

Contents



- ✓ 팀원 별 역할
- ✓ 프로그램 정보
- ✓ 개발 프로세스
- ✓ 디자인 다이어그램
- ✓ MVC 아키텍처 패턴의 적용
- ✓ 테스트 과정 및 방법
- ✓ 객체지향적 개발방법의 적용
- ✓ 프로그램 예제
- ✓ 프로젝트를 통해 느낀 점

1) 팀원 별 역할

팀원 및 역할	신우근	Model
	이창수	코드개발 및 문서작성
	경세준	Controller
	배수열	각 Diagram 및 프로그램 스토리 작성
	황정근	View GUI 설계

2) 프로그램 정보

이번 프로젝트를 통해 개발한 프로그램은 DSM 입니다. 프로젝트의 목적 자체가 새로운 프로그램의 개발이 아닌 기존에 있던 프로그램인 Titan을 본 떠 만드는 것이었기 때문에 기본적인 기능은 Titan과 동일하다고 할 수 있습니다. 프로그램의 기본 요구조건인 dsm과 clsx파일의 입출력 및 이에 대한 matrix로의 표현, 프로그램 상에서의 dsm및 clsx파일의 수정 및 저장 등이 이에 해당됩니다.

프로그램을 기능적인 측면이 아닌 구조적인 측면에서 본다면 다음과 같은 특징을 가집니다. 기본적으로 입력되는 모듈들을 제어하는 자료구조는 자식 노드 개수에 제한이 없는 트리 입니다. 트리를 통해 상, 하위 폴더 및 하위 모듈들의 상관관계를 처리하며 모듈의 추가, 삭제, 수정 등 일괄작업이 이루어집니다.

다음으로 프로젝트의 기본 요구조건이기도 한 MVC architectural pattern의 적용으로 Model, View 그리고 Controller를 분리시켰고 이를 통해 개발상의 수정 시의 장점, 프로그램 유지보수 상의 유리함 등을 이끌어 낼 수 있었습니다.

또한 컴포넌트 사이의 데이터 공유 시 Singleton pattern 방법을 사용하여 instance를 한번만 생성해도 모든 컴포넌트의 데이터사용이 가능하도록 구현했습니다. 이를 통해 시간적인 오버헤드와 메모리 낭비차원의 개선을 만들어 낼 수 있었습니다.

2) 개발 프로세스

저희 조가 이번 프로젝트에 적용하기 위해 선택한 개발 프로세스는 Waterfall Process입니다. 이 프로세스를 선택한 자세한 이유는 다음과 같습니다.

먼저, 저희는 이번 프로젝트를 프로그램 개발 자체가 아닌 지금까지 배운 지식들을 실제로 적용해 보기 좋은 기회라고 생각했습니다. 이러한 소규모 프로젝트에는 Implemental Development가 적합하다는 것을 알고 있었지만 저희는 지금까지의 프로그래밍 프로젝트처럼 세세한 계획이나 자세한 명세 없이 개발하려는 프로그램의 주제와 대략적인 기능적 명세만으로 프로그램 개발을 시작하는 것은 이번 학기 수업에서 배운 것들을 제대로 사용하지 못할 것 같다고 판단했습니다. 이에 따라 실제 대형 프로젝트에서의 절차대로 Use-Case Diagram이나 Sequence Diagram, Class Diagram같은 자세하고 확실한 명세를 바탕으로 한 개발 프로세스를 선택하게 되었습니다.

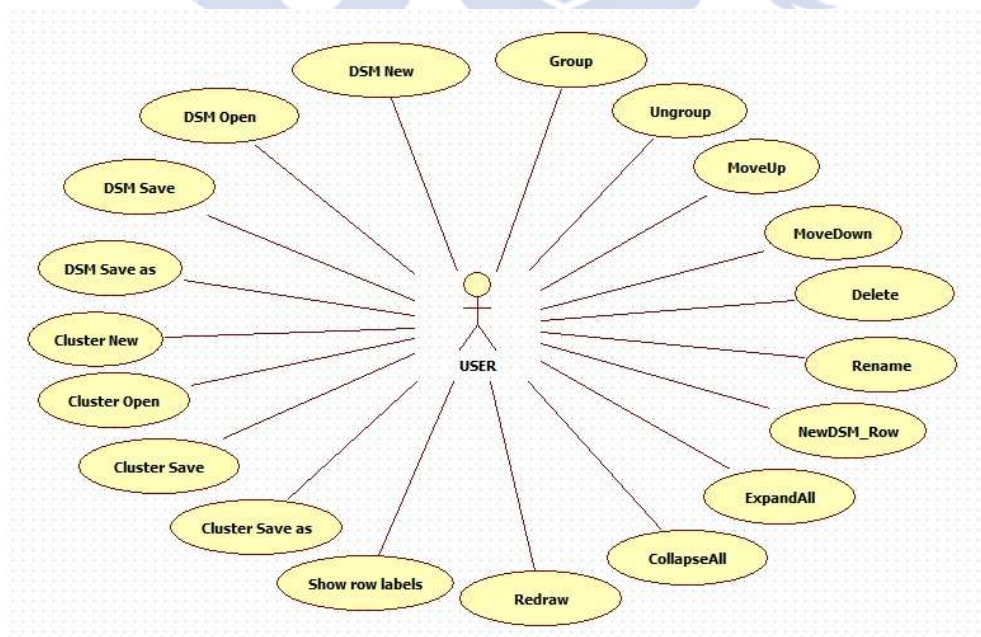
다음으로, Waterfall Process만이 갖는 장점을 직접 체험해보고 싶었습니다. 위에 설명했듯이 소프트웨어공학에서 배우는 지식들, 예를 들면 Waterfall Process나 Implemental Development가 갖는 각각의 장점 및 단점 등을 배우기 전에는 자세한 명세 작성 후 설계보다는 구현이 더 우선시되는 개발 방법을 택했던 것이 보통입니다. 이에 따라 저희는 지금까지의 방식이 아닌 Waterfall Process를 통해 이 방식의 장점을 느껴보려고 했습니다.

물론, Waterfall Process를 처음 사용하면서 겪은 시행착오도 많습니다. Diagram 작성에 대한 숙련도가 부족한 탓에 처음 작성한 Diagram대로 개발을 해나가는

것이 어려웠고 도중에 수정이 필요한 곳이 많았습니다. 하지만 이를 통해 Diagram 자체의 설계뿐 아니라 프로그램 전체적인 설계에 대한 충분한 연습이 되었고 경험이 되었다고 생각합니다.

