****

**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.

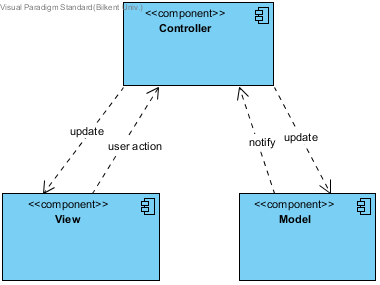
**2. Software Architecture**

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.

