휴먼 ICT 소프트웨어공학

Term Project

제 출 일	2015.6.5
학 부	컴퓨터공학부
담당교수	이찬근 교수님
팀	5

박찬주 (20112205)

송민석 (20111084)

신재훈 (20113334)

안재혁 (20111466)

안준형 (20115725)

1. 프로그램 묘사

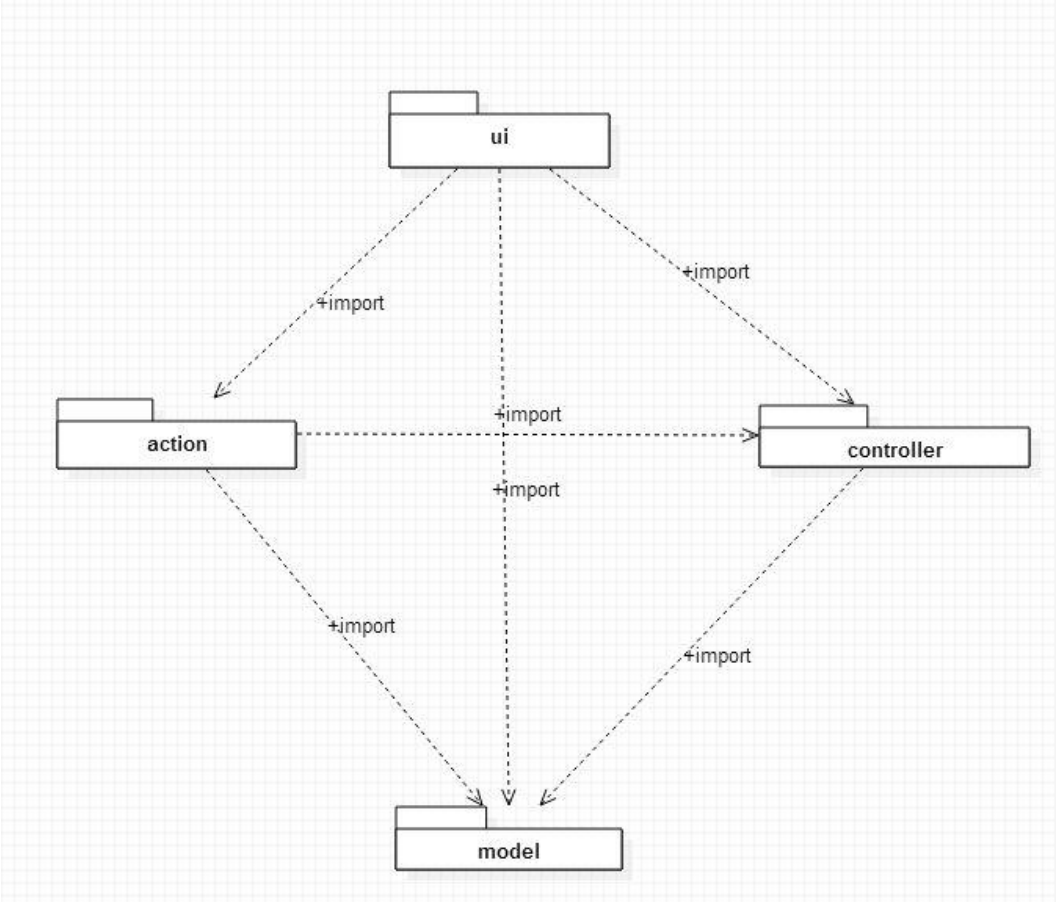
이 프로그램은 어떤 소프트웨어의 설계를 개선 혹은 수정을 위하여 소프트웨어의 각 모듈들의 dependency 를 보여주고 수정 할 수 있는 기능을 가진 프로그램입니다. 타이탄은 DSM 파일을 읽어와 보여주는 tool 입니다. DSM 파일을 open 후 사용자는 클러스터 파일을 load 해서 DSM 을 미리 정의된 순서에 따라 나타낼 수 있습니다. 프로그램의 기본적인 기능은 'Redraw', 'Rename', 'Sort', 'Group', 'Ungroup', 'Up', Down' 입니다. 여기에 추가적으로 'duplicate' 와 'folk'를 구현하였습니다. 두 기능은 선택된 그룹의 편집을 위해 새로운 titan 창을 띄우는데, Duplicate 는 선택된 그룹을 그대로 복사해 원래 그룹에 영향을 끼치지 않는 기능입니다.

Titan 프로그램 구현을 위해 MVC 아키텍처 모델을 적용하여, 모델과 view 를 분리했습니다.