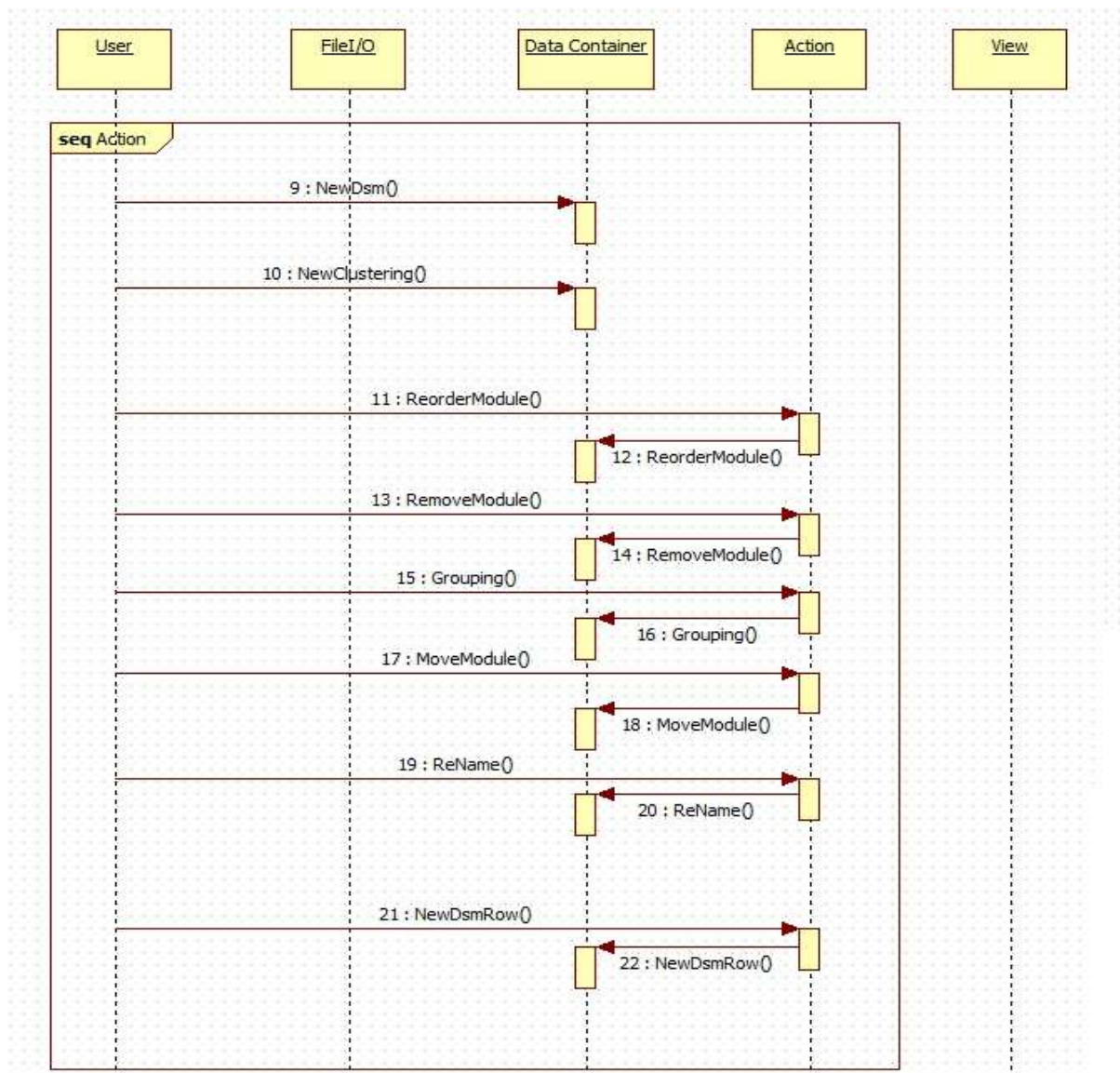
3) 디자인 다이어그램

✓ Use-Case Diagram

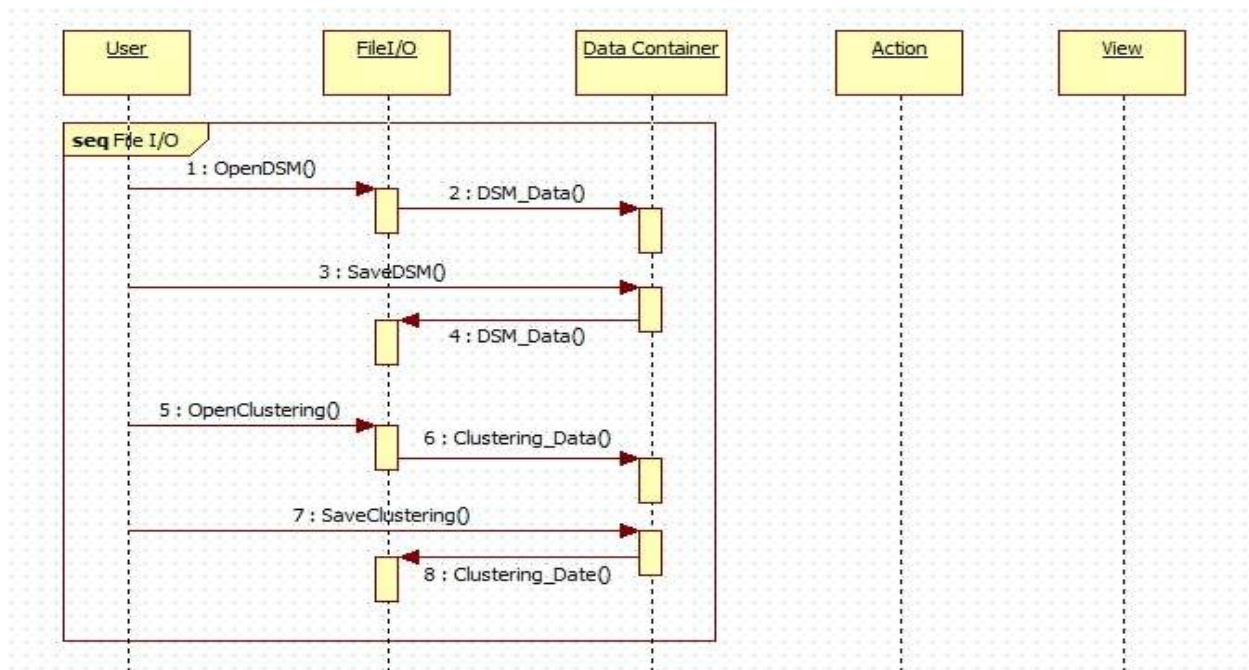


✓ Sequence Diagram

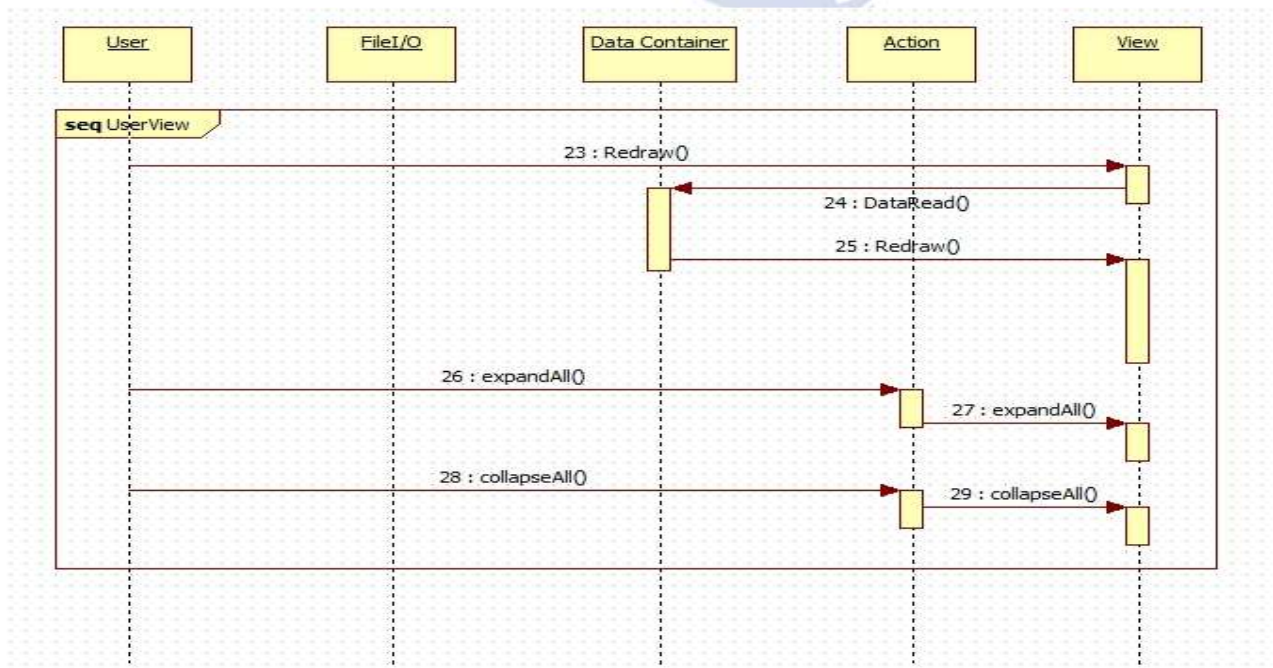
① Action



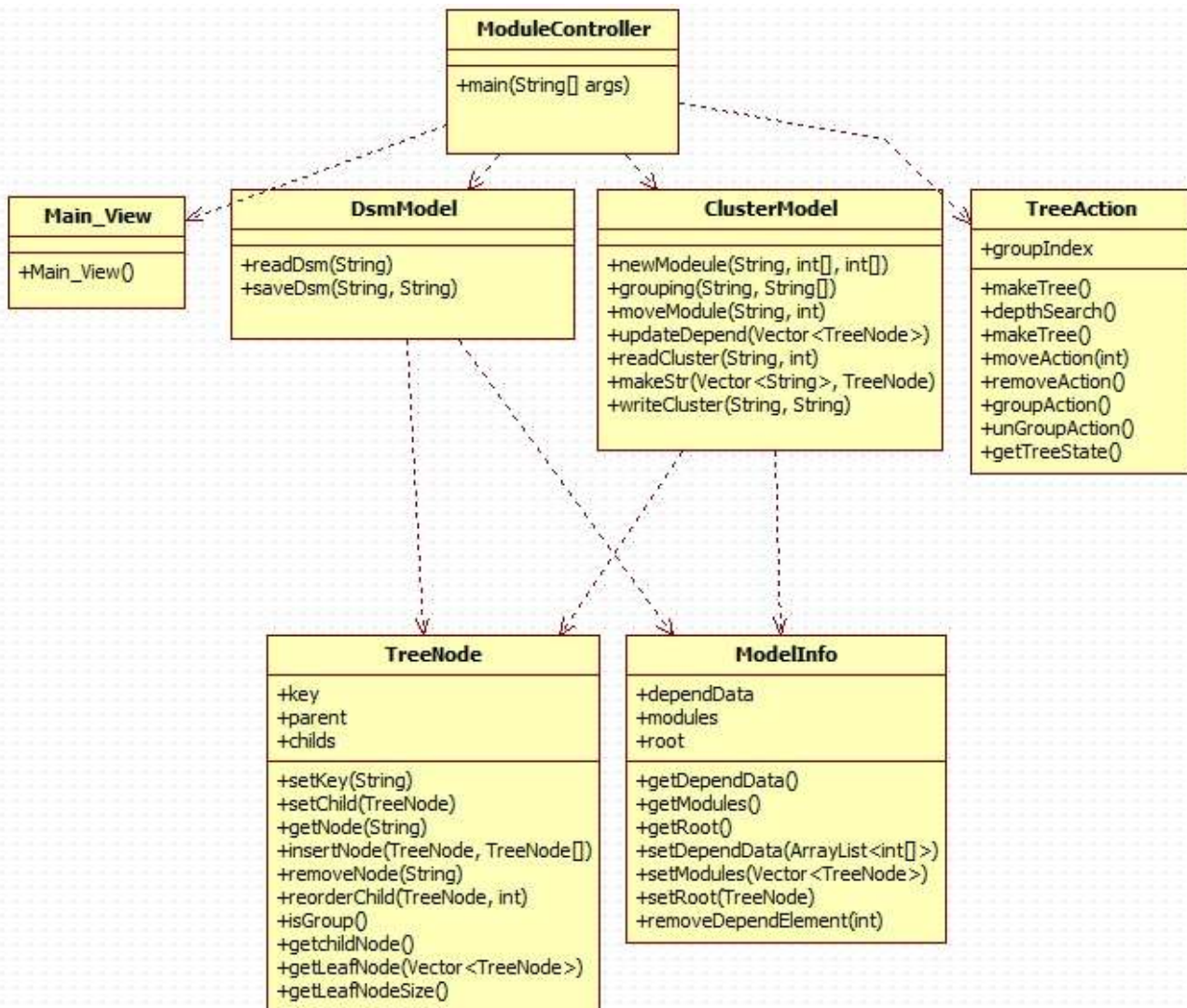
② File I/O



③ User View



✓ Class Diagram



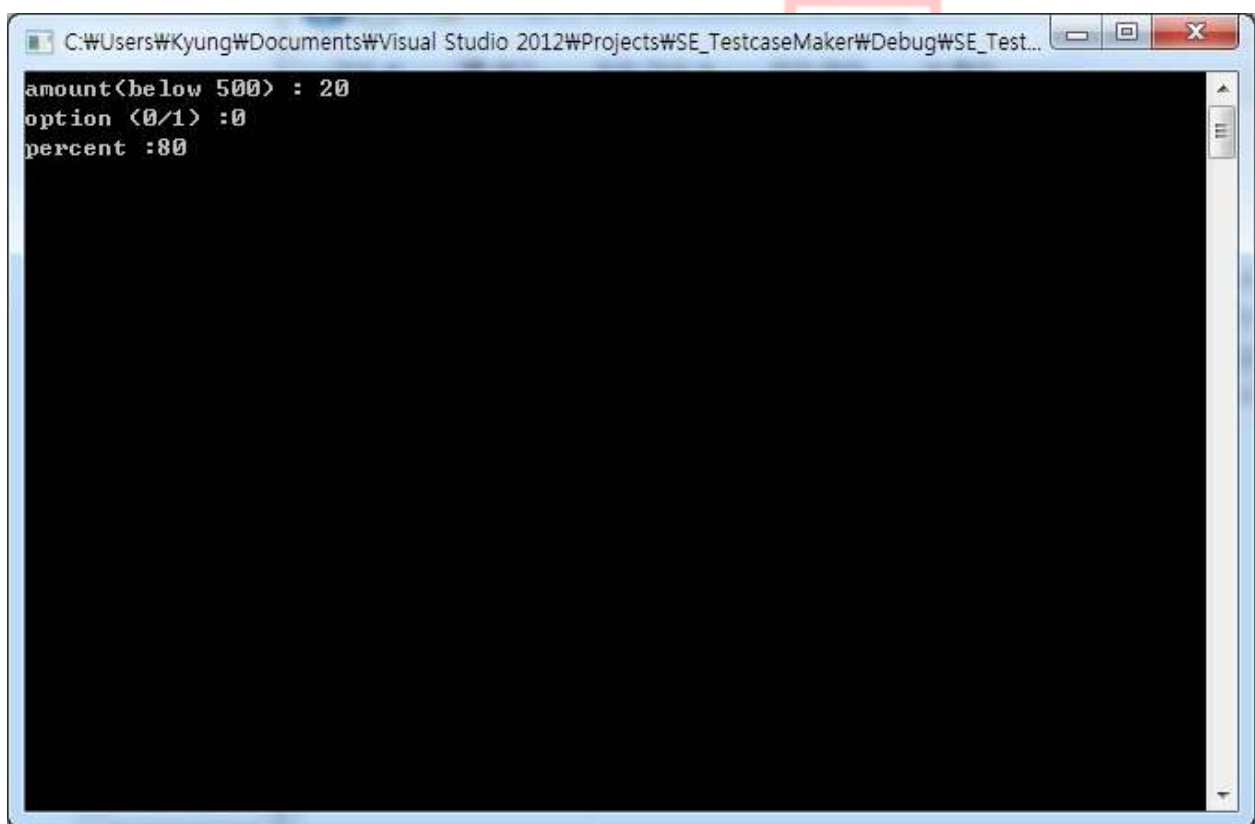
4) MVC 아키텍처 패턴의 적용

