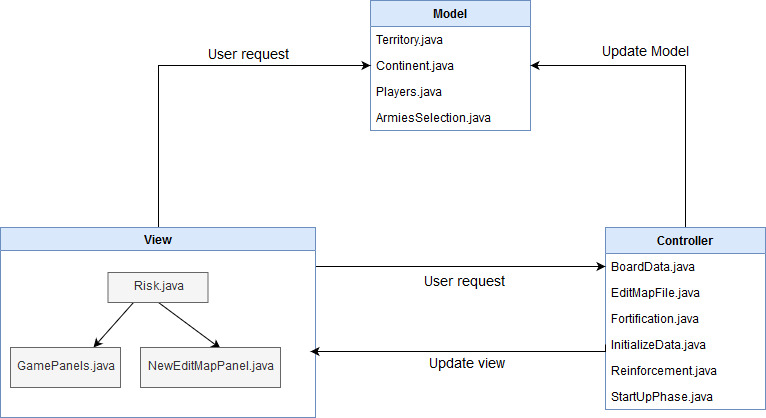
Architectural Design Document

**Introduction**

Design and develop a RISK Strategy game using suitable software design architecture with iterative software development approach to make a modular design and deliver several working coherent modules in small increments or builds. We implemented a Model View Controller (MVC) architectural design model. It was an effort to use extreme programming key features such as Pair programming, Collective ownership, Coding Standards and many more with the focus of System Metaphor.

**Overall Design Architecture**

****

**Module descriptions:**

|  |  |
| --- | --- |
| **Model** | **Description** |
| Terittory.java | This model class contains the Territory data generated from map file. |
| Players.java | This model class contains the data of Players. |
| Continent.java | This model class contains the Continent data generated from map file. |
| ArmiesSelection.java | This model class set the Number of initial armies depending on number of players playing. |
| **View** |  |
| Risk.java | A User-Interface to Launch the Game Play, Create Map or Edit Map and quit the Game. |
| NewEditMapPanel.java | A User-Interface to edit existing map file. |
| GamePanels.java | A complete User-Interface to play game, select players. |
| **Controller** |  |
| BoardData.java | This Controller used to parse map file, validate map file and generate map data from it. |
| CreateMapFile.java | This controller takes user input to create new map file from scratch and generate a new .map file. |
| EditMapFile.java | This Controller is used to edit existing map file and save the new edited map file. |
| InitializeData.java | This controller used to initialize data for game play and commence the start-up phase of RISK game. |
| Reinforcement.java | This controller is used to generate Reinforcement armies during Reinforcement phase. |
| StartUpPhase.java | Process the start-up phase, assign the territories to players in round-robin fashion. |
| **Exception** |  |
| InvalidMapException.java | This class is used to print InvalidMapException message if map is not valid. |
| **Validate** |  |
| MapValidator.java | This class is used to validate the map file. |
| **Test** |  |
| ReinforcementTest.java | Test all methods of Reinforcement.java class. |
| ArmiesSelectionTest.java | This test class test all methods of ArmiesSelection.java Model class. |
| ContinentTest.java | This test class test all methods of Continent.java Model class. |
| PlayersTest.java | This test class test all methods of Players.java Model class. |
| TerritoryTest.java | This test class test all methods of Territory.java Model class. |
| MapValidatorTest.java | Test all methods of MapValidator.java class. |
| RiskTestSuite.java | Run all test cases together. |

2.1 MVC Class Diagram

//

Fig 2

3. Model Implementation

//

//

//

Fig 3

4. View Implementation

//

//

//

Fig 4

5. Controller Implementation

//

//

//

Fig 5

6. Test Implementation

//

//

Fig 6

Rules followed of Ultra Board Games : <http://www.ultraboardgames.com/risk/game-rules.php>