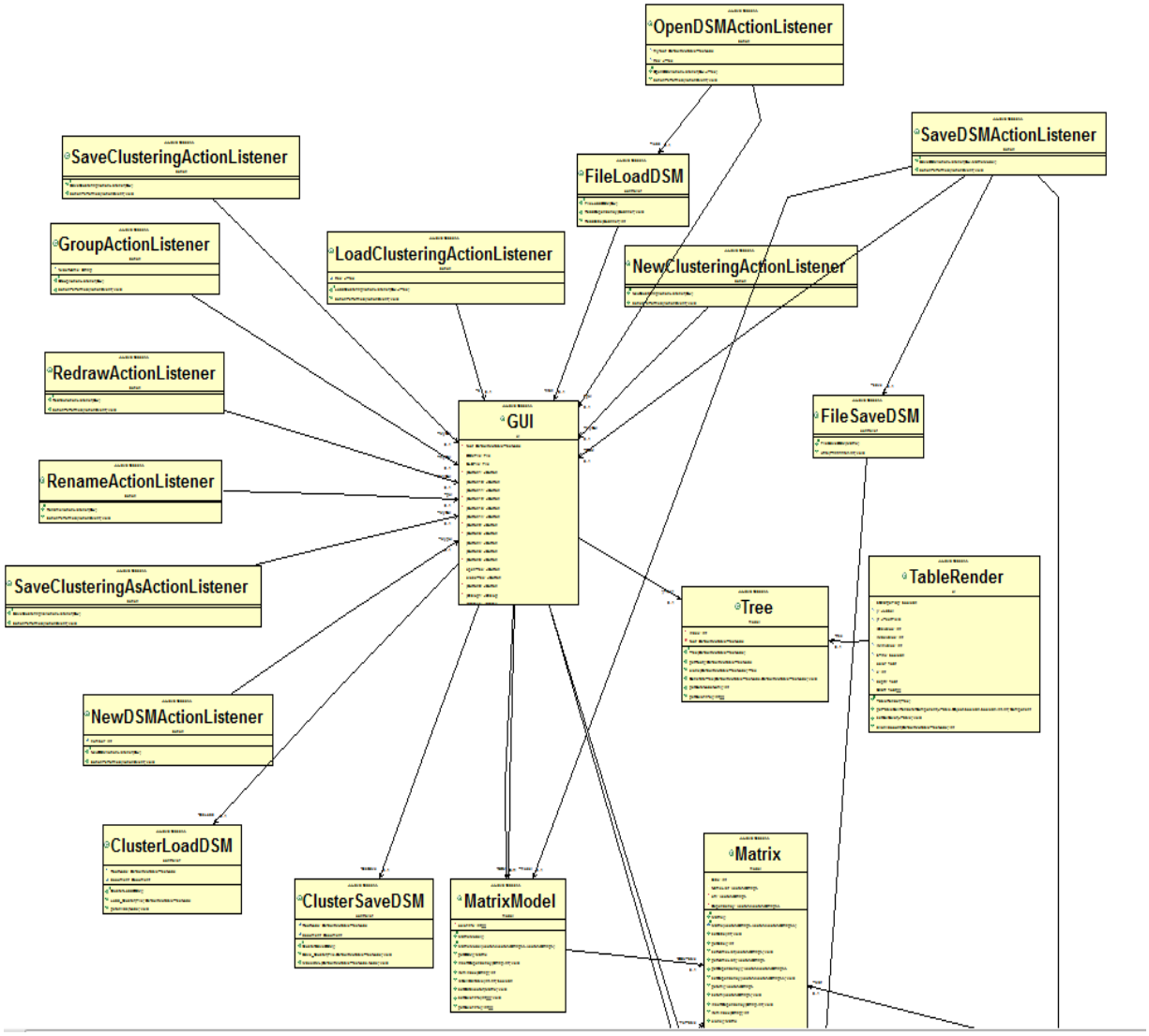
OOP 디자인을 적용해, 다른 기능을 가진 데이터와 그와 관련된 클래스들을 분산시켰습니다.

2. 전체적 디자인

1) Package Diagram



2) Class Diagram



3. MVC 아키텍쳐 패턴 적용 여부

MVC (Model-View-Controller) 패턴을 명시화하기 위해 프로젝트의 패키지를 나누었습니다. Model Package(Model), UI Package(View), 그리고 Controller Package(Controller)로 총 세 개를 생성했습니다. Model 은 상태 변화가 있을 때 컨트롤러와 UI 에 이를 통보하는 역할로, 우리 프로그램에는 각 모듈들의 이름을 표시해주는 table 을 구현하기 위해 DefaultTableModel 을 상속받는 MatrixModel class 를 정의했습니다. MatrixModel class 는

Matrix class 를 변수로 가지고 있어 사용자가 편집한 tree 구조에 맞게 Dependency Matrix 를 유연하게 편집할 수 있도록 해줍니다.

View(UI) Package 에서는 사용자의 결과물을 보여주기 위해 모델로부터 정보를 받아옵니다. GUI 클래스가 해당됩니다.