저희 조에서는 MVC architectural pattern의 적용의 완성도와 실제 구현 과정에서의 자연스러운 패턴 적용을 위해 개발 시 역할 분담 자체를 Model, View, Controller의 독립적 개발을 할 수 있도록 나누었고 각각의 통합을 통해 프로그램을 완성하는 방향으로 프로젝트를 진행했습니다. 그 결과 저희가 의도했던 대로 개발 과정에서의 MVC pattern의 자연스러운 적용이 가능했습니다. 물론 이 방법으로 제한되는 점도 생기긴 했습니다. Model, View, Controller 개발 간에 서로의 완벽한 이해가 떨어지게 되어 integration시에 어려움이 생기기도 했습니다.

다음으로, 개발 도중 MVC architectural pattern의 장점을 직접 느낄 수 있는 기회가 있었습니다. 개발 도중 Model과 Controller쪽의 완성의 거의 되가는 상황에서 View 컴포넌트 쪽에서 기능적 문제 때문에 Class Diagram과는 달리 구현해야 하는 상황이 있었습니다. View 컴포넌트를 수정하면서 Model, Controller, View를 분리해놓은 것으로 인해 View에서의 수정이 나머지 두 컴포넌트에 영향이 거의 없다는 것을 실제로 알 수 있었습니다. 이를 통해 순수한 아키텍처 설계는 프로젝트의 개발, 그 이후의 유지 보수에 긍정적인 영향을 크게 미친다는 것을 다시 한번 느낄 수 있었습니다.

