SOEN 6441

Advanced Programming Practices

Architectural Design Document

Team 20

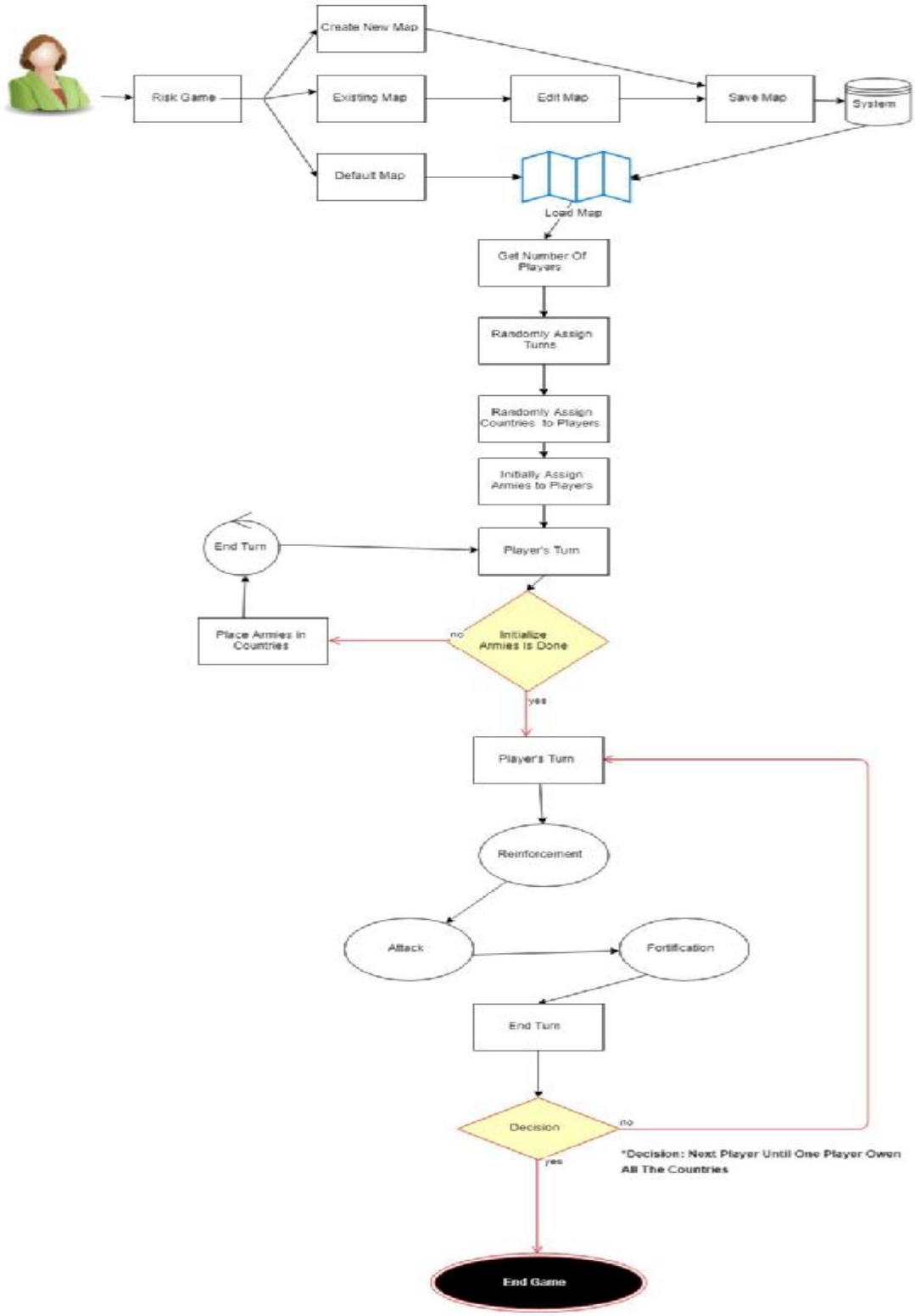
Konstantin Chemodanov…. 40049285
Farzaneh Jamshidi…..…...…40075234
Saman Safaei……. 40119399
Ali Molla Mohammadi …….40042597
Negar Ashouri……40071530

March 2019

Department of Computer Science and Software Engineering

Introduction:

The purpose of this project is to build a single-page web application of Online Risk game implementing an MVC (Model, View, controller) architectural design model. We designed the project as web client-server application, where server running backend is implemented using Java language and client interface is using Angular.

