받는다. Frame에게 변경된 노드의 이름을 다시 반환하도록 하였다. Frame은 변경된 다이어그램을 식별하기 위해서 Model에게 반환하고, Model은 InformationStroage에게 변경된 노드의 이름으로 ClassInformation을 식별한다. 클래스를 표현할 때 UML spec을 바탕으로 클래스의 이름과 가시성은 반드시 작성되어야 하지만 속성과, 오퍼레이션은 사용자 정의에 따르도록 하였다. 그래서 InformationStroage는 식별된 ClassInformation에게 클래스 이름과 클래스의 가시성을 요청하고 반환받는다. 그리고 속성과 오퍼레이션도 요청하고, 정보가 저장되어있으면 반환받는다. InformationStroage는 Model에게 변경된 클래스의 정보를 다시 반환한다. Model은 반환받은 변경된 클래스의 정보를 화면에 표현하기 위해서 Fame에게 이 정보를 전달한다. 클래스를 표현하는 GClass에게 Frame은 반환받은 정보를 저장하도록 전달하고, DiagramPanel에게 화면 갱신을 요청한다. 요청받은 DiagramPanel은 화면을 갱신하기위해서 GClass에게 변경된 클래스의 정보를 표시하도록 요청을 한다. GClass는 화면을 갱신하고 변경된 클래스의 정보를 불러와서 화면에 표현하는 과정을 마치도록 하였다.

기존에 UML을 이용하여 연구된 저작도구와 본 논문에서 제시한 저작도구를 비교 하였다. 아래의 그림 12와 같다.

	NutCase	Pounamu /Thin	BlueJ	Violet	본 연구
Class Diagram 지원	O	O	O	O	O
Attribute, Operation 지원	O	O	X	O	O
Multi – Attribute, Operation 지원	O	O	X	X	O
Visibility 지원	X	X	X	Attribute, Operation Visibility 지원	Name, Attribute, Operation Visibility 지원
Relationship 지원	6 Relationship (관계 모두 지원)	1 Relationship (Association 지원)	2 Relationship (Dependence, Generalization 지원)	6 Relationship (관계 모두 지원)	6 Relationship (관계 모두 지원)

그림 12. 관련 연구와 본 연구 비교
Figure 12. A Compares Related Research and Our Research

본 연구의 저작도구는 BlueJ에서 지원하지 않는 attribute, operation을 지원하고, Violet에서 지원하지 않는 복수개의 attribute와 operation을 지원한다. 관련 저작도구 (NutCase, Pounamu/Thin, BlueJ, Violet) 에서 지원하지 않는 클래스의 가시성을 지원하도록 하였다.

VI. 결 론

본 논문에서는 MVC 패턴에 기반한 클래스 다이어그램 저작도구의 설계와 구현을 제시하였다. MVC 패턴을 적용한 목적은 클래스 다이어그램 저작도구를 새로운 기능에 대처하는 유연함과 검증된 패턴을 적용하므로 안정성 있도록 하였다. 클래스 다이어그램 저작도구의 ClassInformation, ScreenDisplay, ToolManager를 독립적으로 책임을 부여하고 느슨하게 결합하여 서로 상호작용 하도록 구성하였다. 그리고 다이어그램 정보가 변경되면 정보에 관련된 저작도구의 화면을 구성하는 각각의 클래스 상태를 갱신하기 위해 Observer 패턴을 사용하였다. 안정성을 위한 것은 이미 많은 개발자들의 경험에 의해 검증된 패턴을 적용하였다.

참고문헌

[1] OMG Group, UML Specification, www.omg.org.
[2] Grdy Buooch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide 2nd ED, Addison-Wesley, 2005
[3] Martin Fowler, UML Distilled, 3rd Ed, Addison Wesley, 2004
[4] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns - Elements of Reusable Object-Oriented Software., AddisonWesley, 1994.
[5] Cay Horstmann, Object-Oriented Design & Patterns 2nd ED, Wiley, 2005
[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] 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
[8] Sun Microsystems, BlueJ, www.bluej.org
[9] Horstmann, Violet, www.horstmann.com
[10] Kathy Sierra, Head First Design Patterns, O'RRILLY, 2004

저자소개

김재훈(Jae-Hoon Kim)

2009.2 안동대학교 컴퓨터공학과 공학사
2009.3~현재 안동대학교 컴퓨터공학과 석사과정

※관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 리팩토링 등

김윤호(Yun-Ho Kim)

1983 경북대학교 전자공학과 공학사
1993 경북대학교 대학원 컴퓨터공학과 공학석사
1997 경북대학교 대학원 컴퓨터 공학과 공학박사
1997~현재 안동대학교 전자정보산업학부 교수

※관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 병렬처리 등