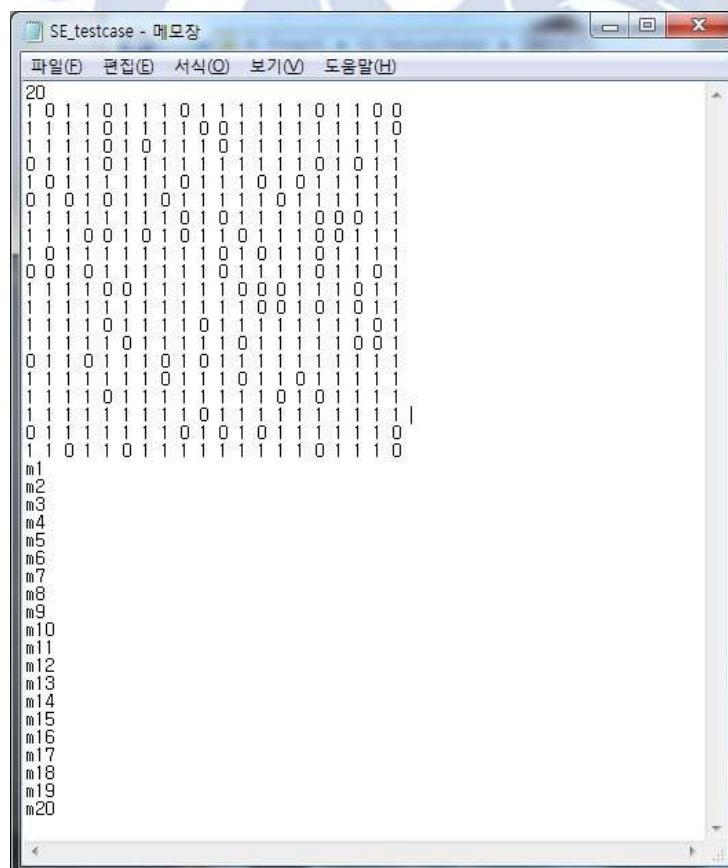
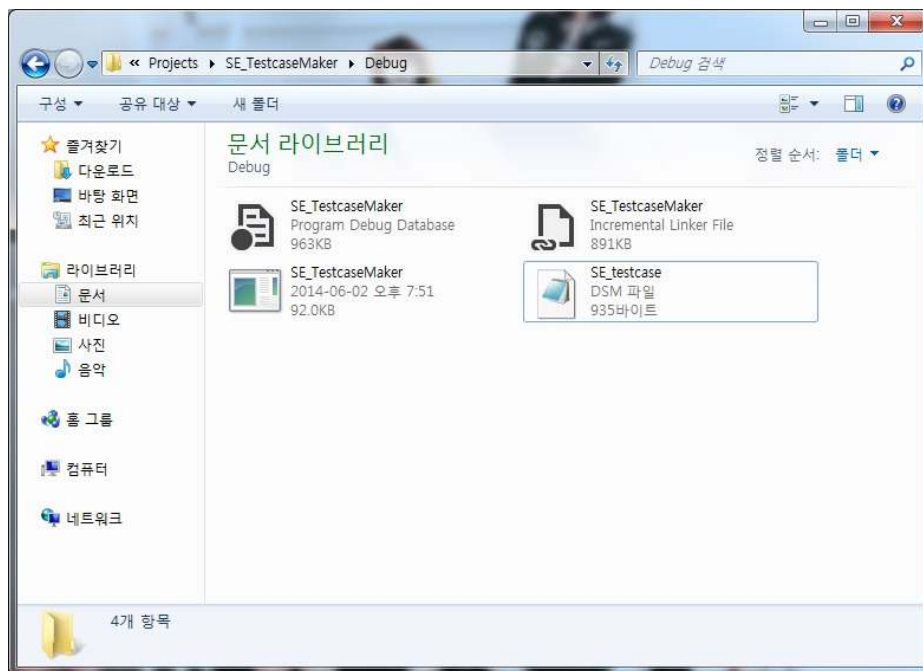
5) 테스트과정 및 방법

저희 조에서는 테스트 케이스들을 만들기 위해 필요한 조건에 따라서 dsm파일을 만드는 간단한 프로그램을 만들었습니다. 프로그램은 다음과 같습니다.

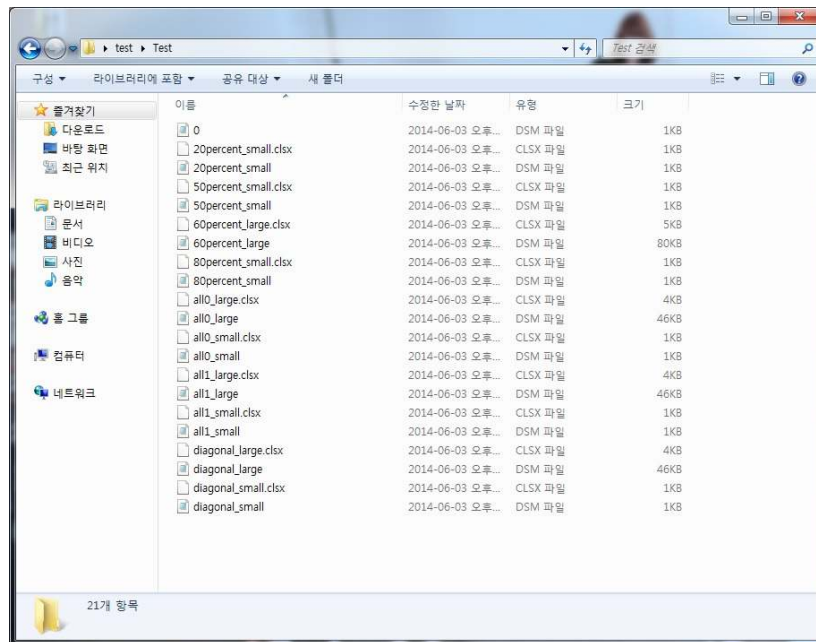


```
C:\Users\Kyung\Documents\Visual Studio 2012\Projects\SE_TestcaseMaker\Debug\SE_Test...
amount(below 500) : 20
option (0/1) : 0
percent : 80
```

맨 위의 항목은 모듈의 개수를 의미하며 두번째 항목은 diagonal 형태로 dsm을 만들지 결정합니다. 그리고 마지막 항목은 1과 0의 비율을 조절하는 항목이며 숫자가 높을수록 1의 비중이 많아집니다. 프로그램 실행 결과는 다음과 같습니다.



이 프로그램을 이용해 테스트 케이스들을 만드는 기준인 Equivalence Partition을 준수하여 다음과 같은 여러가지 테스트 케이스를 만들었습니다.