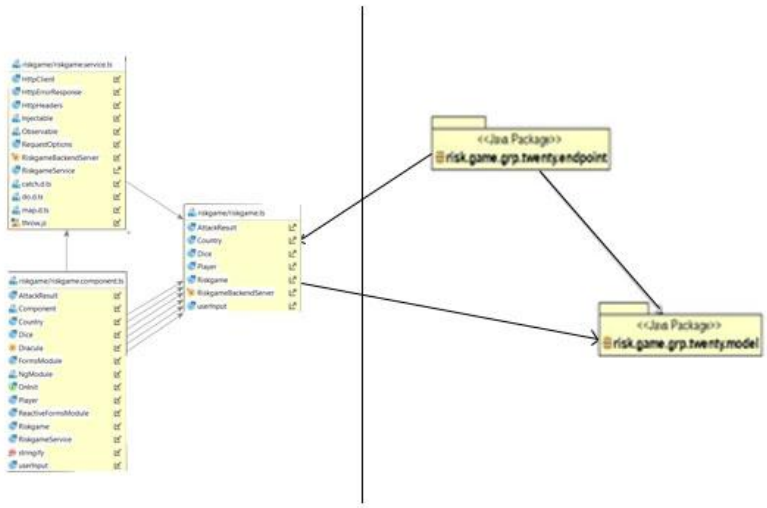


We have followed the extreme programming approach in development of our project:
Pair Programming: After dividing the project into modules, two programmers or more worked on the same tasks in order to have better and quicker results and desired functionalities.
Simple Design: In order to avoid faults, the software was designed as simple as possible and any extra complexity was removed.
Continuous Integration: Bit bucket is a web-based version control repository for development of projects that use Git revision control systems, same as this project. Using Bit bucket made all the programmers on the same branch and eased the Maintenance of concurrent changes and errors detections by automating integrating tasks.
Collective Ownership: Using a repository also made all members able to make changes anywhere in the code anytime.
Testing: Several unit tests have been written in order to make sure each unit is functioning as it should.
Coding Standards: Oracle coding standards were accepted by all programmers and was performed in the whole project. Like naming conventions, file organization, commenting conventions and etc. Following these standards made the code more readable and understandable for everyone which led us to a more maintainable and sustainable code that will be helpful in our further builds.

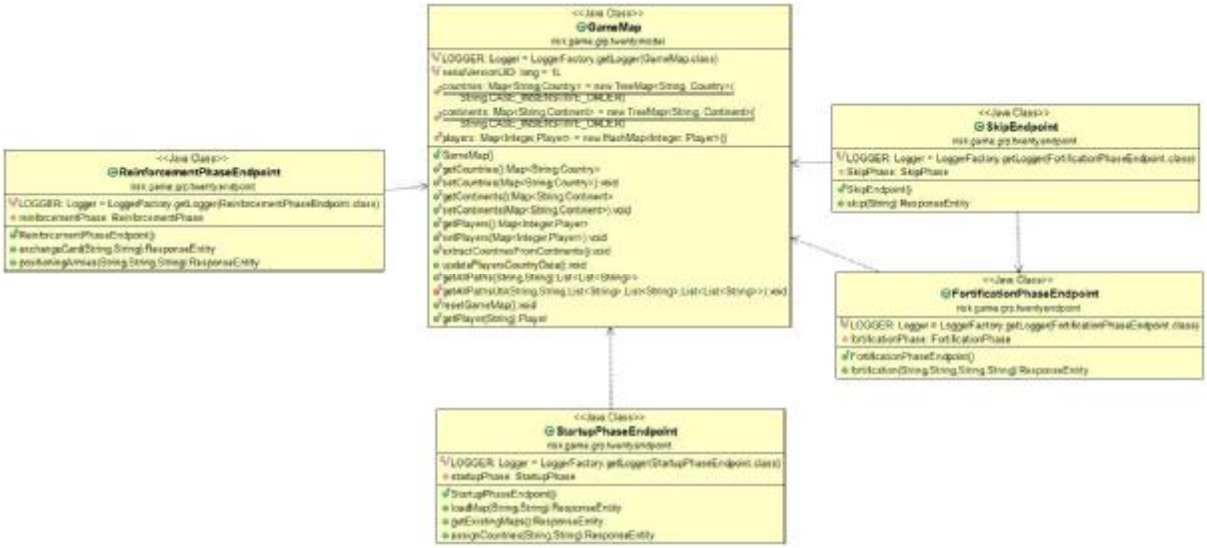
Architectural design:

