

**SOEN 6441 (Advance Programming Practices)**

**Project: Risk: Strategy Build 1**

GitHub repository - <https://github.com/credo92/RiskConquer>

Team 19:

|  |  |
| --- | --- |
| Member Name | Student Id |
| Rahul Goyal |  |
| Garvpreet Singh |  |
| Gurpreet Singh | 40027930 |
| Vipul Srivastav | 40027931 |

**Submitted to**: Dr. Joey Paquet

Contents

[1.1 Scope: 2](#_Toc496190205)

[1.2 Architecture Style 3](#_Toc496190208)

[1.3 Risk Strategy. Modules Description 3](#_Toc496190209)

[1.3.1 1.1 Controllers 3](#_Toc496190210)

[1.3.2 1.2 Entity 4](#_Toc496190211)

[1.3.3 1.3 Utility 4](#_Toc496190212)

[1.3.4 1.4 Model 5](#_Toc496190213)

[1.3.5 1.4 Validate 5](#_Toc496190214)

[1.3.6 1.4 Main 5](#_Toc496190215)

[1.3.7 1.4 Exception 5](#_Toc496190216)

[1.4 Technologies and Tools used: 6](#_Toc496190217)

[1.4.1 Tools and technology used for the development of the game. 6](#_Toc496190218)

Introduction

This introduction provides an overview of the entire ***Software Architecture Document* for the Risk Strategy game**. It includes the purpose, scope, overview of the **system**.

## Scope:

## The scope of the build 1 is as per the instruction guidelines for the build:

## MapEditor: Covering the following below functionality, which is a form of a connected graph with proper interconnection between the continent and territories and abiding to the conquest map file format.

* Create a new map file
* Edit an existing map file
* Add/update/delete Continent
* Add/update/delete Territory
* Add/Delete Adjacent Territory
* Make sure that the integrity of the connected graph is maintained.
* Game Play: The game play covers:
  + Assigning territory to player
  + Player ability to assigning armies to each territory in round robin manner
  + Reinforcement phase, with proper calculation of armies
  + Fortification phase, with a valid fortification move.

## Furthermore, the scope of the build 2 is as per the instruction guidelines for the build:

## Conquer View : The conqueror player dominator view shows percentage of territories owned by player graphically

* + Dice Roll: The dice roll covers:
    - Roling the dice
    - Decide who wins on the basis of values of number of dice rolled by attacker and defender.
    - Validates the number of dices attacker/defender can use in single turn.

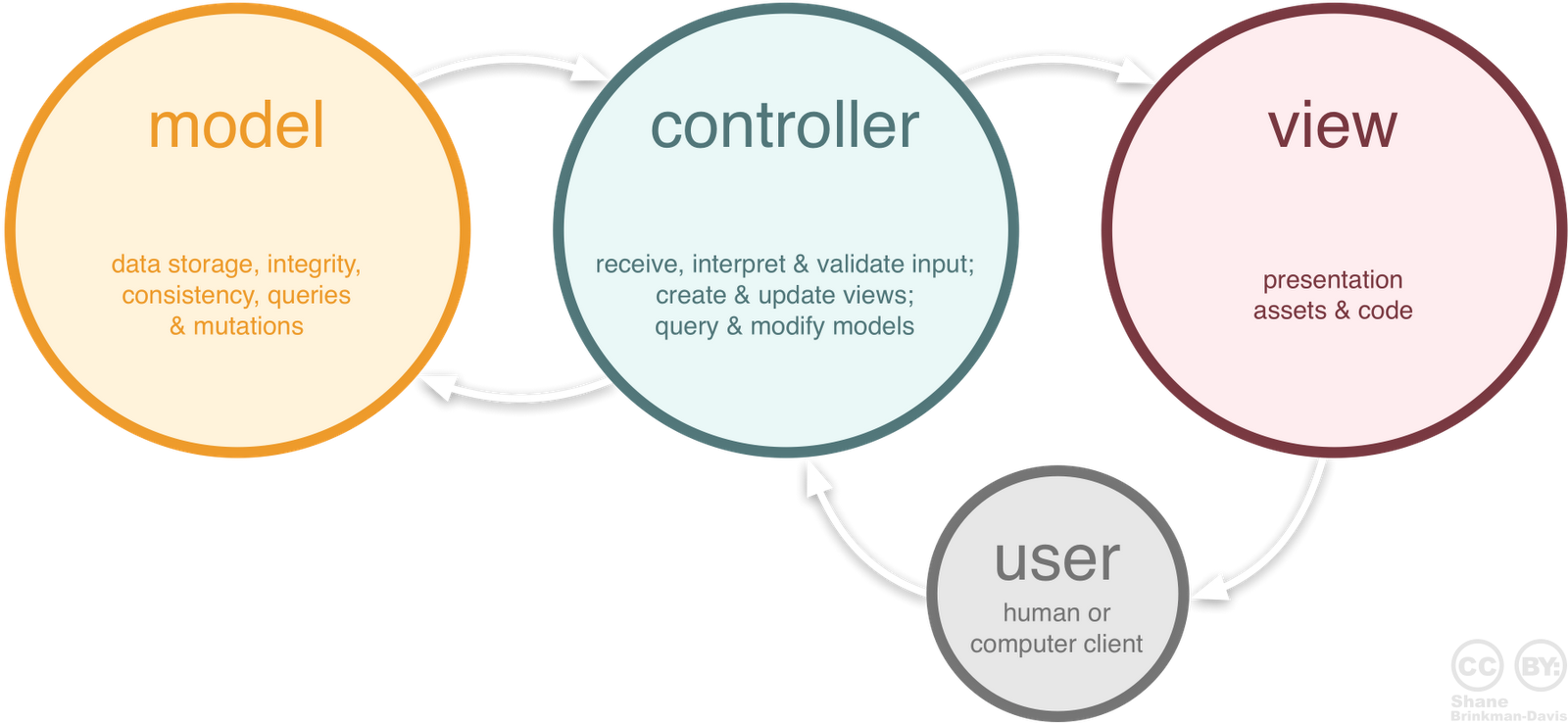
Player: Observer pattern is applied to this class containing methods mainly attack, reinforcement and fortification.