마지막으로 Controller Package 는 모델에 명령을 내리고 모델의 상태를 변화시킵니다. 또한 Controller 에 명령을 보내 모델의 표시 방법을 바꿀 수 있습니다. 본 프로그램의 action_controller 패키지에 해당되는 ClusterLoadDSM, FileSaveDSM, ClusterUtilities 클래스등이 이에 해당됩니다.

4. 주요 디자인 결정에서의 장점과 단점

장점 : MVC pattern 을 이용하기 위해 패키지를 나누고 클래스의 성격을 구분 지어 구조 파악이 용이합니다. 클래스 별로 unit 테스트가 편리합니다. 모듈 별로 각각 수정이 가능하여 프로그램 maintenance 가 향상되었습니다.

View 와 Model 간의 간섭을 피하고 Controller 가 중간관리를 하는 역할을 하여 좀 더 유연한 구조를 가지게 되었습니다.

단점 : GUI 클래스에서 ui 의 모든 method 와 attribute 가 포함하는데, 그러다 보니 해당 클래스의 크기가 너무 커져버렸습니다. 프로젝트를 관리하는 데 있어서 각 팀원이 작업한 코드를 합치는 데 어려움이 있었습니다. 그 때문에 모듈들간의 Coupling 이 높아지는 단점이 생겼습니다. MVC pattern 에서 Model 과 view 의 의존성을 완벽히 분리할 수 없기 때문에 패턴이 모호해질 수 있고 변형이 생길 가능성이 있습니다.

Naming 과 comment 를 적절하게 표시하지 않아서 팀원들간의 프로젝트 수행에 효율이 안 좋았습니다.

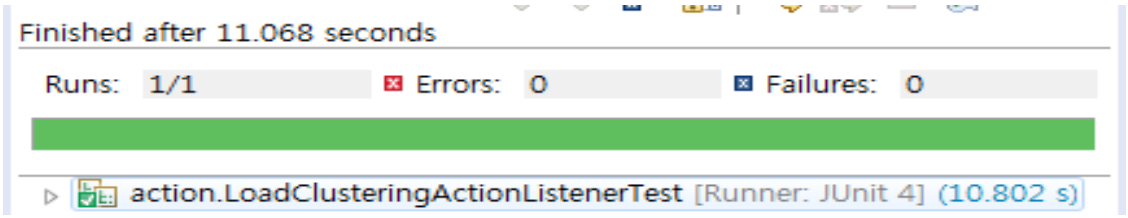
5. 프로그램 테스트

저희는 이클립스에서 지원하는 junit 으로 테스트를 시행하였습니다. 각 클래스나 메소드에 맞는 input 을 주고, 그 결과가 어떻게 나오는지에 대해 저희가 예측한 output 과 클래스나 메소드가 주는 output 을 assert~라는 함수를 이용하여 비교 하였습니다.

예)

```
package action;

import static org.junit.Assert.*;

public class LoadClusteringActionListenerTest {

    GUI gui = new GUI();
    JTree tree = new JTree();

    @Test
    public void testActionPerformed() {

new OpenDSMActionListener(gui,tree).actionPerformed(null); //titan.dsm

        new LoadClusteringActionListener(gui,tree).actionPerformed(null); //titan_DRH+Bunch.clsx

        System.out.println(gui.getRoot().getChildAt(0).toString());

        assertEquals("L0",gui.getRoot().getChildAt(0).toString());

        assertEquals("edu.drexel.cs.rise.titan.util.IconFactory", gui.getRoot().getChildAt(0).getChildAt(0).toString());

        assertEquals("edu.drexel.cs.rise.titan.util.IconFactory", gui.getRoot().getChildAt(0).getChildAt(0).getChildAt(0).toString());

        assertEquals("L1",gui.getRoot().getChildAt(1).toString());

    }

}
```