The purpose of the MVC design pattern is to separate content from presentation and data-processing from content.
Separation of concerns (SoC) is a design principle for separating a computer program into distinct sections so that each section addresses a separate concern, which means each component should not do more than one thing. [2]
Model View Controller architecture aims for separation of Concerns, by dividing the components into three parts: Model, View, Controller.

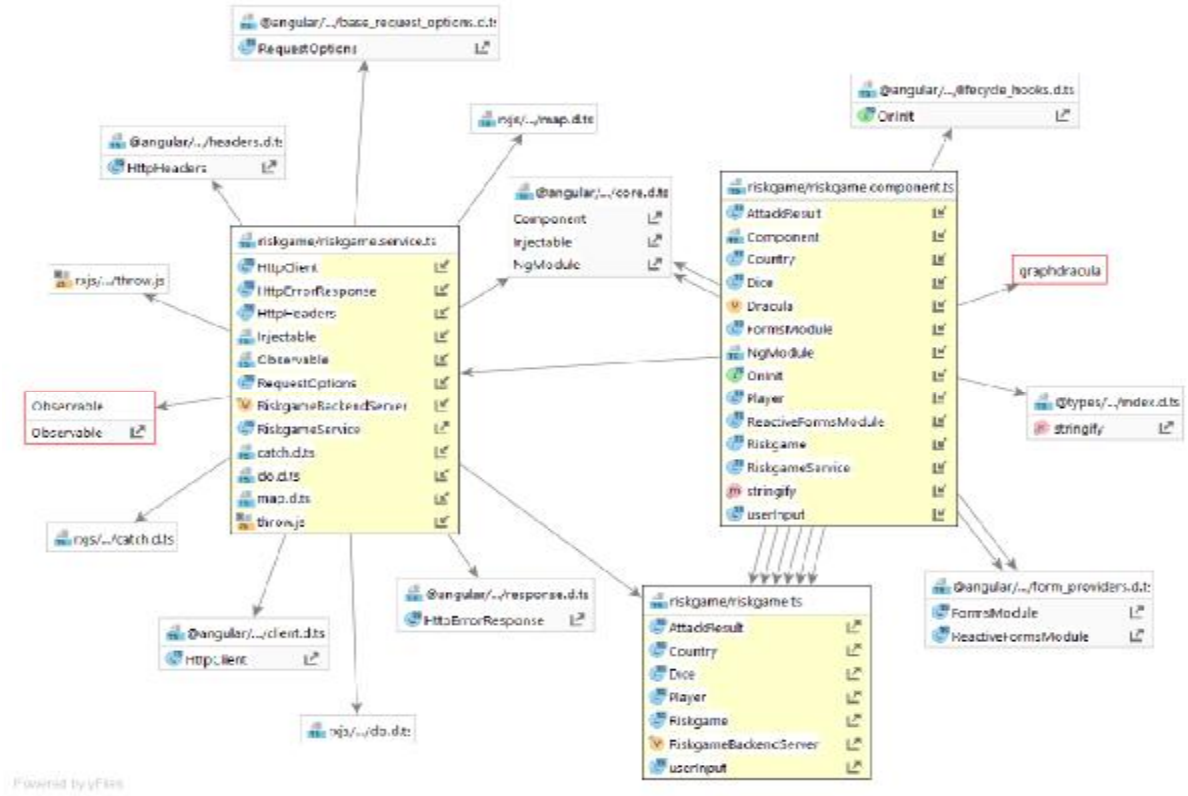
As shown in this picture how MVC has been implemented in this project:



Models *(*GameMap, Player, Continent, Country, etc.) manages the data of the application domain. If the model gets a query for change state from the Views, they respond to the instruction via Controllers.