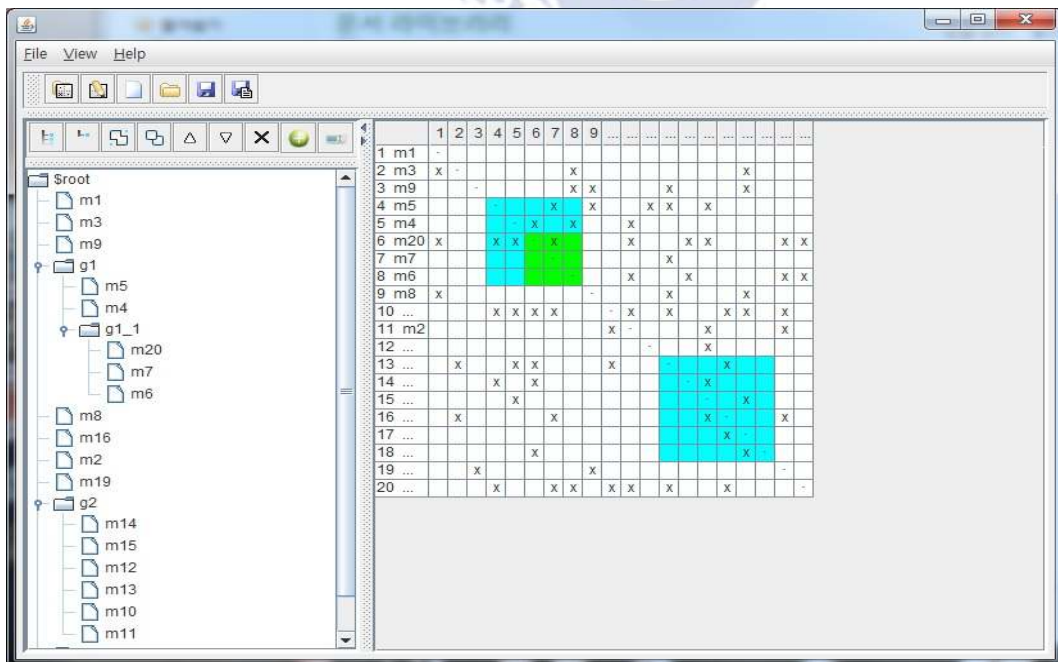
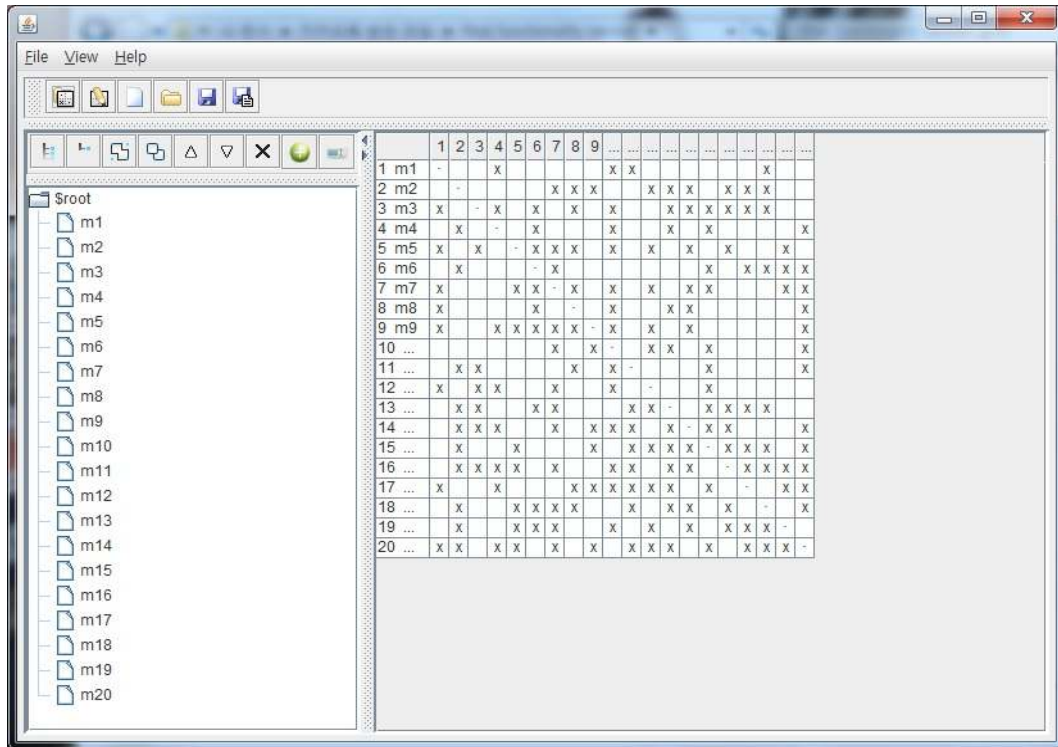
dsm파일을 만든 후 기존의 Titan 프로그램을 이용해 clsx 파일도 dsm파일과 같이 일정한 기준을 두어 여러가지 케이스로 생성하여 테스트를 시행할 수 있도록 했습니다.