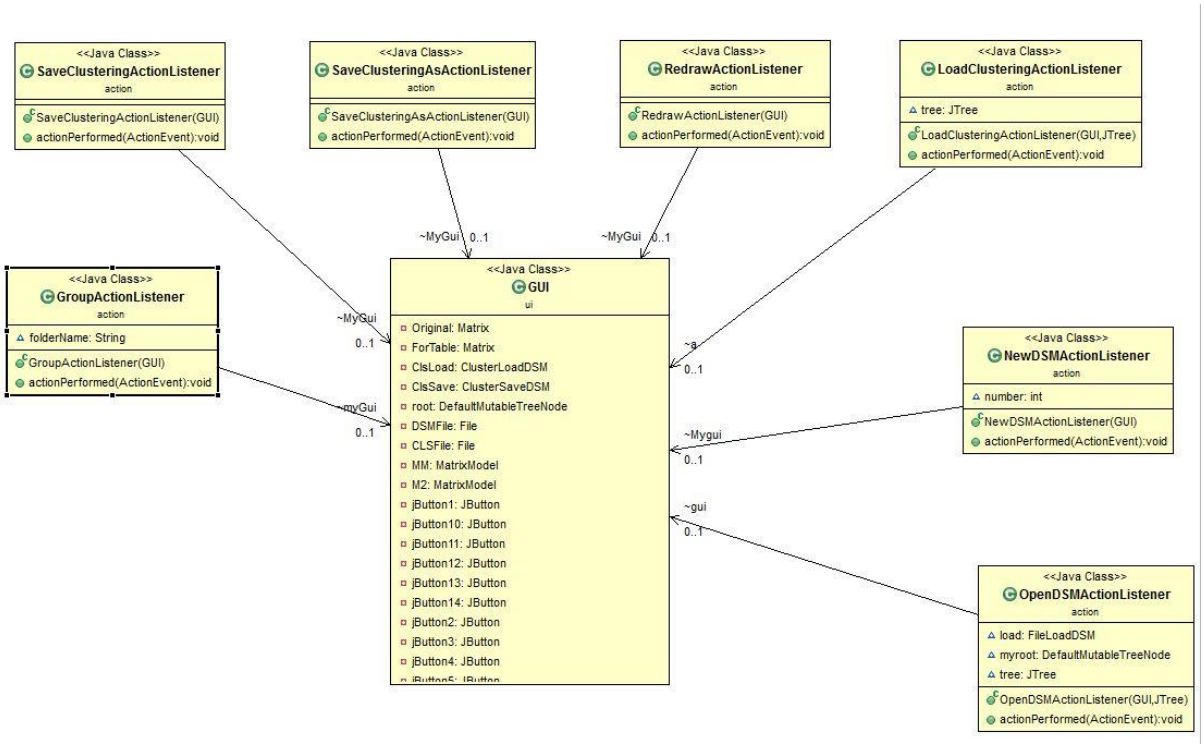
->위 코드에서 테스트 하려는 클래스는 'LoadClusteringActionListener'입니다. 이 클래스의 테스트에서의 시나리오는 'titan.dsm'파일을 열고 'titan_DRH+Brunch.clsx' 파일을 열었을 때, 'LoadClusteringActionListener'는 열었던 클러스터 파일에 맞게 matrix 와 tree 등을 바꿀 것입니다. 그 결과 중 몇 가지를 assert 함수를 통하여 저희가 예상한 값과 이 클래스의 결과값을 비교하였습니다.



->Test 결과 저희가 예상한 값과 클래스의 결과값이 일치 하였으므로 녹색 줄이 뜨게 됩니다.