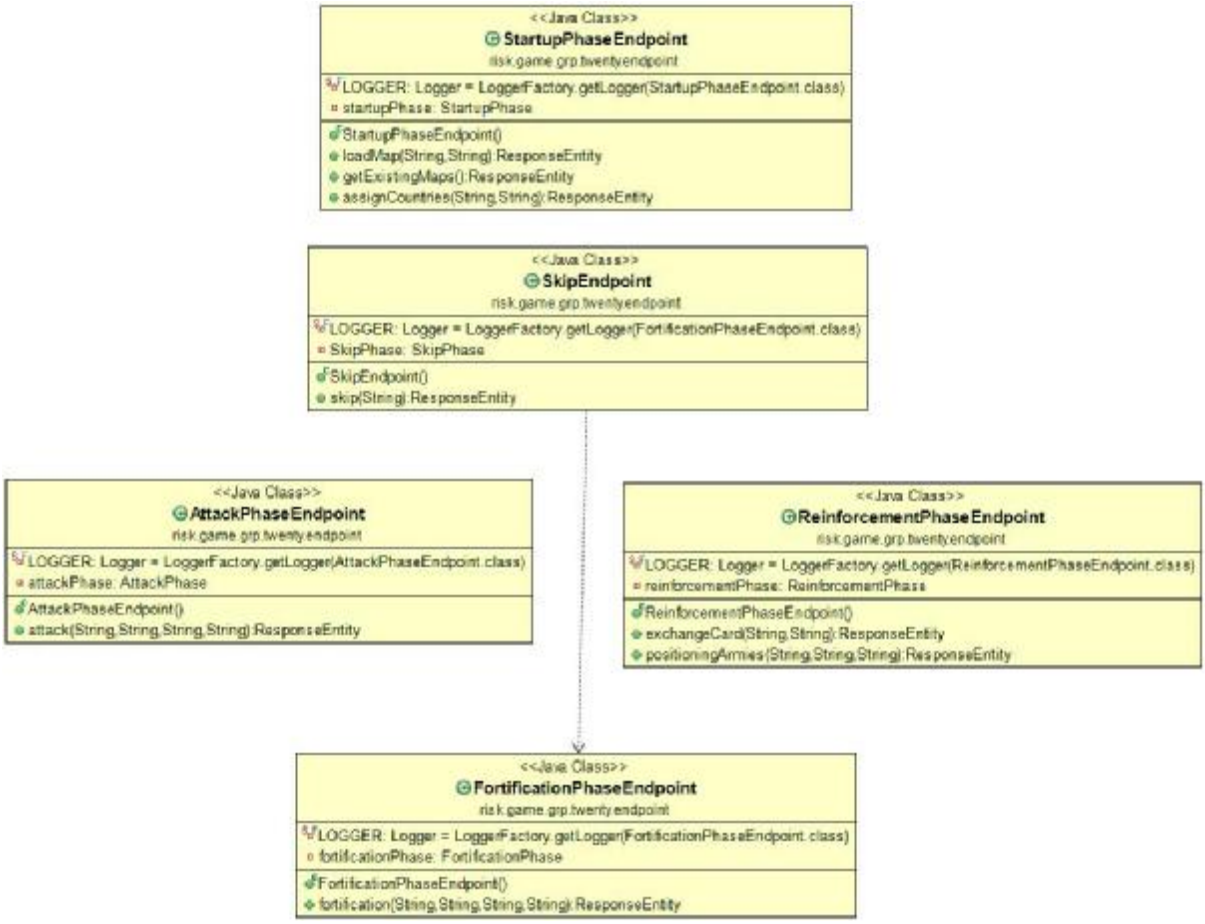
Views (SelectMap, mapGraph etc.) visualize the models and have interaction with the user. If the model data changes, the view must update its presentation as needed. Angular is a web application framework which is used to create the user interface of this project.



