****

**Object Oriented Software Engineering Project**

**Design Report**

**CS 319 Project: RISK: LOTR**

**Group 1J**

* **Miraç Vuslat Başaran**
* **Hazal Buruk**
* **Elena Çina**
* **Doğa Zeynep Germen**

**Intructor: Bora Güngören**

**Table of Contents**

1. **Introduction**
   1. **Purpose of The System**
   2. **Design Goals**
2. **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**

1. **Subsystem Services**

**1. Introduction**

**1.1 Purpose of the System**

Risk-LOTR is a desktop based game which we are designing with the primary aim to entertain people who are willing to play it. We were inspired by the traditional Risk table game, but we have adopted it and added some features from the famous Lord of Ring movies, hence the fans of these movies will enjoy playing this game even more. Additionally, developing this software system will help us understand and practice the object oriented programming concepts, gain experience on developing a real software system and improve our programming and team working skills.

**1.2 Design Goals**

* **Usability**

One of the main goals of our design is to develop a user friendly game. Menu will help users to access all the features of the game and through the help option user will be able to deliver information on tactics and logic of the game.

* **Performance**

*Response Time*

Risk-LOTR is an interactive game, hence we will be sure that the response time will not exceed a certain small threshold.

* **Well defined interfaces**

We aim to develop a game that will have well defined interface. All characters of the game such as different kind of units and factions. We will also provide animation for the execution of each phase of the game such as deployment, attacking and battle execution.

* **Extensibility**

We aim to build a system which can be updated without causing complication to the current system. It will be extensible in terms of its content, mechanics, interface and graphics. So, in the future we can enhance the system by adding additional new features.

* **Reliability**

Our goal is to build a reliable system which will not crush or give any run time errors, hence preventing players from any unpleasant experience.

* **Good Documentation**

We aim to well document all the work that we will do while developing this game.

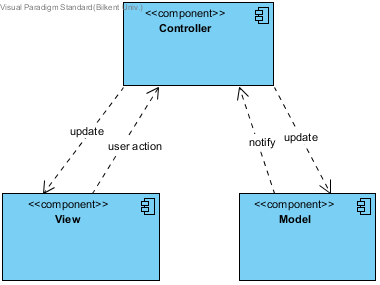
**2. Software Architecture**

This section includes a subsystem decomposition of our software project to make it understandable and easy to implement. Also, which hardware/software tools are needed and the database management is explained here. Finally, access control and security issues as well as boundary conditions are examined in their respective subsections.

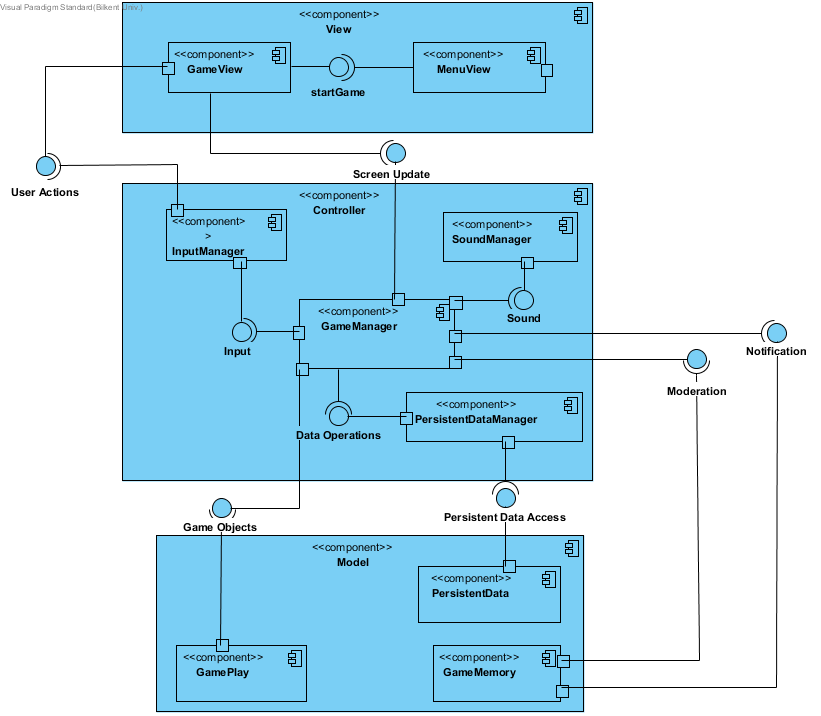
**2.1 Subsystem Decomposition**

RISK: LOTR, being a graphical strategy game, has a couple of interfaces to display the status of the game and to take user input as well as a complex logic that runs behind the scenes. To capitalize on this interface-logic separation, the game is designed on MVC (Model-View-Controller) pattern. This will help minimize coupling between main subsystems and maximize cohesion within those subsystems.

The “View” subsystem deals with handling user interfaces and getting user input. “Model” subsystem handles the data, rules and the logic of the application. Finally, “Controller” subsystem gets input from the other subsystems and updates them.



Below, the MVC architecture is expanded to show the subsystems of these three components and their interactions.



**2.2 Hardware / Software Mapping**

**2.2.1 General**

This game will be implemented in Java programming language using the latest JDK (8u121). Because our program doesn’t include high graphics and intense physical operations while it’s running, most of the computers will be able to run it. All of the graphical and physical calculations will be implemented by using Java libraries. However, most of the graphics will be uploaded to the game after they designed with other graphic design programs. Because of excessing graphical content and excessing the short term memory may require a time, the operations may require a short time but we expect them to be fast enough so that the player cannot understand the require time.

Because every new game requires new conditions and different players as enemies, we won’t collect the previous scores in anywhere and because the game only requires a little amount of graphics, we don’t plan to use any database system in our game. Because the current states of the game will only requires a little amount of memory, it’s unnecessary to create a database system.

**2.2.2 Input / Output System**

User will need mouse clicks to select its action, finish the phases. However, the keyboard will be used only when the player enters her name or enters the number of the soldiers that she wants to buy or when she wants to escape the game. Otherwise, mostly the mouse will be used. The output will be monitored by monitor and the sound effects and music will come out via speakers. The only required hardware are them. There is no additional hardware system.

**2.2.3 Memory**

The PC memory will be enough to meet the requirements of the game.

**2.2.4 Processor**

The visual subsystems will only require GUI and some pictures designed with the help of photoshop, it will not require heavy computational power. So that, there will be no rendering process while running. Game engine will only make some basic calculations to pick which graphic or picture will be monitored in that specific time. It will check the new condition in every change and change the current output. So that, the PCs with single processors will do the required operations easily.

**2.3 Persistent Data Management**

The only required persistent data management will be the game base data like help script, some pictures to be monitored and they will be kept in documents or in .txt files. There will be nothing to keep persistent after playing the game because this game will not have a save button or high scores table.

**2.4 Access Control and Security**

Our game will not include an authentication system. Each player will have the same rights to play the game. Namely, no player has special rights. Each player will have equal access to the game. There will be no need for an authentication system, as there is no difference in restrictions for players.

**2.5 Boundary Conditions**

**Initialization**

CASE: RISK: LOTR will be initialized if user launches the program via .exe file.

**Termination**

CASE 1: RISK: LOTR can be terminated if user clicks Quit Game.

CASE 2: RISK: LOTR can be terminated if user presses ESC key on the keyboard while running the game on Windows or on other OSs.

**Failure**

CASE: If there appears a system failure or a hardware failure (.txt file not found, .txt file corrupted etc.), there is no way to prevent the crash of the program; in this case, user will lose all progress since our game does not have a progress-saving feature, in other words, user will not be able to continue the game from the point game crashed.