6. OOP 디자인 적용

a) The single-responsibility principle(SRP)



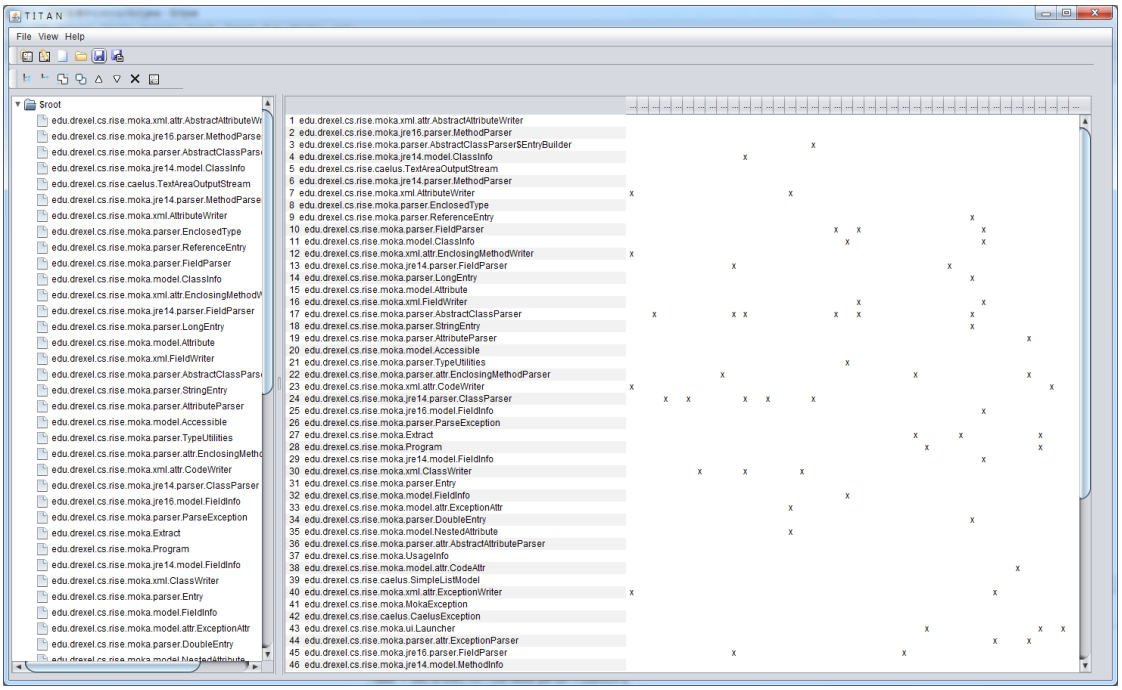
SRP 는 하나의 클래스 혹은 메소드는 하나의 일에 대해서만 책임을 질 수 있도록, 잘 분리되어야 한다는 소프트웨어 설계 원칙을 의미합니다. 저희가 구현한 프로그램에서 예를 들면, ActionListener 클래스는 오직 GUI 클래스에 대해서만 dependency 를 가지고 있습니다. 즉 위 클래스 다이어그램에 나오는 각 클래스들은 오직 GUI 클래스에 대한 ActionListener 에 대한 책임만을 가지고 있어 클래스의 Cohesion 을 높이는 구조를 가집니다.

