OOP Project

# Related image**Project Design Report: Iteration 1**

CS 319 Object Oriented Software Engineering  
Eray TÜZÜN  
Group 1-I: Private Moon Inc.

Project Group Members

• Doğa Oruç – 21602022

• Çağrı Orhan - 21503102

• Fazilet Simge Er - 21602358

• Kaan Ünlü - 21602622

• Ege Turan – 21502441

Contents

[**Project Design Report: Iteration 1** 1](#_Toc3395914)

[1 Introduction 3](#_Toc3395915)

[1.1 Purpose of the system 3](#_Toc3395916)

[1.2 Design goals 3](#_Toc3395917)

[2 High-level software architecture 3](#_Toc3395918)

[2.1 Subsystem decomposition 3](#_Toc3395919)

[2.2 Hardware/software mapping 3](#_Toc3395920)

[2.3 Persistent data management 3](#_Toc3395921)

[2.4 Access control and security 3](#_Toc3395922)

[2.5 Boundary conditions 3](#_Toc3395923)

[3 Subsystem services 3](#_Toc3395924)

[4 Low-level design 3](#_Toc3395925)

[4.1 Object design trade-offs 3](#_Toc3395926)

[4.2 Final object design 3](#_Toc3395927)

[4.3 Packages 3](#_Toc3395928)

[4.4 Class Interfaces 3](#_Toc3395929)

[5 Glossary & references 3](#_Toc3395930)

# 1 Introduction

Private Moon Inc.’s Project Risk is a digital revival of the 1957 table-top classic Risk [1]. The original game is a 2 to 6 player game that rewards strategizing and a silver tongue.

The game consists of a game board, 6 sided dice, cards and pawns. The game board is usually a map of the world with named provinces. The provinces are grouped inside continents of different colours. There are two types of dice: Red and white. Also, there are 2 different card types: Province Cards and Secret Mission Cards. Finally, there are 6 different colours of pawns meant to be used by up to 6 players. These pawns can be found in three different shapes, usually dubbed infantry, cavalry and artillery, symbolising 1, 5 and 10 armies respectively.

The objective of the game for a player is either to conquer the entire world or if the players decide to play with secret missions, finish the objective on their secret mission card before the others do. For this, each player takes a turn one after another and act to conquer new lands. Each player turn has three phases:

* As their turn begins, the player gets a number of new armies equal to their total number of provinces divided by 3 rounded down, plus the value of their continent if they completely have conquered one plus the next number on the army table to the bottom of the game board if they have turned in 3 province cards of the same marking, or 3 province cards each of a different marking. Also if any of the cards the player has turned in that turn has a province they already have on pictured on it, they receive an extra 2 armies specifically for that province. The rest of the armies can be deployed however the player wants.
* Then, if wished so, the player can attack a province neighbour that neighbours one of their own provinces where the attacking province must have at least 2 armies in it. The player can attack as many times as they want on the same or on different provinces. On any singular attack, the attacking side can choose to roll up to 3 red dice (where maxDiceNumber = armiesOnAttackingProvinceNumber - 1) and in response the defending side can choose to roll up to 2 white dice (where maxDiceNumber = armiesOnDefendingProvinceNumber). All dice are rolled, and the maximum white dice number of attacker’s dice rolls are compared against the results of the white dice. The red dice have to score higher than white dice, or for every die that doesn’t, an army on the attacking side perishes. The opposite of this happens for every taken red dice value, an army on the defending side perishes.
* Finally, to fortify their position, the player can move as many armies as they want from one and only one province to one and only one other neighbouring province.

Up to three red dice can be used to attack a province and up to two white dice can be used to defend said province.

## Purpose of the system

Risk is a strategy game that incentivizes the players into thinking ahead. As however, the original Risk only consists of a single map with a relatively small province connection graph, strategies can start getting stale very quick, with some moves getting more and more familiar from previous playthroughs. This digital adaptation of Risk therefore aims to include more maps and the opportunity to play against AI or other players to keep the experience novel for longer, and allow for a more engaging gameplay that gives space to players to strategize more freely.

## 1.2 Design goals

Planning the design of the game works on a must have-should have-could have-won’t have basis. Design goals are distributed between these categories according to their priority and how essential they would be at improving the player experience and make this game more than its traditional predecessor. The MoSCoW analysis below shows the details of these goals:

The Game must have:

* A Map
  + Whose provinces can be selected accurately and viewed
  + On whose provinces armies can be deployed, viewed and sent to other provinces in transportation or attack
* Logic and visualization to show the number of armies received at the start of a turn
* Dice and card shuffling logic
* Province and secret objective cards
* Offline playing capability with friends
* Online playing capability over direct IP
* Offline playing capability over AI
* Limited multiple language support
* Limited colour blindness support

Should have:

* Mod Support
  + Filter based colour blindness support
  + Multiple resolution support
  + Custom map addition support
* More languages
* Better dice and card visualization

Could have:

* Music and sounds
* Map zoom
* Better province name visualization
* Server based online multiplayer capability
* AI, with flavour
* Game saving in offline games

Won’t have:

* Micro-transactions, loot boxes
* Accounts
* Game saving in online games

# 2 High-level software architecture

## 2.1 Subsystem decomposition

## 2.2 Hardware/software mapping

## 2.3 Persistent data management

## 2.4 Access control and security

## 2.5 Boundary conditions

# 3 Subsystem services

# 4 Low-level design

## 4.1 Object design trade-offs

## 4.2 Final object design

## 4.3 Packages

## 4.4 Class Interfaces

# 5 Glossary & references