***Build 1 Architecture Design Document***

**Introduction**

Our project’s first build mainly focuses on few of the core functionalities such as map editor, new map controller, startup phase, reinforcement phase and fortification phase.

The design pattern that we used is Model-View-Controller. The reason for using MVC design pattern is that it allows different programmers to work on different business logic simultaneously. This makes the work faster and thereby saves time. Our build is filled with multiple functionalities, and keeping in mind its compatibility to Extreme Programming, we decided to use Incremental approach.

**Development Model**-Agile (Incremental)

**Design Pattern**- MVC

**MVC Architecture**

It is an architecture pattern that separates the application into 3 components- the model, the view, the controller. Each component is built to handle specific aspects of an application.

**Model**: It corresponds to all data related logic that the user works with. It represents the data that is being transferred between view and controller components. Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

In our project Build 1, Model contains class files.

* GameDetails- It contains methods to check the validity of the map, store continents and territories from the map, create player objects, set player colors and no. of player armies based on no. of players and distribute armies to the players.
* Player- It contains the player name, and territories, card and no. of armies held by the player.
* Territory-It holds the territory details such as name of the territory, continent and player it belongs to and no. of armies it has.
* Continent-It contains the territory list belonging to that continent.

**View**: It is used for all user interface logic of application. This depends on data received from model or sometimes from controller.

In our project Build 1, View contains class files.

* Sample-It will display main user interface.
* Newmap- It will display new map window containing text area, save button and cancel button.
* MapEditor- It will display map editor window containing text area to display map file contents and save and cancel buttons.
* Gameparameters- It is a dialog that will have a checkbox for no. of players, player characters and file chooser to choose the map file.
* Loadgame- It will display game window where map is displayed using canvas, textfield to display reinforcement armies for each player and finish button to send control to next player in round robin manner.

**Controller**: Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

In our Project Build 1, Controller contains class files.

* Controller- It is responsible for controller main interface.
* GameparametersDialogController- It is responsible for controlling dialog containing user preferences like no. of players, player characters and choosing the map.
* MapEditorController-It is responsible for controlling map editor window.
* NewMapController- It controls new map editor window for example, what to do when user presses save button, cancel button etc.

**Extreme Programming features used:**

1. Planning-

A general plan for all the 3 builds was made and a detailed plan for the current build was built on the basis of which the whole team started the work.

1. Sustainable pace-

We all have been meeting regularly and discussing the work that is to be done. We have been helping each other in difficult and time-consuming parts of the project to reduce the pressure on a particular team member and ensure timely completion of a particular part of the build.

1. Small Releases-

We developed and deployed partially working application with certain features for every release regularly.

1. Collective Ownership-

Everyone in the team was allowed to make changes in any part of the code. Any team member could fix the errors in the code by just informing about the changes to the member who made the code.

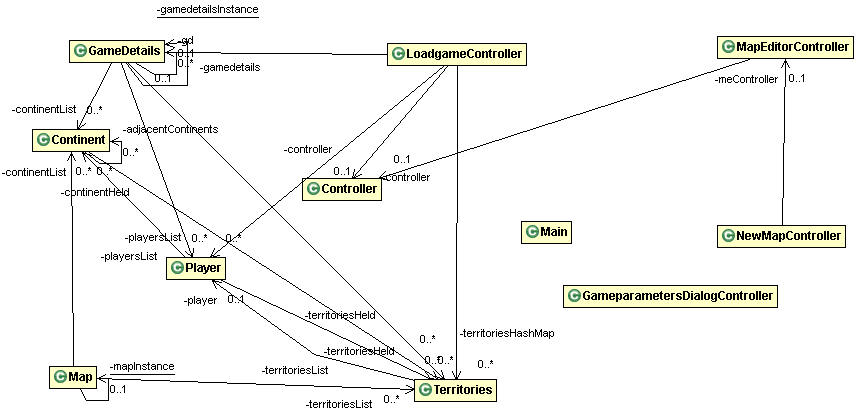
1. Continuous Integration-

With a distributed repository such as GIT, we were able to commit our code frequently thereby, testing and integrating it.



1. **MVC Architecture**

**UML Diagram**



1. **Class Diagram**

**References**

1. <https://www.whizsolutions.co.uk/advantages-using-mvc-framework-web-development/>
2. <https://www.tutorialspoint.com/extreme_programming/extreme_programming_additional_features.htm>
3. https://moodle.concordia.ca/moodle/pluginfile.php/3506605/mod\_resource/content/1/Lecture%202.pdf