b) MatrixModel 클래스가 DefaultTableModel 를, Tree 클래스가 jTree 클래스를 상속해서 클래스 재사용을 했습니다

c) Matrix 와 Tree 클래스가 Cloneable interface 를 implement 해서 clone 메소드를 구현하였습니다. 클래스 스스로가 복제를 허용합니다.

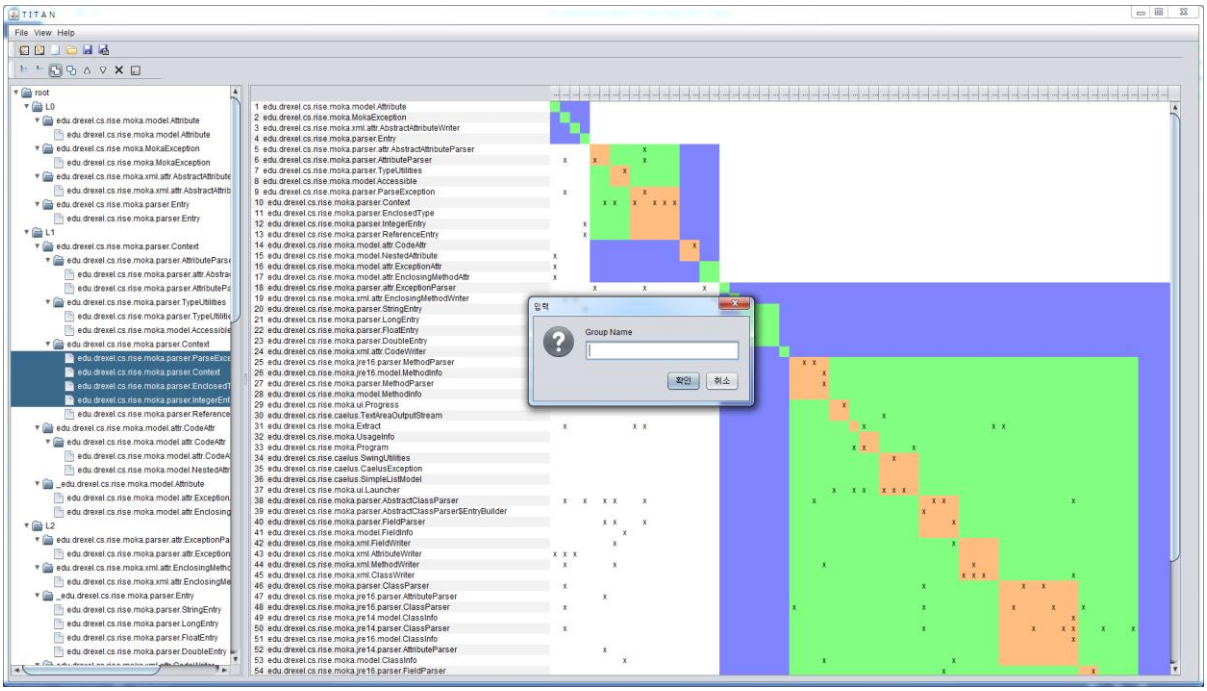
7. 스크린샷