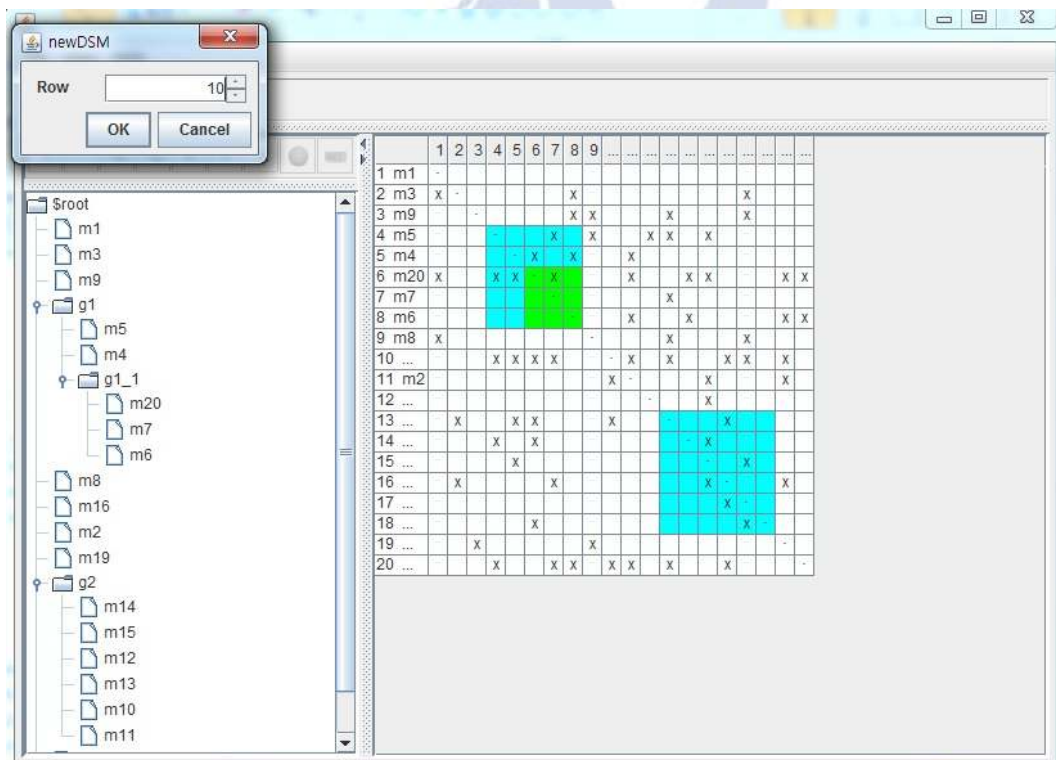
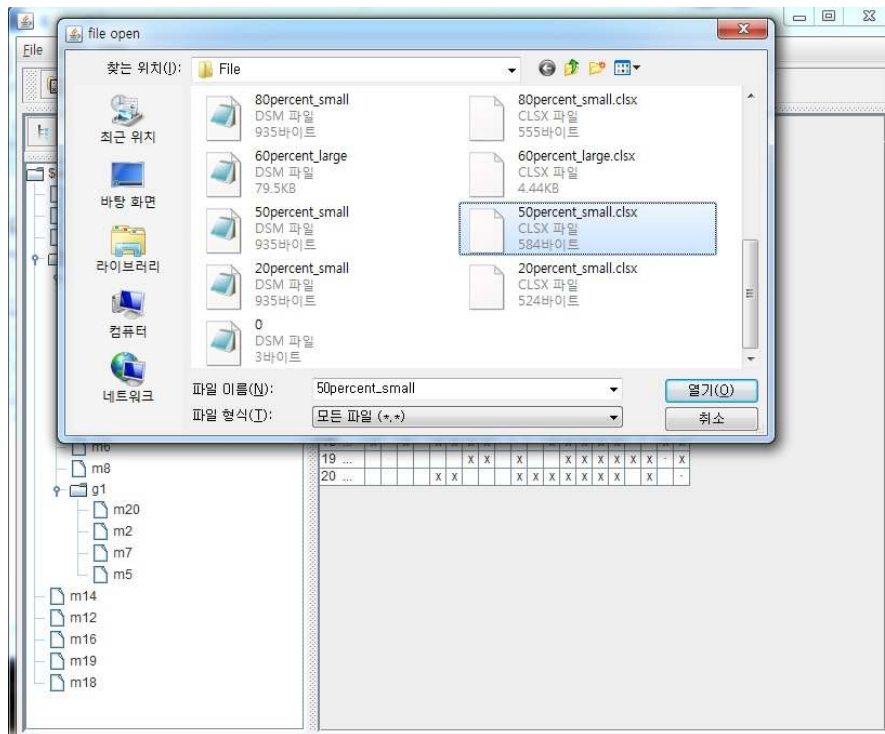
6) 객체지향적 개발방법의 적용

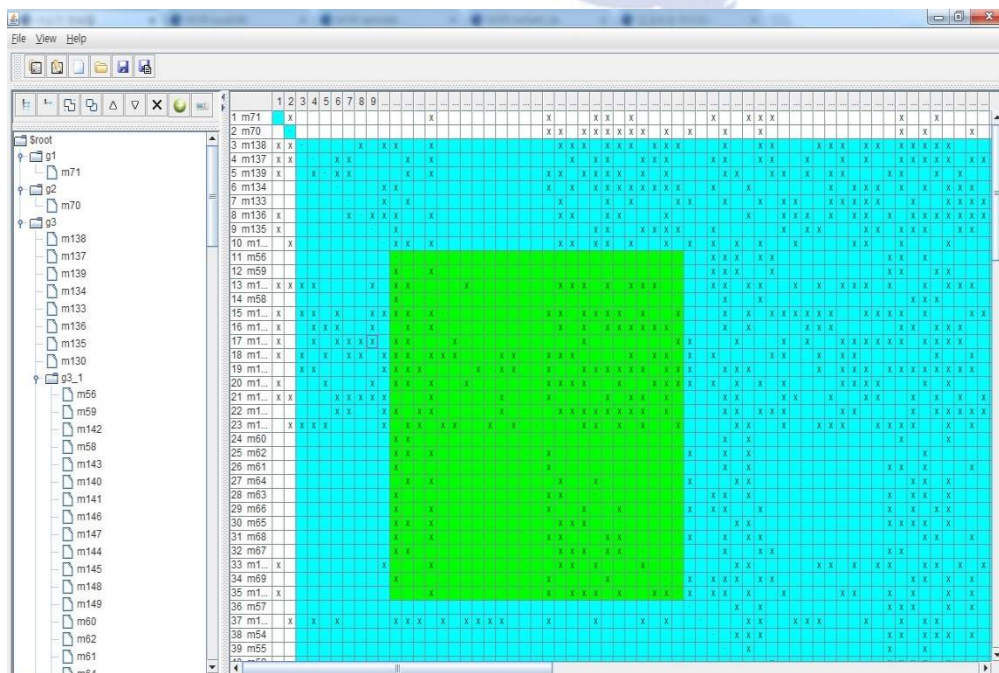
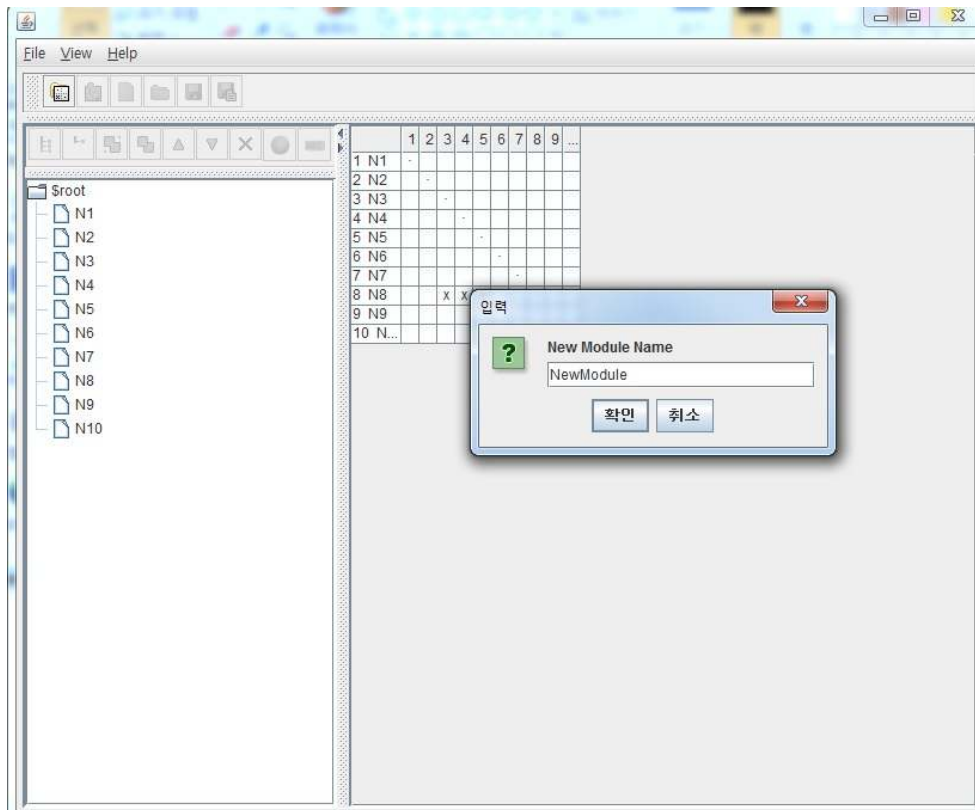
TreeNode 클래스를 따로 만들어 DsmModel이나 Cluster-Model 에서 데이터 관리를 위해 트리를 사용할 때 Tree-Node의 내부 구현을 세세히 알 필요 없이 해당 객체의 인터페이스만을 통해 데이터를 관리 할 수 있게 캡슐화 했습니다. TreeNode뿐 아니라 DsmModel이나 ClusterModel, Tree-Action 클래스도 각각 캡슐화를 시켜주었고, 이에 따라 ModuleController 입장에서 Main_View를 포함시켜 인터페이스에 따라 사용하며 프로그램의 동작을 제어 할 수 있게 설계했습니다.