Card: Card functionality covers: -

* Assigning card to a player after he wins.
* Trade card player has valid set of cards
* Open card window before reinforcement.

## Architecture Style

Following the MVC design pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.



## Risk Strategy. Modules Description

### 1.1 Controllers

The Controllers folder includes the Game and Map controller modules of the risk game.

| **File\_Name** | **Description** |
| --- | --- |
| GamePlayController | The class acts as a mediator between the GameModel class and the GameView file. It captures all the user action like :   * Creation of player * Assigning armies * Assigning cards and updates the GameView based on the data changed published by the GameModel class. * It also serves to all the request issued by the GameView. |
| MapEditorController | This class act as a mediator between the GameModel and the MapEditorView. It captures all the user actions like:   * Add/Update/delete Continent/Territory/Adjacent Territories * It update the MapView based on the data changed published by the MapModel. * It also serves to all the request issue by the MapView |
| CardController | This class act as a mediator between the CardModel and the CardView. It captures all the user actions like:   * Creation of card * Creation of valid card sets * Card Trade and assign armies according to the rules of the RISK game. |
| DiceRollController | This class acts as a mediator between the DiceModel and the Diceview file. It captures all the user action like:   * Setting dice values after roll * Creation of dice as per number of armies * Other operations like move armies, cancellation of dice roll, continuation of dice roll. |

### 1.2 Entity

The Entity folder includes all the entities used in the game.

| **File\_Name** | **Description** |
| --- | --- |
| Map | It contains all the information of the Map file like: author, name  It contains a list of the continents that forms a map |
| Continent | It contains all the information related to the continent like , name , control value  It contain a list of all the territories that belong to a continent. |
| Territory | It contains all the information related to the territory like , name, x/y coordinates  It also maintains a reference to which continent the territory belongs  It also maintains a list of all the adjacent territory of this territory.  It contains the count of armies currently residing on the territory  It contains a reference to the player, indicating which player holds the territory. |
| Player | It contains all the information related to a player, like player name, list of territory assigned to the player.  It contains the number of armies associated with the player. |
| Card | It contains all the information related to card.  It contains territory name and card type which is a type of army. |

### 1.3 Utility

The Utility folder includes all the elements used in the map file.

| **File\_Name** | **Description** |
| --- | --- |
| MapFileParser | This class is responsible for reading the Conquest map file format and parsing in to Map object. It also checks for the validity of the data of the map file. |
| MapFileWriter | This class is responsible for writing the Map object to the file in the same format as read from the conquest file |
| MapUtil | Contains all the utility method like: pushing data to console, saving map object, opening a dialogue box. |
| Gameutil | Contains all the utility method like: pushing data to console, opening a dialogue box, disabling and enabling controls. |

### 1.4 Model

The Model folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| GameModel | This class represents the Game Model like creating players, assigning territory to the player, calculating the reinforcement armies, get the continents are owned by the players. |
| MapModel | This class represents the Map model and perform operation like , add/update/delete territory, add/update/delete continent. |
| PlayerModel | This class represents the Player Model. This class acts as an Observer as well as Observable. It consists of: -   * All the three phases of the risk game (Reinforcement, Attack, Fortify) * Validation for attacks and moves * Creation of Players |
| Dice Model | This class represents the Dice Model and perform operations: -   * Assigning values to each dice randomly * Validation before dice rolls * Assigning Territory and move armies when player wins |
| Card Model | This class represents the Card Model |
| Player Word Dominator | This class represents the Player Word Dominator and perform operations like populate world domination data. |

### 1.4 Validate

The Validate folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| MapValidator | This class represents the validation functionality of the map. It validate all the possible combination of Continents and territory and making sure that the remain as a connected graph. |

### 1.4 Main

The Main folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| CreateMap | Helps to load the new map screen |
| GamePlay | Helps to load the Game play screen |
| Main | Main entry point for the application |
| MapEditor | Help to load the map editor screen |
| MapOption | Helps to load the map options screen |

### 1.4 Exception

The Exception folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| InvalidMapException | Custom exception class to manage exception of the game related to map. |
| InvalidGameMoveException | Custom exception class to manage exception of the game related to the game play. |

## Technologies and Tools used:

### Tools and technology used for the development of the game.

| **Technology and Tools** | **Description** |
| --- | --- |
| Eclipse | IDE for the game development |
| Maven | Maven as a build automation tool to manage all project dependencies. |
| JavaFx | Library to control the UI components of the Risk Game |
| FXML Editor | To generate the UI components for the Risk Game |
| Junit 4 | Junit 4 for writing test cases |