- moka DSM 파일 열었을 때,

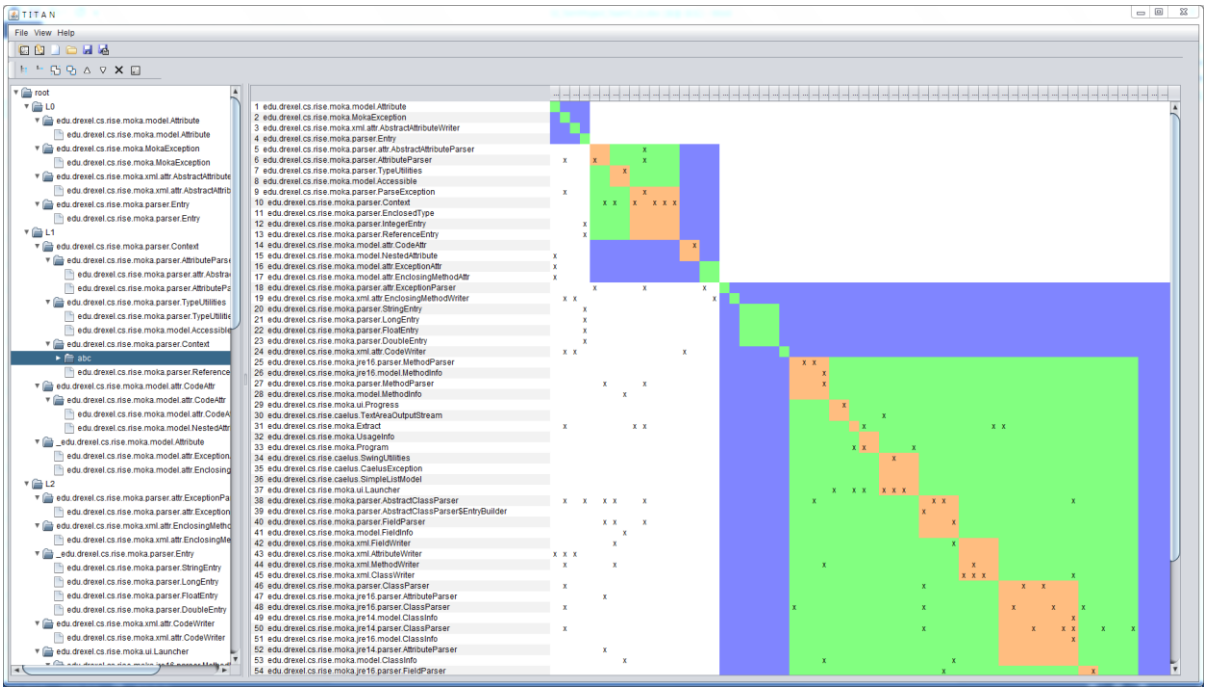


- moka_DRH+ACDC 파일을 열고 Redraw 를 한 화면

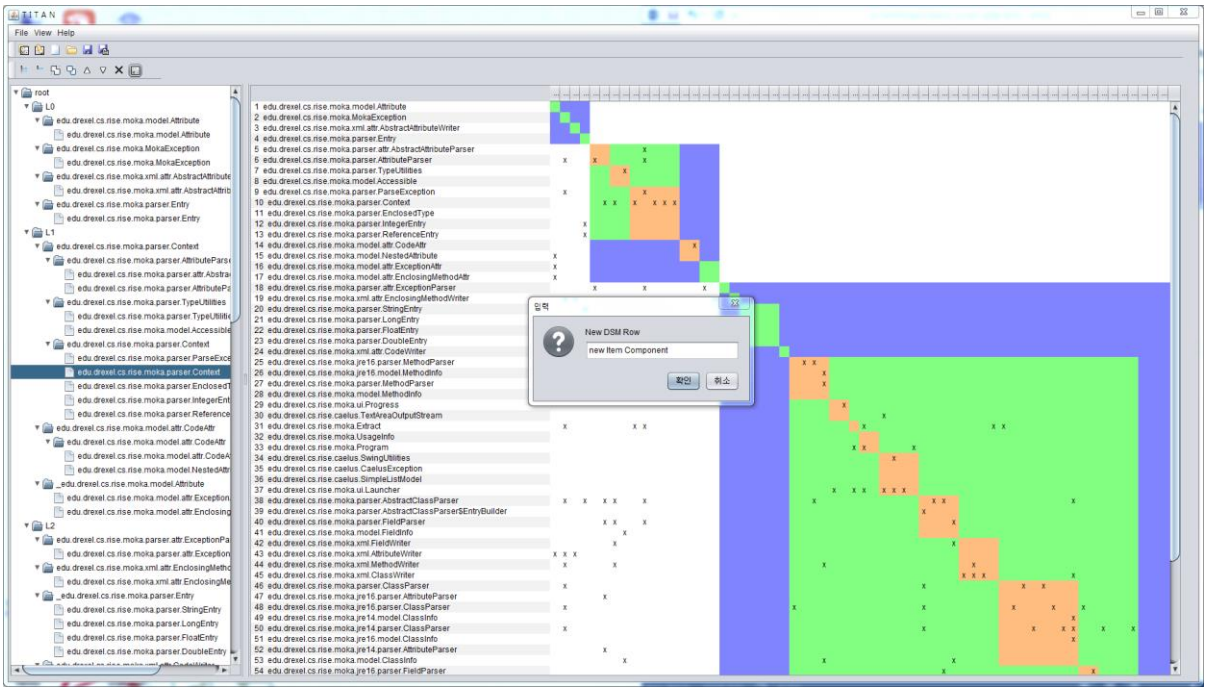
- Item 들을 Group 지는 화면



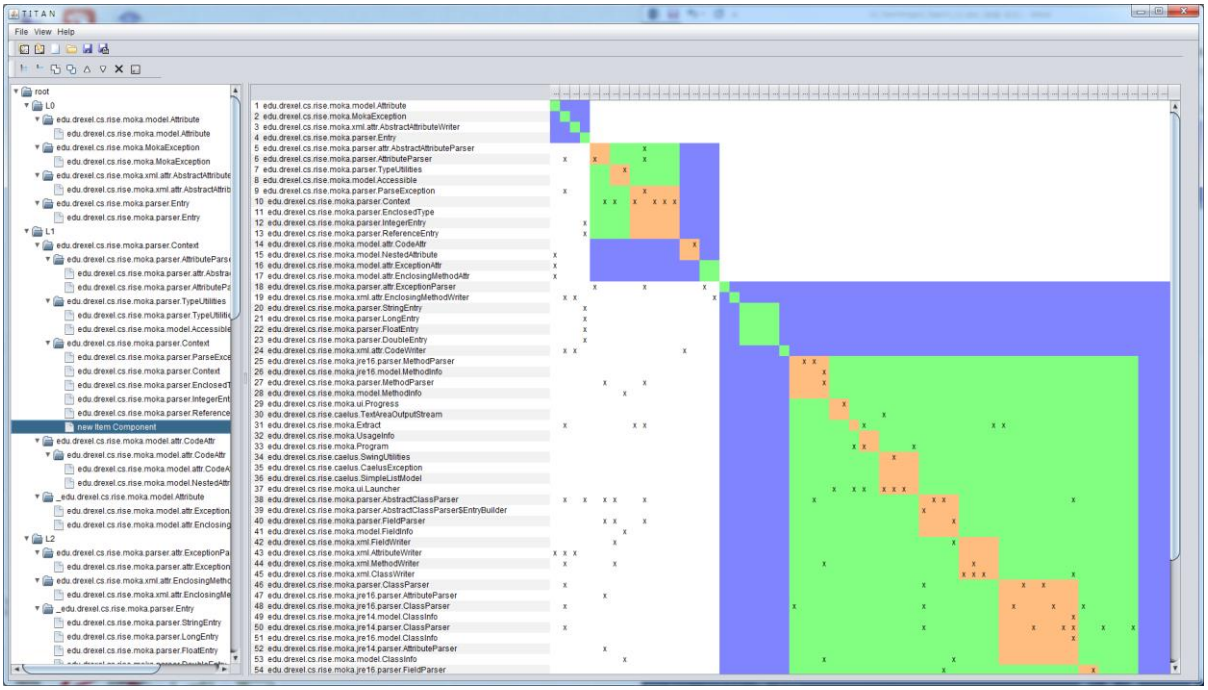
- ‘abc’로 그룹이 생겼습니다.