또한 각각의 모듈에 대한 액션은 모두 TreeAction 클래스에 따로 구분해 놓음으로써 응집도를 높일 수 있는 방향으로 개발을 진행했습니다. 또한 결합도를 낮추기 위해 DSM과 Cluster를 각각 별도의 클래스로 나누어 설계하였습니다. 이러한 방식의 진행으로 SRP(Single Responsibility Principle)적인 특성 또한 만족시킬 수 있었습니다.

7) 프로그램 예제







8) 프로젝트를 통해 느낀 점

신우근 : 이번 프로젝트에서 제가 맡은 역할은 모델 부분 개발이었습니다. 개발이 mvc에 따라 나누어져 있어서 다른 부분을 신경쓰지 않아도 된다는 점이 개발하는데 있어서 편리한 점도 있었지만, 단점도 느꼈습니다. View에서 jtree로 모듈의 트리를 보여주려면 defaultmutabletreenode형태에서 변형해야 했는데 모델 부분 개발인 저는 그 점을 알지 못했습니다. 그 결과, model에서 제가 따로 구현한 다원트리를 jtree로 변형해 주는 class를 controller에 새로 만들어야 했습니다. mvc패턴에서도 개발에 있어서는 다른 분야에 대해서도 충분한 discussion이 필요하다는 것을 느낄 수 있었습니다.

경세준 : 이번 프로젝트를 하면서 제시된 요구조건에 따라 기능을 설계하고 클래스를 설계하는 것이 개발 이전에 얼마나 중요한지 느끼게 되었습니다. Waterfall Process인만큼 초기 프로그램 명세가 상당히 중요한데 그게 부실할 때 개발자 입장에서 애로사항이 많아지고 기초 명세가 탄탄할수록 후회가 없게된다는 사실을 직접적으로 느낄 수 있었고 즉흥적, 마구잡이로 개발하는 것이 아닌 큰 그림부터 그려가며 체계적으로 개발하는 것의 장점을 느낄 수 있었습니다.

이창수 : 저는 이번 프로젝트를 통해 물론 프로그램 개발에 대한 자체적인 항목에 대한 것에도 느낀 점이 있었지만 그보다도 소프트웨어 개발 시 프로젝트의 흐름에 대해 알 수 있었던 것이 가장 좋았던 것 같습니다. 위의 Waterfall Process를 선택한 이유에 대해 설명한 것처럼 지금까지 대부분의 프로젝트는 개발이 우선 시 되어 진행되게 되는 것에 반해 시행착오도 많고 부족했지만 새로 배우게 된 Waterfall Process라는 개념을 프로젝트를 통해 애매모호한 개념이 아닌 실제로 프로젝트를 진행해 가면서 간접적으로 체험 할 수 있었던 점이 좋았습니다.

배수열 : 이번 과제는 혼자하는 과제가 아니고 여럿이 같이 하는 과제여서 큰 프로젝트에서 사용하는 waterfall model을 사용해볼 수 있었습니다. 이 모델은 프로그램 작성을 시작하기 전에 무슨 기능이 필요한 지, 어떤 방식으로 동작할 지, 구조를 어떻게 할 지를 정해놓고 시작하다 보니 구조가 복잡하지 않고 전체적인 완성도를 높이기도 좋다고 느껴졌습니다. 학부과정에서 하는 프로그래밍은 대부분 작은 프로젝트들이라 waterfall model을 사용할 일이 없는데 이렇게 사용해 볼 수 있어서 좋은 기회였습니다.

황정근 : 이번 프로젝트에서 제가 맡은 역할은 뷰 부분 개발이었습니다. 사용자에게 보여지는 다양한 컴포넌트들을 배치하고 컨트롤러 부분과 상호작용을 잘 하기 위해서 인터페이스를 만드는 작업에 신경을 썼습니다. 또한 코드의 전체적인 질 향상을 위하여 디자인 패턴을 사용하여 Refactoring을 하였습니다.

이번 개발 중 JTable 과 관련된 부분이 예상 외로 복잡했는데 원하는 UI를 만들기 위하여 기존 컴포넌트들을 상속 받아 customizing 하는 방법들을 사용해 보았습니다.

개발을 진행하며 Git을 이용한 Source Code의 공유 및 버전관리를 사용해 보았는데 기능 자체는 매우 좋으나 아직은 익숙해지는데 시간이 필요할 것 같습니다.