

Overall, your document is well written and your descriptions are brief and up-to-the-point. Well done!
I included one comment about your schema on the last page.

Web Diplomacy

Design Document

10/16/2017

Version 1.0



Diplomats

Connor Wright

Sean Kallagher

Hugh McGough

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

Introduction	3
Architecture Design	3
Overview	3
Design Details	3
Subsystem Design	3
User Interface	4
Game Engine	4
Database	4
Data design	4
User Interface Design	5
Testing Plan	5
References	5
Appendix A: UML Subsystem Block Diagram	5
Appendix B: SQL Schema	6

I. Introduction

This document is intended to illustrate our process for implementing web-based Diplomacy and describe the tools and procedures used therein. Diplomacy is a strategic board game of wits, cunning, negotiation, and deceit. This project is aimed at providing a platform to play the game of Diplomacy remotely with ~~their~~ friends.

In Section II, the overall architecture of the project is described.

In Section III, further details of each subsystem are provided.

In Section IV, the testing methods for the project are outlined.

In Section V, we provide useful references related to the project.

Document Revision History

Rev 1.0 2017-10-16 Initial version

II. Architecture Design

II.1. Overview

The major subsystems of this project include the User Interface, Database, and Game Engine. The relations between these subsystems are described in Appendix A. The Game Engine processes information from the UI and in the Database to determine the results of the players actions. The User Interface takes information from the user and sends it to the game engine using http requests. The database stores game information such as the map unit position and control points. The Database communicates with the Game Engine thorough queries. We reduced coupling having a linear form of communication. Every subsystem has at most two other subsystems it is talking to. Cohesion was reduced by giving each subsystem a clear task that is its only job. The model we are applying to our program is the Model View Controller. The User utilises the User Interface to send orders to the Game Engine (controller). The Game Engine then processes the orders and updates the Database (model). After updating the Database the Game Engine then updates the User Interface (view).

III. Design Details

III.1. Subsystem Design

This section provides more detail about each subsystem:

III.1.1. User Interface

The User Interface will use a combination of HTML, Javascript, and CSS to display and receive information about the game from the user. Communications between the User Interface and the Game Engine will be using HTTP requests through Flask. The types of communication will be defined as follows.

1. send order (POST): This will send an order to the game engine
 - a. URL: /api/send_order
 - i. JSON: { "unitID"= int, "targetName"=string, "orderType"=string }
 - b. Response: 200 if valid, 418 if invalid
2. Confirm orders (POST): confirms the orders of the player.
 - a. URL: /api/confirm_order
 - i. JSON: N/A
 - b. Response: 200 if valid, 418 if invalid
 - i. JSON: contains all of the troops and their positions

III.1.2. Game Engine

The Game Engine will be written in Python and utilize Flask for handling HTTP requests as well as Psycopg2 to generate PostgreSQL queries. The communications between the Game Engine and the User Interface are outlined above. Internally, the Game Engine will process requests from the User Interface and determine the outcome of a myriad of actions. The most important piece of the Game Engine will be the action resolution algorithm which will determine the resulting game state given a set of actions from all players.

III.1.3. Database

The Database uses a PostgreSQL database to store all of the critical information required for the Game Engine to operate. SQL queries will be used to access and modify this data from the Game Engine. The Database will be used as a representation of the current state of all games as well as a method to validate orders the users make.

III.2. Data design

Appendix B contains a schema diagram outlining the tables used by the Database subsystem. Internally, the Game Engine will rely heavily upon the Database as its primary data structure and will not utilize any other data structures.

III.3. User Interface Design

User Interfaces have not currently been designed nor implemented. The protocols for how the User Interface and Game Engine will communicate have been implemented.

IV. Testing Plan

To test the Game Engine, we will be using Python's unittest library. This will provide an easy to use testing framework for the backend Python code in the form of unit tests for modules such as the order resolution algorithm. This could also be easily expanded to perform API testing.

V. References

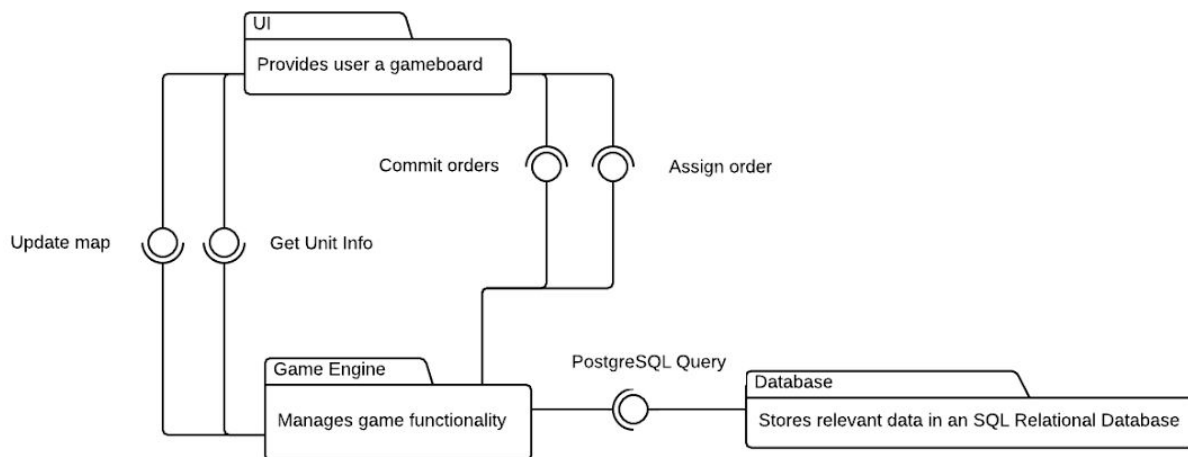
The Rules of Diplomacy, Stephen Majesk, <http://faculty.washington.edu/majeski/426/sim1.html>

PostgreSQL 10.0 Documentation, <https://www.postgresql.org/docs/10/static/index.html>

Psycopg Documentation, <http://initd.org/psycopg/docs/>

Python Unittest API, <https://docs.python.org/3/library/unittest.html>

VI. Appendix A: UML Subsystem Block Diagram



VII. Appendix B: SQL Schema