Controllers (StartupPhaseEndpoint, FortificationPhaseEndpoint, ReinforcementPhaseEndpoint, etc.) receive user input and initiate a response based by making calls on appropriate model objects.

The controller translates the user's interactions with the view it is associated with, into actions that the model will perform that may use some additional/changed data gathered in a user-interactive view.

Controllers are also responsible for spawning new views upon user demands. [1]



Pros of Angular

MEAN stack

Angular belongs to the so-called MEAN stack (MongoDB + Express + Angular + Node.js). We decided to rely on an existing firm Angular ecosystem which allowed us quickly develop frontend GUI for the Risk project.

One of the advantages of using Angular is that its components can be thought of as small pieces of an interface that are independent of each other. Imagine that you have a simple application with a list of items and a corresponding search box to retrieve the items by word matches. The box with listed names, the search box, and the main sheet where the other two boxes are placed are all considered separate components in Angular.

We selected Angular because it is built mainly around the Model-View-Controller (MVC) architecture. This allows for building UIs with many moving parts and, at the same time, streamlines the development course for engineers. There are many benefits of such architecture:

Reusability. Components of a similar nature are well encapsulated, in other words, self-sufficient. Developers can reuse them across different parts of an application. This is particularly useful in enterprise-scope applications where different systems converge but may have many similar elements like search boxes, date pickers, sorting lists, etc.

Readability. Encapsulation also ensures that new developers – who've been recently on boarded to a project – can read code better and eventually reach their plateau of productivity faster.

Unit-test friendly. The independent nature of components simplifies unit tests, quality assurance procedures aimed at verifying the performance of the smallest parts of the application, units.

Maintainability. Components that are easily decoupled from each other can be easily replaced with better implementations. Basically, your engineering team will be more efficient in maintaining and updating the code within the iterative development workflow.

Angular is written using Typescript language, which is basically a superset for JavaScript. It fully compiles to JavaScript, but helps spot and eliminate common mistakes when actually typing the code. While small JavaScript projects don't require such enhancement, the enterprise-scale applications challenge developers to make their code cleaner and verify its quality more often.

Angular has some components important for our project: Angular Universal and Ivy renderer.

Angular Universal is a service that allows for rendering applications view on a server instead of client browsers. Google provides a set of tools to either pre-render your application or re-render it for each request by a user.

Ivy renderer. A renderer is an engine that translates templates and components into JavaScript and HTML that browsers can understand and display.

Another important aspect of Angular is that it has been around for quite a long time, so we were able to use many tools developed by community. [3]

To integrate GUI frontend with Java backend we decided to use **JAVA Restful web-services**.

For our project RESTful architectures and implementations have following benefits:

Ease of integration - RESTful applications can be easily integrated in the web as they use HTTP methods explicitly

Increased Scalability - RESTful interactions are stateless and caching semantics are built into the protocol).

Evaluability - All resources are identified by URIs which provides a simple way to deal with the evolution of a system.

Reliability - RESTful systems typically achieve reliability through idempotent operations.

Transfer XML, JavaScript Object Notation (JSON)

RESTful architecture allowed us to work separately on Backend and Frontend from the very early phase of the project. We were able to invoke RESTful requests directly from the web-browser while Frontend was under development. This independence was extremely important as we divided different development tasks between Frontend and Backend teams. RESTful architecture is easy to debug and trace on both sides of the project software. Testing RESTful architecture can also be automated in the future phases of the project. [4]

References

[1] [Online]. Available: En.m.wikipedia.org. (n.d.). Separation of concerns. [online] Available at: https://en.m.wikipedia.org/wiki/Separation_of_concerns .

[2] [Online]. Available: Tutorialsteacher.com. (n.d.). MVC Architecture. [online] Available at: https://www.tutorialsteacher.com/mvc/mvc-architecture ..

[3] [Online]. Available: AltexSoft. (n.d.). The Good and the Bad of Angular Development. [online] Available at: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development.

[4] [Online]. Available: MuleSoft. (n.d.). RESTful Web Services using Mule ESB | REST Services for Integration. [online] Available at: https://www.mulesoft.com/resources/esb/restful-web-services-esb-rest-services-integration.