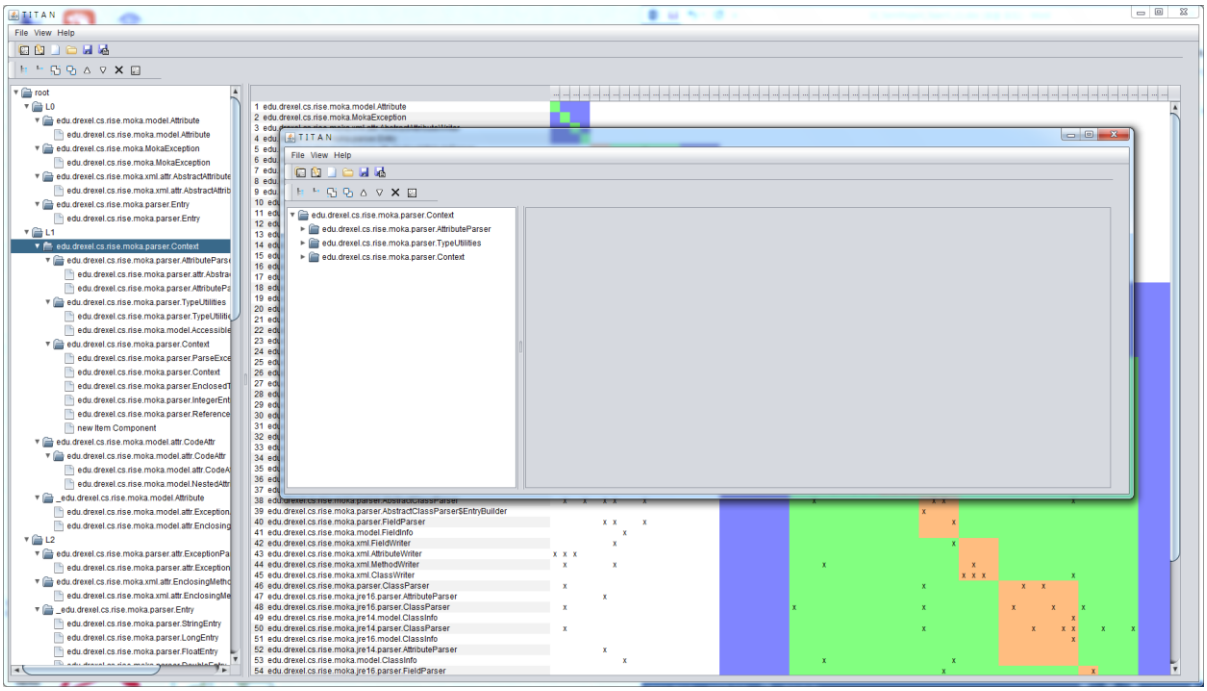
- new component를 생성하는 화면



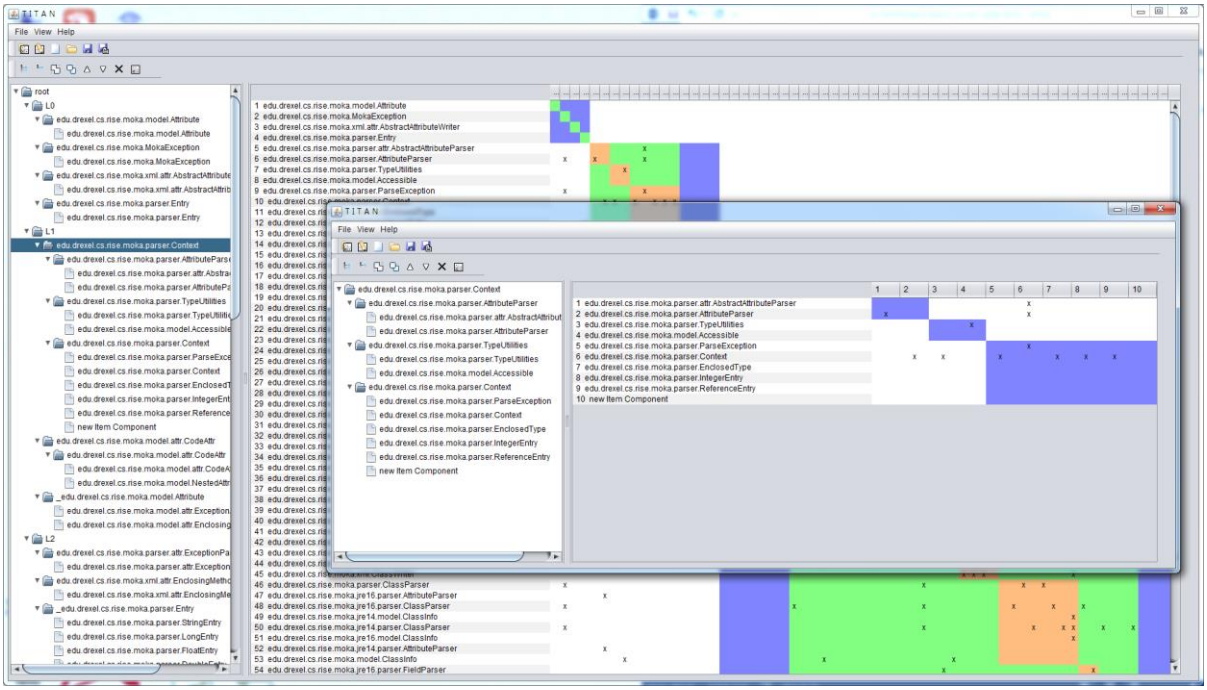
- 생성된 결과



- Duplicate 실행화면



-Duplicate draw 부분



8. 구현하지 못한 부분

- Folk, 부분 저희가 만든 matrix Class 와 tree Class 를 직접 공유 할 수 있게 만드는 법을 못 찾아서, 간접적으로 참조할 수 있는 방법을 찾아보았습니다. 그러나 그 방법에는 시간이 많이 들고, 코드 수정이 많이 필요한 부분이라서 구현하지 못했습니다.

9. 팀원들의 역할

a) 20112205 박찬주

•Tree, Coloring implementation (Redraw table, Tree model design)

•Program implementation, test

b) 20111084 송민석

• Program implementation (Tree, Cluster, File IO)

• Algorithm design

c) 20113334 신재훈

• Program implementation (Overall system)

• Program structure design

• Analysis & Test Titan

d) 20111466 안재혁

• Program implementation (Sorting Algorithm)

• Documentation

• Test program

e) 20115725 안준형

• GUI Design

• Making PPT

• Program Implementation